LA VOZ DE LOS EXPERTOS

INCLUYE CD-ROM

Daniel L. Morrill

Configuración de sistemas Linux

Domine el funcionamiento de los sistemas Linux.

Conozca a fondo diversas distribuciones Linux: Red Hat, Slackware y Debian GNU/Linux.

Aprenda a desplegar en su sistema el software más actual, como el servidor HTTP Apache, el sistema de versiones concurrente (CVS) y la plataforma de desarrollo Java JDK de Sun Microsystems.

a! apress

ANAYA MULTIMEDIA

Índice de contenidos

Introducción **22**

Lo que este libro cubre 23
Lo que aprenderá 24
Lo que no aprenderá 25
Lo que necesitará 26

Parte I. Antedecentes **27**

1. La fisiología de Linux **28**

Los orígenes de Linux 30
La naturaleza de un sistema operativo 32
Recursos 32
Núcleo 33
Bibliotecas de sistema 33
¿Es Linux un sistema operativo? 34
Linux y GNU 34
GNU y Linux: el sistema operativo de los campeones 36
El programa ls 37
gzip, bzip2, tar y vim 37
El secreto del éxito 38
El zen de la plataforma 39
El debate GNU/Linux o Linux 40
Referencias 41

2. Introducción al código abierto **42**

La naturaleza del software 43
Código abierto frente a código libre 46
Software de código abierto en acción 46
Aplicaciones horizontales 47
Desarrollo cooperativo de software 48
Licencias de código abierto 49
La Licencia Pública General (GPL) de GNU 51
La licencia BSD 52
La Licencia Pública Menos General (LGPL) de GNU 53
La Licencia Pública Mozilla 53
Otras licencias 54
Licencia para programar 54

Parte II. Distribuciones Linux **55**

3. La naturaleza de una distribución **56**

Una instantánea del Continuo Linux 57
Los objetivos de una distribución 59

Disección de una distribución Linux 60
Núcleo Linux 61
Formato de empaquetado 63
Gestor de paquetes de Red Hat 63
Archivos Deb 64
Formato de empaquetado frente a distribución 64
Estructura del sistema de archivos 65
Directorios 65
Estándar de jerarquía del sistema de archivos 66
Guiones de arranque del sistema 69
Versiones de las bibliotecas de sistema 71
Escritorio X Window 73
Transparencia de red 73
Modelo de GUI en capas 74
Escoger un servidor X, conjunto de componentes, y gestor de ventanas 76
Software de aplicación 78
Base Estándar de Linux 79
Resumen 80

4. Red Hat Linux 82

Antecedentes y filosofía de Red Hat 83
Núcleo Linux 84
Formato de empaquetado 85
Resumen técnico 86
Convenciones de nombrado de archivos 86
Arquitecturas 87
Formato del archivo del paquete 88
Uso de RPM 89
Modos del programa 89
Permisos de acceso 89
Comprobaciones de dependencia 90
Información sobre paquetes no instalados 90
Instalación de paquetes 92
Actualización de paquetes 93
Información sobre paquetes instalados 93
Verificación de los paquetes instalados 94
Reconstrucción de paquetes 95
Desinstalación de paquetes 96
Creación de nuevos paquetes 97
Prestaciones adicionales 97
Mecanismo frente a interfaz de usuario 98
El programa up2date de Red Hat 98
Otros programas GUI 98

Ejemplos de uso de RPM ... 98
Estructura del sistema de archivos ... 99
Guiones de arranque del sistema ... 101
Comprensión de los niveles de ejecución iniciales ... 102
Estructura de archivos y directorios ... 104
El paquete SysVinit ... 104
El paquete initscripts ... 105
Seguir la pista a una ejecución ... 107
Guión de configuración del nivel de ejecución ... 110
Configuración manual del sistema ... 112
Uso de las herramientas de Red Hat ... 114
/sbin/service ... 114
/sbin/chkconfig ... 114
/sbin/shutdown ... 115
Bibliotecas centrales del sistema ... 115
El sistema X Window y el escritorio ... 116
XFree86 ... 117
GNOME y KDE ... 118
Aplicaciones de usuario ... 119
Comprensión de la distribución central ... 120
CD de código fuente y documentación ... 121
Ediciones de Red Hat Linux ... 121
Encontrar software adicional ... 122
Idiosincrasias de Red Hat Linux ... 122
Uso el cargador de arranque GRUB ... 122
Gestión de software similar ... 123
Actualización a un núcleo estándar ... 123
RPM kernel-headers ... 125
Enlace simbólico /usr/src/linux-2.4 ... 125
Construcción del núcleo a partir del código fuente ... 126
Entorno de usuario ... 127
Red Hat Linux e Inetd ... 128
Dominio de los directorios de configuración de Red Hat ... 129
Trabajar con Kudzu ... 130
Instalar fuentes TrueType ... 131
Garantizar la seguridad del sistema ... 132
Uso de un cortafuego ... 132
Gestión de valores de seguridad con PAM ... 133
Derivados de Red Hat Linux ... 134
Mandrake ... 134
Caldera ... 134
TurboLinux ... 135
Resumen ... 135

5. Slackware Linux 8.0 136

Antecedentes y filosofía de Slackware 138
Núcleo Linux 139
Formato de empaquetado 140
Resumen técnico 140
Administración de paquetes instalados 140
Instalación de paquetes 141
Actualización de paquetes 141
Desinstalación de paquetes 142
Creación de nuevos paquetes 142
Mecanismo frente a interfaz de usuario 142
Estructura del sistema de archivos 143
Guiones de arranque del sistema 144
Estructura de archivos y directorios 144
/etc/inittab 145
/etc/rc.d/rc.S 147
/etc/rc.d/rc.K 147
/etc/rc.d/rc.M 147
/etc/rc.d/rc.0 y /etc/rc.d/rc.6 148
/etc/rc.d/rc.modules 149
/etc/rc.d/rc.inet1 y /etc/rc.d/rc.inet2 149
/etc/rc.d/rc.sysvinit 150
Otros archivos 150
Modificación de los guiones de arranque 150
Seguir la pista a una ejecución 151
/etc/rc.d/rc.S 151
/etc/rc.d/rc.M 151
Otras acciones 152
Bibliotecas centrales del sistema 152
El sistema X Window y el escritorio 153
XFree86 153
GNOME y KDE 154
Aplicaciones de usuario 154
Aplicaciones incluidas 154
ZipSlack 155
Idiosincrasias de Slackware Linux 155
Configuración de inetd 156
El archivo /etc/services 156
El archivo /etc/inetd.conf 157
Instalación de un nuevo servicio basado en inetd 159
Garantizar la seguridad del sistema 159
Inhabilitar los servicios inetd 161
Inhabilitar el demonio portmap 161
Inhabilitar el demonio de impresión lpd 162

Inhabilitar el servidor SMTP sendmail 163
Inhabilitar el servidor HHTP Apache 163
Otros servidores 164
Personalización del entorno de usuario 164
Cambiar el entorno de escritorio de X Window de un usuario 165
Añadir hardware nuevo 166
Derivados de Slackware Linux 167
SuSE Linux 167
Peanut Linux 167
Resumen 168

6. Debian GNU/Linux 3.0 170

Antecedentes y filosofía de Debian 172
El Contrato Social Debian 172
Las Políticas Debian 173
Núcleo Linux 174
Empaquetado Debian 174
Resumen técnico 174
Uso del programa dpkg 175
Uso del programa apt-get 176
Uso del programa dselect 178
Uso del programa tasksel 178
Entrar en detalles 179
Administración de paquetes con el sistema de Debian 179
Trabajo con dpkg 179
Gestionar fuentes de paquetes con apt-get 181
Administración de paquetes con dselect y tasksel 182
Configuración de paquetes Debian 183
Actualización del sistema 185
Manipulación de archivos de paquete Debian 186
Administración del código fuente 187
Instalación de paquetes no Debian 187
Mecanismo frente a interfaz de usuario 188
Guiones de arranque del sistema 189
Encontrar los directorios SysV 189
Trabajar con las herramientas de Debian 190
Uso de update-rc.d 190
Uso de invoke-rc.d 191
Diferencias en /etc/inittab 191
Bibliotecas centrales del sistema 192
El sistema X Window y el escritorio 193
Aplicaciones de usuario 193
Idiosincrasias de Debian GNU/Linux 194
Visita al directorio /etc 194

El directorio /etc/default 194
El directorio /etc/network 195
El directorio /etc/dpkg 197
Comprensión del directorio /etc/alternatives 197
Configuración del gestor de visualización X 198
Garantizar la seguridad del sistema 199
Cierre de servicios SysV innecesarios 199
Eliminación de servicios inetd 200
Uso del programa update-inetd 200
Derivados de Debian GNU/Linux 201
Resumen 201

Parte III. Instalación de software 205

7. Instalación y configuración de software 206

Comienzo 207
Lectura de la documentación 208
Aceptar la licencia 208
Instalación de software en seis pasos sencillos 209
Elección entre una instalación fuente o binaria 209
Comprensión de las plataformas hardware y las familias de procesadores 209
Opciones de configuración del software 210
Toma de la decisión 211
Decidir la ubicación 211
Instalación en /usr/local 212
Instalación en /opt 213
Toma de la decisión 214
Construcción del software 214
Especificación de opciones en tiempo de compilación 214
Realizar la compilación 215
Configuración de los parámetros del software 215
Parámetros en línea de comandos 216
Herramientas y ayudantes específicos de la aplicación 216
Archivos de configuración global 217
Directorios para depositar archivos de configuración 217
Archivos o directorios planos 218
Servidores arrancados por inetd 218
Bibliotecas de usuario 219
Configuración del entorno de usuario 219
Enganche con el sistema operativo 219
Herramientas de instalación de software 220
Uso de autoconf de GNU 220
Uso de las herramientas de Red Hat 222

RPM y arquitecturas 222
Construcción de RPM fuente 222
Comandos RPM 223
Contenidos del directorio /usr/src/redhat 223
Uso de las herramientas de Slackware 224
Uso de las herramientas de Debian 226
Resumen 228

8. La shell segura OpenSSH 230

Instalación de OpenSSH 232
Opciones en tiempo de compilación 233
Selección de una ruta de instalación 233
Selección de las opciones de configuración 233
Instalación a partir del código fuente 234
Configuración de OpenSSH 234
Configuración del cliente ssh 236
Habilitar el reenvío por agente SSH 236
Inhabilitar la comprobación estricta de claves de máquina 238
Configuración del servidor sshd 239
Habilitar el reenvío de conexión con X Window 239
Resumen de la configuración de OpenSSH 240
Instalación de OpenSSH en diferentes distribuciones 240
Instalación de OpenSSH en Red Hat Linux 241
Instalación de OpenSSH en Slackware Linux 242
Instalación de OpenSSH en Debian GNU/Linux 243
Resumen 244

9. Módulos de autenticación enlazables (PAM) 246

Instalación de PAM 248
Configuración de PAM 249
Construcción del archivo de configuración 249
El elemento servicio 250
El elemento tipo 251
El elemento control 252
El elemento módulo 253
El elemento argumentos 253
Las pilas PAM 253
Análisis de un archivo de ejemplo 255
Instalación de PAM en diferentes distribuciones 257
Instalación de PAM en Red Hat Linux 257
Instalación de PAM en Slackware Linux 257
Instalación de PAM en Debian GNU/Linux 258
Resumen 258

10. Biblioteca SOCKS Dante 260

Instalación de Dante 264
Preparación de la ruta de instalación 264
Compilación e instalación 265
Resumen de la instalación 265
Configuración de Dante 265
Encontrar el archivo de configuración 266
Obtención de los parámetros de red 266
Formato del archivo de configuración 267
Opciones globales 268
Creación de un bloque de ruta 269
Un archivo de muestra 270
Instalación de Dante en diferentes distribuciones 271
Instalación de Dante en Red Hat Linux 271
Instalación de Dante en Slackware Linux 271
Instalación de Dante en Debian GNU/Linux 272
Resumen 272

11. El servidor HTTP Apache 274

Instalación de Apache 276
Opciones en tiempo de compilación 277
Elección de un directorio de instalación 278
Habilitación del soporte de módulos de carga dinámica 280
Elección de un MPM 281
Otras opciones 282
Completar la instalación 283
Configuración de Apache 283
Exploración del directorio de instalación 284
Programas en el directorio bin 285
Archivos de configuración en el directorio conf 288
Archivos de anotaciones en el directorio logs 289
Directorios de contenidos 289
Directorios de documentación 290
Otros directorios 291
Personalización de Apache 291
Instalación de Apache en diferentes distribuciones 293
Instalación de Apache en Red Hat Linux 294
Elección de la versión correcta 294
Inicio del servidor Apache durante el arranque del sistema 294
Hacer apachectl compatible con chkconfig 295
Instalación de Apache en Slackware Linux 296
Instalación de Apache en Debian GNU/Linux 296
Resumen 296

12. Sistema de versiones concurrentes (CVS) 298

Instalación de CVS 301
Opciones en tiempo de compilación 301
Instalación a partir del código fuente 303
Configuración de CVS 304
Creación del directorio repositorio 305
Creación de cuentas de usuario 306
Confiar en las cuentas del sistema 306
Compartir una única cuenta de usuario 307
Establecimiento de permisos 307
Parámetros para el caso de autenticación de sistema 308
Parámetros para el caso de cuentas virtuales 310
Establecimiento de parámetros en tiempo de ejecución 311
Instalación de CVS en diferentes distribuciones 312
Instalación en Red Hat Linux 312
Actualización del archivo /etc/services 312
El directorio /etc/xinetd.d 313
Construcción del archivo de configuración 314
Administración de xinetd 316
Instalación en Slackware Linux 317
Instalación en Debian GNU/Linux 317
Resumen 318

13. Kit de desarrollo de Java de Sun Microsystems 320

Instalación de J2SDK 323
Configuración de J2SDK 325
El directorio /opt/java 325
Instalación de múltiples JDK 326
Instalación de archivos JAR 326
Configuración del entorno de usuario 327
Fijar CLASSPATH 327
Fijar PATH y JAVA_HOME 329
Juntarlo todo 330
Instalación de J2SDK en diferentes distribuciones 331
Instalación en Red Hat Linux 331
Instalación en Slackware Linux 332
Instalación en Debian GNU/Linux 332
Resumen 333

Parte IV. Casos de estudio 335

14. Construcción de un sistema de escritorio 336

Comienzo 337
¿A quién se dirige este ejemplo? 337

Definición de un escritorio 338
Contenidos de este capítulo 339
Construcción del sistema 340
Elección de los paquetes 340
Detalles de la configuración 340
Uso de aplicaciones de escritorio y productividad 341
Entorno de escritorio 342
La elección: KDE 342
Las alternativas: GNOME, GNUStep, Sawfish, BlackBox, y FVWM 342
Productividad ofimática 343
La elección: OpenOffice 343
Las alternativas: Koffice, Abiword, HancomOffice, Applixware, y Wordperfect Office 344
Finanzas personales 345
La elección: GNUCash 345
Las alternativas: Kapital 346
Navegación web 346
La elección: Konqueror y Mozilla 346
Las alternativas: Galeon y Opera 348
Extensiones del navegador 348
Uso de extensiones con los navegadores 348
La extensión Macromedia Flash 349
La extensión Java 350
Soporte para hardware 350
Uso de una disquetera ZIP 351
Configuración de una disquetera ZIP 351
Montaje de una disquetera ZIP 353
Uso de dispositivos USB 354
Marco de trabajo principal 355
Controlador de la controladora USB 355
Controladores específicos de dispositivo 356
Conexión en caliente del núcleo 356
Uso de una cámara digital 356
Uso de un lector de tarjetas flash 357
Uso de un reproductor MP3 358
Uso de otros dispositivos USB 358
Uso de una grabadora de CD 359
Comprensión de la emulación SCSI 359
Red Hat Linux y SCSI 359
Configuración de su grabadora 360
Uso de tarjetas gráficas 3D 360
Exploración de las opciones multimedia y de entretenimiento 362
Trabajo con cámaras digitales 362
Manipulación de fotografías con GIMP 363

Creación de discos compactos 363
Uso del programa mkisofs 364
Uso del programa cdrecord 365
Uso del programa xcdroast 365
Reproducción de archivos de audio digital 365
El programa mpg321 366
El programa ogg123 366
El sistema multimedia X 367
Reproducción de archivos de vídeo digital 367
El reproductor multimedia avifile 367
El programa smpeg 368
Otros programas 368
Juegos 369
Implementación de la conectividad y el acceso remoto 370
Acceso remoto al escritorio 370
Provisión de otros servicios 371
Resumen 372

15. Un entorno de desarrollo de software corporativo 374

Comienzo 375
¿A quién se dirige este ejemplo? 375
¿Para qué se utiliza este sistema? 376
Construcción del sistema 376
Comienzo con el escritorio 377
Elección de los paquetes 377
Detalles de la configuración 378
Herramientas de administración y red 378
Compresión de la red local 378
Uso de DHCP en el sistema 379
Instalación del servicio portmap 380
Uso de servicios de información de red (NIS) 380
Modificación del archivo /etc/sysconfig/network 381
Modificación del archivo /etc/yp.conf 382
Uso de servicios del sistema de archivos de red (NFS) 382
Preparación de los servicios de apoyo 383
Compresión del montador automático 384
Administración de activos 386
Provisión de acceso remoto como root 386
Entorno de desarrollo 387
Instalación de las herramientas centrales de desarrollo 387
Configuración del entorno Java 388
Instalación de herramientas adicionales 388
Advertencias y mejoras 392
Uso del servidor HTTP Apache 394

Ejecución de aplicaciones J2EE con Tomcat 394
Interoperación corporativa 395
Uso de OpenOffice para documentos de empresa 396
Funcionamiento de VMware 397
Cuestiones de seguridad 398
Compresión de la seguridad de la red local 398
Protección de la estación de trabajo 399
Configuraciones alternativas 400
Uso de un servidor de aplicaciones alternativo 400
Uso de otros lenguajes web 401
Desarrollo de software tradicional 401
Resumen 401

16. Construcción de un cortafuego de red 404

Comienzo 406
¿Qué es un cortafuego? 406
Prácticas seguras 407
¿A quién se dirige este ejemplo? 408
Sustitución del cortafuego por defecto 409
Elementos de un cortafuego Linux 409
Minimizar la exposición 409
Bloquear a los malos 411
Enmascaramiento 411
Servicio de red 414
Construcción del cortafuego 414
Hardware necesario: tarjetas Ethernet 415
Problemas de las tarjetas Ethernet 415
Nombrado de los dispositivos Ethernet 415
Configuración de las interfaces 417
Selección de un medio de instalación 418
Reducción de la lista de paquetes 419
Adición de paquetes de seguridad 421
Preparación de las cuentas de usuario 422
Configuración de un nuevo núcleo 422
Creación de los guiones de inicio 423
Selección de los servicios de arranque 423
Presentación de un guión iptables 424
Análisis del guión 427
Comprensión de las reglas de cortafuego 432
Arranque del cortafuego 433
Detención del cortafuego 436
Comprobación del estado del cortafuego 437
Reinicio del cortafuego 437
Validación de la instalación 437

Resumen y limitaciones 438
Soporte a las máquinas internas 439
Instalación de DHCP para máquinas internas 439
Instalación de un servidor de nombres local 444
Lo que ya ha conseguido 446
Vigilancia y mantenimiento del sistema 446
Configuración del sistema de anotaciones del sistema 447
Configuración del servidor de anotaciones interno 447
Perforación del cortafuego local 448
Configuración de syslog en el cortafuego 449
Vigilancia del tráfico del cortafuego 450
Clasificación de paquetes 452
Cuidado y mantenimiento de su cortafuego 452
La lista de correo Bugtraq 453
Listas de fabricantes 453
Otras fuentes 454
Seguridad en profundidad 454
Investigación de técnicas de seguridad avanzada 455
Uso del sistema de detección de intrusiones Tripwire 455
Integración de una LAN inalámbrica 457
Mejora del servidor DNS 458
Arranque desde medios de sólo lectura 459
Configuración de una VPN mediante SSH 459
Resumen 461
Una despedida 461
Bibliografía 462

Parte V. Apéndices 463

Apéndice A. Reuniéndolo todo 464

Escapada en solitario: configuraciones adicionales 465
Educación continua 466
Lista del núcleo Linux 466
Listas de correo de las distribuciones 467
Seguir la pista del trabajo activo y diversas iniciativas 467
En busca de la seguridad 467
Mejoras del núcleo 468
Recorte y compresión del sistema 468
Resumen 469

Apéndice B. Contenido del CD-ROM 470

Índice alfabético 473

En 1991, un estudiante de ciencias informáticas finés llamado Linus Torvalds necesitó un sistema operativo similar a Unix para sus estudios. En vez de comprar una de las versiones comerciales (¡y porque andaba algo escaso de dinero!) se planteó el interesante ejercicio mental de escribir su propio sistema Unix. Linus comenzó a trabajar con otros desarrolladores y en poco tiempoe lograron el corazón de un sistema operativo y el inicio del fenómeno Linux.

Desde entonces, y en particular en el último par de años, la fama de Linux (y, por supuesto, su notoriedad) ha crecido de manera increíble. Algunos podrían incluso decir que el ascenso de Linux ha alcanzado el nivel de "palabra-rumor". Se ha convertido en un tópico común en las salas de estrategia de muchas compañías dedicadas a la tecnología. "¿Tenemos una estrategia Java?" "Olvídate de eso, ¡¿tenemos una estrategia *Linux*?!" A pesar de ello, en el fondo, el éxito de Linux comenzó (y, en un sentido real, termina) con sus usuarios. Sin una estrategia de mercado y sin respaldo corporativo, el advenimiento de Linux fue preparado únicamente por sus millones de usuarios satisfechos.

Lo que este libro cubre

Tanto si eres un usuario de un sistema Linux como si no, es probable que hayas escuchado un montón de cosas sobre lo que estos sistemas pueden hacer por ti. Desde el servidor al ordenador de escritorio, y de aplicaciones corporativas a usos

domésticos, la variedad de software que Linux soporta es excitante, pero también puede resultar desconcertante, especialmente para usuarios novatos y de nivel intermedio.

Por eso, cuando se trata de sistemas Linux, la pregunta más común que escucho es, "¡Acabo de instalar Linux en mi ordenador! ¿Qué hago ahora?" Los creadores de distribuciones Linux han simplificado tanto las tareas de instalación y de administración que casi cualquiera puede instalar un sistema Linux. Ahora mismo, el auténtico truco está en aprender cómo doblegar el sistema ante tus deseos. Los sistemas Linux ofrecen una inmensa cantidad de flexibilidad y potencia, pero también presentan gran complejidad que puede ser díficil de dominar.

Si es un usuario de Linux experto, podrá hacer que el sistema haga más o menos lo que quiere de él si es suficientemente persistente. Esta maestría proviene de la comprensión del modo en que se disponen los sistemas Linux (y en general los sistemas Unix) y de los patrones y técnicas más habituales para conseguir realizar cosas.

Este libro tiene dos objetivos: transmitir un entendimiento de estas técnicas y patrones, y proporcionar suficiente conocimiento práctico como para que sea capaz de abordar cualquier tarea que quiera completar, incluso si no se encuentra cubierta directamente en este libro. En esencia, este libro pretende mostrarle cómo están construidos los sistemas Linux y cómo "piensan" y actúan.

Aclaremos esto, este libro no es una referencia. Existen tantas cosas que puede hacer con un sistema Linux que una referencia detallada resultaría enciclopédica en su extensión, que no es el caso de este libro. Tampoco es un tutorial. Se han escrito numerosos libros excelentes que pueden guiarle en el proceso de instalar un sistema Linux, configurarlo y manejarse con él. Este libro le mostrará el paso siguiente : cómo convivir y trabajar con un sistema Linux.

Por lo general, este libro no le explicará cada paso de una tarea, puesto que asume que ya sabe cómo completar las tareas más básicas. Por ejemplo, este libro no describe cómo extraer *tarballs* (archivos .tar.gz); asume que ya sabe cómo hacer esto. Si es un usuario con alguna experiencia en sistemas parecidos a Unix, o incluso con otros sistemas como Windows de Microsoft, esto no tendría que suponer ningún tipo de problema. Si se está iniciando o es su primer contacto con un sistema parecido a Unix, puede que necesite una referencia adicional como complemento a a este libro sobre aspectos como la instalación de su sistema y la realización de tareas básicas.

Lo que aprenderá

Este libro está divido en cuatro partes. La primera parte, "Antecedentes", cubre algunos de los asuntos básicos relacionados con los sistemas Linux, tales como la naturaleza del software libre y de código abierto, y algunos aspectos generales acerca de lo que incluye una distribución Linux. La primera parte le proporcionará una sana introducción al "Zen" del mundo Linux.

La segunda parte, "Distribuciones Linux", introducirá el concepto de una distribución Linux. Tras desentrañar el armazón de una distribución, comentaré tres distribuciones actuales en detalle como ejemplo. Los capítulos de esta segunda parte muestran diferentes acercamientos, técnicas y soluciones al problema de construir un sistema Linux. Esta parte le ayudará a penetrar bajo la superficie de un sistema Linux e hincar el diente en la carne que se esconde debajo. Si lee atentamente este capítulo, habrá conseguido sentirse bastante cómodo con los detalles de algunas de las distribuciones Linux más populares y estará bien equipado para enfrentarse con otros sistemas Linux (o incluso sistemas similares a Unix) que se encuentre más adelante. También será capaz de entender rápidamente cualquier cambio realizado sobre las distribuciones existentes en las nuevas versiones que aparezcan.

La tercera parte, "Instalación de software", parte del material de la segunda parte para mostrar cómo instalar software en su sistema. Esto quiere decir que, mientras que en la segunda parte se muestra cómo es el sistema, la tercera mostrará cómo ajustar y personalizar el sistema dentro de los límites impuestos por la distribución. Los ejemplos contenidos en estos capítulos demuestran la mayoría de las técnicas más comunes para instalar software, de modo que si lo lee y asimila debería estar preparado para comenzar a doblegar el sistema a su voluntad, de modo que pueda realizar un trabajo real con el software que necesite.

La cuarta parte, "Casos de estudio", parte a su vez del material de la tercera parte y presenta varios casos de estudio distintos de sistemas Linux. Cada capítulo de esta parte trata sobre un uso real y actual de un sistema Linux. Estos capítulos realmente ejemplifican el punto en que las ruedas tocan la carretera, mostrando cómo el mismo sistema base puede ser adaptado para una amplia variedad de necesidades. Si lee u hojea esas páginas, habrá avanzado mucho hacia el objetivo de alcanzar pericia en el manejo de los sistemas Linux.

Entonces, ¿qué es lo que aprenderá? Bueno, si se lo plantea, realmente va a construir una casa de conocimientos. La primera parte realiza un reconocimiento del terreno, la segunda parte construye los cimientos, la tercera parte da forma a las paredes y al techo y la cuarta parte instala todos los detalles y convierte la casa en un hogar. Al menos, eso es lo que se intenta. ¡Espero que esté de acuerdo!

Lo que no aprenderá

Después de que haya estado usando su sistema basado en Linux durante un tiempo, inevitablemente descubrirá algo para lo que este libro no le ha preparado. (Por ejemplo, puede desear instalar fuentes TrueType en un sistema Linux, que no es un tema comentado explícitamente en este libro). Es evidente que ningún libro puede cubrir todas las cuestiones posibles ; en cualquier caso, partimos de esa certeza. Por esta razón, el libro no intenta ser una referencia enciclopédica (como ya dije), sino que en su lugar trata de ofrecerle las habilidades básicas que necesita para ser autosuficiente. Por ello, cuando encuentre su siguiente obstáculo infran-

queable, recurra a las técnicas generales y a las vías de aproximación de este libro, en lugar de buscar ejemplos específicos. Con este conocimiento y una mínima experiencia, dominará su sistema en poco tiempo.

A pesar de esto, aquí tiene un consejo. Algunas personas dicen que es mejor saber para qué se emplean todas y cada una de las herramientas de su caja de herramientas en lugar de dominar únicamente unas pocas; puede ir aprendiendo lo que necesite saber sobre cada herramienta según vaya siendo necesario. Cada sistema Linux posee su propia caja de herramientas software, patrones y técnicas para llevar a cabo tareas, y por eso resulta útil conocer bien esta caja de herramientas. (Por ejemplo, las herramientas chkconfig y service de Red Hat, que se tratarán en el capítulo 4, resultan muy convenientes, pero si no las conoce jamás las usará).

En otras palabras, merece la pena conocer su distribución, saber de qué herramientas dispone y qué puede hacer con ellas. Esta es realmente la filosofía que rige este libro: conoce tu caja de herramientas. ¡Lo más importante que puede hacer es tener la valentía de explorar sus herramientas y su sistema Linux!

Lo que necesitará

Para usar este libro y beneficiarse de sus ejemplos, necesitará un ordenador con una distribución Linux instalada. Podrá hacer uso de este libro sin importar qué distribución tenga, pero tendrá que encargarse de la instalación por sí mismo. Eche un vistazo a los capítulos de la segunda parte para encontrar información sobre varias distribuciones comunes y populares.

En cuanto al ordenador en sí (el hardware), casi cualquier configuración valdrá; una de las bellezas de Linux es que puede ejecutarse sobre configuraciones hardware relativamente modestas. Con una única excepción, cada distribución, paquete de software y caso de estudio tratado en este libro fue instalado y ejecutado en un sistema con un procesador de gama media, 128 MB de memoria RAM y un disco duro de 30 GB. Este sistema al completo (incluido un monitor de alta calidad) se compró por menos de mil euros. Aunque seguramente no desee utilizar esta configuración para el servidor de una empresa, en un nivel de producción, es más que suficiente para la mayoría de los usos (por ejemplo, se usó incluso en la escritura de este libro). ¿Y cuál es la excepción? Bien, la excepción es el caso de estudio del cortafuego del capítulo 16, que se ejecutó sobre un hardware aún más modesto.

Los sistemas Linux pueden liberarle de su PC y permitirle centrarse realmente en lograr que su ordenador solucione sus problemas. Como todas las libertades, esta es a veces sencilla y a veces un reto, pero siempre es apasionante. Este libro le ayudará a lo largo de ese camino. Enhorabuena por zambullirse en el mundo Linux; ¡buena suerte, buena lectura y feliz programación! Intente no divertirse demasiado a lo largo del camino...

Parte I

Antedecentes

Linux es un sistema operativo, pero también representa algo más. Desde muchos puntos de vista, Linux es el paladín de un conjunto de ideales. El que se sienta identificado con estos ideales es un tema aparte de que haga uso de la tecnología, por supuesto, pero no obstante una completa comprensión de Linux y de los sistemas basados en Linux requiere como mínimo tener conciencia de su legado. Por ello, en este capítulo aminoraremos el paso y caminaremos tranquilos describiendo por encima la historia que condujo finalmente a Linux.

Cuando se pasa suficiente tiempo dedicado a los aspectos técnicos, se suele tender a olvidar el origen de todo. Se emplea tanto tiempo en el uso del conocimiento que se pierde parte de la maravilla sentida en el momento de su adquisición. De todos modos, si se encuentra leyendo este libro, probablemente esté bastante entusiasmado, o al menos sienta curiosidad sobre Linux. Aún posee esa sensación de aventura y desea saber todo lo posible sobre todo este "asunto Linux". Al fin y al cabo, este libro erraría si no deja en el lector la sensación de poder y frescura que los sistemas Linux proporcionan.

Una gran parte de esa frescura se deriva del pedigrí prestigioso y colorista de Linux; cualquier auténtico gurú de Linux comprende la historia y la cultura de Linux de la misma manera en que implícitamente comprende los niveles de inicio, las bibliotecas C y los gestores de paquetes. Por eso, este capítulo cubrirá parte de la historia y la cultura de Linux. Si ya le resulta familiar la historia sobre los antecedentes de Linux (o si simplemente no le interesa) siéntase libre de hojear o saltarse el resto de la primera parte.

Los orígenes de Linux

Si retrocede lo suficiente, descubrirá que la historia de Linux está fuertemente relacionada con la historia de Unix e incluso con la historia de la informática. La historia de Unix es una historia interesante, ya que a pesar de haber tenido un impacto sutil,ha modificado de manera decisiva los esquemas mentales sobre la informática en general. Todos los caminos llevan a Unix, podría decirse. Alrededor de 1971, el grupo de laboratorios Bell de AT&T desarrollaron la primera encarnación de Unix (que en origen era conocida como Unics o *Uniplexed Information and Computing System*, sistema informático y de información uniplexado). El nombre de Unix era una broma sobre Multics, un precursor mucho más complejo de Unix. Durante un tiempo, Unix tuvo muy poca proyección pública. Sobre todo se utilizó dentro de AT&T para propósitos internos. Finalmente se lanzó como un producto para todo el mundo y comenzó a coger impulso. El sistema Unix de AT&T alcanzó su cima en la versión denominada *System V* (sistema cinco, a veces abreviada como SysV). Quizás inevitablemente, sin embargo, comenzaron a aparecer ramificaciones y productos derivados.

En 1978, la University of California en Berkeley presentó un sistema operativo conocido como *Berkeley Systems Distribution* (distribución de sistemas de Berkeley), o BSD, que era muy similar a Unix. Los motivos para esta divergencia eran varios. Había problemas con la licencia que AT&T requería para Unix sobre el código fuente; algunas personas argumentaban que AT&T no era lo suficientemente previsora en la mejora de aspectos esenciales de SysV. Pero, sean cuales sean las razones, BSD se volvió tremendamente popular y se convirtió en la base para muchas otras ramificaciones

Cuando el polvo se asentó (si se puede decir de verdad que se asentó, incluso hoy en día), la mayoría de los "sabores" de las ramificaciones Unix ofrecidas por diversos vendedores eran derivados de SysV Unix, BSD o un híbrido de ambas. Además, se creó la especificación conocida como POSIX por parte del IEEE (Institute of Electrical and Electronics Engineers, Inc.) con la intención de codificar las características comunes de varias de estas ramificaciones en un único estándar.

En la actualidad, existen muchos sabores de Unix en lucha unos con otros. Cada fabricante importante de hardware tiene generalmente su propio sabor, y también existen varios sabores libres. Entre los fabricantes comerciales, IBM tiene AIX, Hewlett-Packard tiene HP-UX, Compaq tiene Tru64 Unix, SGI tiene Irix, Sun Microsystems tiene Solaris, y esta lista ni siquiera es exhaustiva. Incluso Microsoft tiene su sabor, denominado Xenix, y existen herramientas para proporcionar un entorno similar a Unix sobre los sistemas operativos Windows de Microsoft.

Las versiones libres de Unix incluyen diversas ramificaciones de BSD (FreeBSD, NetBSD y OpenBSD) y, por supuesto, Linux. Linux y los sistemas libres de BSD son similares en cuanto a origen e intención, pero presentan una gran variación en cuanto al enfoque y la ideología asociada. Los sistemas Linux parecen tener un impulso más grande en la industria, pero a pesar de ello son bastante similares (al

menos hasta que se echa un vistazo bajo la cobertura). Sólo para añadir más confusión, existe un primo comercial de los sistemas operativos libres BSD denominado BSDi. Se pueden hacer algunas observaciones interesantes sobre estas ramificaciones. Quizás lo más notable es que los sabores de los fabricantes comerciales generalmente se ejecutan exclusivamente sobre sus propias arquitecturas hardware. (Incluso Sun, que dispone de una versión Intel de su Solaris, no ha conseguido mucho éxito con ella, según muchos informes.) En contraste, los sabores libres han sido adaptados a plataformas muy distintas. Linux, por ejemplo, se ha adaptado a las arquitecturas Intel, Alpha, PowerPC, SPARC, MIPS, ARM, Motorola m68k, IBM S/390, e incluso a plataformas más exóticas. NetBSD, por su parte, ¡se puede ejecutar sobre más arquitecturas incluso que Linux!

Muchos expertos han descrito este granado conjunto de sabores como la fragmentación de Unix. Es complicado decidir cuál de estos "fragmentos" es el mejor (qué sistema operativo es mejor para una determinada aplicación es algo bastante subjetivo y variable). Algunas plataformas son mejores para las aplicaciones de bases de datos, otras tienen mayor rendimiento como servidores web, mientras que otras vencen en el proceso numérico necesario en las investigaciones científicas.

De todos modos, los sistemas operativos libres están comenzando a revolver las cosas. En principio descartados como irrelevantes, o citados por expertos simplemente como ejemplos lejanos de la fragmentación de Unix, estos sistemas operativos (y Linux en particular) han mostrado trazas de poder revertir la fragmentación de Unix. Esto quiere decir que casi todos los vendedores importantes de sabores Unix han comenzado a aceptar Linux en mayor o menor medida. En el momento de la escritura de este libro, el único que se resiste parece ser Sun Microsystems, lo que quizás no sea demasiado sorprendente, ya que hay quien dice que Solaris es el sabor comercial de Unix más exitoso disponible.

Por supuesto, esto no significa que que Linux aunará inevitablemente el paisaje fragmentado de los sabores Unix, pero, de todos modos, el ritmo al que los fabricantes tradicionales de Linux están adoptando las estrategias de Linux es sobrecogedor. Los lectores curiosos pueden preguntar justificadamente por qué. Bien, la primera razón, por supuesto, es la misma razón de que exista este libro: porque Linux representa un software de calidad, flexible, potente y libre, y el campo de los negocios está tan interesado en estas cosas como los lectores de este libro. Aún así, también es posible que los fabricantes de Unix hayan comenzado a darse cuenta de que el desarrollo y mantenimiento de sistemas operativos similares a Unix es un propósito muy costoso y que no suele ser muy rentable. (La mayoría de los fabricantes de hardware obtienen sus beneficios de los sistemas y servicios que venden, no de los sistemas operativos que venden junto con ellos.) En lugar de competir entre ellos en una especie de carrera armamentística informática, muchos negocios escogerán reducir sus costes de desarrollo mediante la adopción de Linux y centrarán su competición en otras áreas: en el hardware. A pesar de ello, en el fondo es difícil generalizar sobre las estrategias de mercado de grandes corporaciones como acabamos de hacer, ya que se pueden encontrar tantas opiniones sobre este tema como expertos hay en el mundo.

Nota: La marca Unix

El término "Unix" es actualmente una marca comercial. Sólo la versión original de AT&T de Unix puede llamarse propiamente "Unix"; las ramificaciones (incluyendo BSD) son "similares a Unix". El propietario actual de la marca "Unix" es Open Group (www.opengroup.com). Entre otras cosas, intentan promover y certificar la conformidad de sistemas Unix estándar. De todos modos, el descendiente más directo del Unix creado en AT&T es probablemente el producto UnixWare de Caldera.

La naturaleza de un sistema operativo

Toda esta charla sobre sabores y la sopa de letras de Unix probablemente le ha hecho preguntarse cuál es la diferencia entre todos ellos. Si ha escuchado algo acerca de Linux en los medios de comunicación, seguramente haya oído describirlo de un gran número de modos (a veces imprecisos): "sistema operativo alternativo", "sistema operativo de reciente aparición" o "competidor de los sistemas operativos de Microsoft". Estas descripciones son verdaderas hasta cierto punto, pero como muchas otras cosas, desafortunadamente depende de cómo se defina a un sistema operativo y de cómo se defina a Linux mismo.

La definición técnica de un sistema operativo generalmente aceptada se refiere a un componente software que realiza tareas de administración de recursos y procesos para aplicaciones software. Ahora bien, esta es una definición demasiado estéril, así que vamos a descomponerla un poco en las siguientes categorías:

- Recursos
- Núcleo
- Bibliotecas de sistema

Recursos

Un recurso es todo aquello que una aplicación pueda querer usar. Memoria del sistema, unidades de disco duro, puertos serie, tarjetas gráficas, adaptadores de pantalla y tarjetas de red son todos ellos ejemplos de recursos que necesitan ser administrados (en esencia, cualquier componente hardware del ordenador). El sistema operativo debe administrar estos recursos, ¡y protegerlos! En sistemas multiusuario no deseará que usuarios hostiles sean capaces de acceder a sus recursos y a sus datos, e incluso en sistemas monousuario no querrá que un error de un programa bloquee un inocente dispositivo que sólo está esperando una serie de bytes. A esta lista de recursos se puede añadir en la actualidad la CPU (o las CPU, si dispone de un ordenador multiprocesador). Una CPU sólo puede ejecutar una ins-

trucción en cada momento, de modo que si múltiples programas necesitan ejecutarse a la vez, hay que administrarlos. Esta administración de la CPU y las tareas relacionadas de administrar qué programa se ejecutará en la CPU y en qué orden (y cómo iniciar nuevos programas) se denomina administración de procesos.

Así, en pocas palabras, un sistema operativo generalmente sólo tiene que ofrecer una infraestructura sobre la que una aplicación software de usuario puede ejecutarse. Esta definición no dice nada acerca de una interfaz gráfica de usuario (GUI), ni siquiera sobre ninguna interfaz de usuario en absoluto. De todos modos, ya que un sistema informático no es realmente útil sin estas características, muchas personas extienden la definición de sistema operativo para incluir algún tipo de interfaz de usuario. Por ello, generalmente, un sistema operativo se suele dividir en dos partes: el núcleo (a veces llamado el controlador, entre otras cosas) y las bibliotecas de sistema.

Núcleo

El núcleo es la parte del sistema operativo que se encarga de las partes vitales de la definición. El núcleo administra la entrada/salida, la memoria del sistema, las unidades de disco y cualquier otro hardware añadido al sistema, y también realiza la administración de procesos (por ejemplo, ejecutar programas y gestionarlos concurrentemente). Se puede pensar en el núcleo como en la infraestructura esencial de la que depende todo lo demás y sobre la que se ejecuta. Como el núcleo (por definición) no incluye la interfaz de usuario, el usuario nunca ve o interactúa directamente con él; en un sentido muy real, los núcleos no se pueden distinguir desde el punto de vista del usuario final, excepto en la medida en que un núcleo puede parecer presentar un rendimiento mejor que otro. (Los núcleos, por supuesto, son muy distintos desde la perspectiva de los desarrolladores de software)

Bibliotecas de sistema

Las bibliotecas de sistema son lo que realmente distingue a los sistemas operativos entre sí. Son las herramientas y programas que realmente implementan los servicios básicos del sistema, tales como una GUI o una interfaz de texto. Estas son las bibliotecas que convierten al sistema en algo útil. De todos modos, estas bibliotecas no proporcionan ningún programa real; por ejemplo, un cliente de correo electrónico o un navegador web no serán parte de las bibliotecas del sistema, sino que en su lugar se considerarán aplicaciones del espacio de usuario.

Nota: ¿Confuso?

Si se siente un poco (¡o muy!) confuso debido a este batiburrillo de palabras, no se encuentra solo. Este es sólo el modo que emplea un autor para describir

estos conceptos. En la práctica, distintos sistemas operativos exceden estos límites por diferentes partes. El punto en que acaba el espacio del núcleo y comienza el del usuario varía según el producto y, en ocasiones, según la versión del producto. En otras palabras, la definición de un sistema operativo puede variar según el fabricante del sistema operativo. Diferentes sistemas operativos hacen gala de diferentes arquitecturas y, por lo general, no necesitará preocuparse acerca de la arquitectura de ningún sistema operativo más que de la de aquel que vaya a utilizar. Por ello, la siguiente pregunta es, "¿Qué pasa con Linux?".

¿Es Linux un sistema operativo?

Si hablamos estrictamente, Linux es "simplemente" un núcleo. Esto quiere decir que Linux es un componente software que administra recursos y procesos, pero que no ofrece bibliotecas de sistema o interfaces de usuario. Como Linux sólo es un núcleo, necesita un conjunto de bibliotecas de sistema para convertirse en un entorno útil.

Si Linux sólo es un núcleo, ¿de dónde proceden estas bibliotecas de sistema? ¿Quién las escribe? ¿Quién las mantiene? Obviamente, quién mantenga estas bibliotecas realiza una contribución inmensa a los sistemas Linux en general. En la amplia mayoría de instalaciones Linux, las bibliotecas de sistema son utilidades GNU, escritas y mantenidas por la Fundación para el Software Libre (*Free Software Foundation*).

Linux y GNU

La Fundación para el Software Libre (FSF) es una organización fundada por Richard Stallman del Media Lab del MIT. La FSF ha tenido un enorme impacto sobre Internet y el mundo de la informática, pero quizás sus dos mayores contribuciones sean GNU y la Licencia Pública General (*General Public Licence*, GPL) de GNU. (GNU son las siglas de "*GNU's Not Unix!*", "GNU No es Unix". Cuidado con este acrónimo recurrente, puede darle dolor de cabeza.) En sus propias palabras, extraídas de su sitio web:

"La Fundación para el Software Libre (FSF) se ocupa de eliminar las restricciones sobre la copia, redistribución, comprensión y modificación de programas informáticos. Hacemos esto promoviendo el desarrollo y la utilización del software libre en todas las áreas de la informática, pero muy particularmente, ayudando a desarrollar el sistema operativo GNU."

El proyecto GNU se fundó en 1983 y su intención es ser nada menos que un sistema operativo hecho y derecho, incluyendo un núcleo, bibliotecas de sistema y

un exhaustivo conjunto de aplicaciones de usuario útiles. De todos modos, a finales de los años 80 y comienzos de los 90, a GNU le faltaba un núcleo. El núcleo es como la piedra angular del arco de entrada a un edificio, y sin él, el ideal de GNU como un sistema operativo maduro estaba incompleto.

En 1991, Linus Torvalds (en aquel entonces un estudiante universitario) comenzó a escribir el núcleo de un sistema operativo similar a Unix porque no se encontraba satisfecho con las opciones gratuitas existentes y, como tantos de nosotros en la época de estudios, andaba demasiado apurado de dinero como para comprar un sistema comercial. El núcleo de Linus (finalmente llamado Linux) comenzó su evolución rápidamente. Cuando Linus decidió publicar su núcleo bajo la licencia GPL de GNU, ocupó el hueco existente en el proyecto GNU; mientras que GNU ofrecía las bibliotecas de sistema y las aplicaciones de usuario, no era un sistema operativo completo sin un núcleo. Con el "detalle final" de Linux, GNU se convirtió en una realidad. Por supuesto, en años recientes el núcleo propio de los proyectos GNU (denominado HURD) a madurado en gran medida, y se encuentra cerca del punto en que puede utilizarse para reemplazar a Linux en sistemas actuales.

Cuando sucedió esto, causó bastante agitación dentro de la comunidad académica. Quizás desafortunadamente, Linus y su núcleo Linux cosecharon una cuota desmedida de elogios a expensas del software GNU, que convirtió al sistema completo en una realidad, y la gente comenzó a hablar de "sistemas Linux". Ahora, una década después, hemos llegado al punto en que la gente habla del "fenómeno Linux" y del "sistema operativo Linux".

Nota: Otras posibilidades

Si Linux es simplemente un núcleo y GNU sólo es un conjunto de bibliotecas, entonces deberían de ser intercambiables, ¿no es cierto? Quizás se pudiera reemplazar Linux por un núcleo diferente y entonces tendríamos un sistema GNU basado en alguna otra cosa. O, quizás se podría usar el núcleo Linux con un conjunto distinto de bibliotecas de sistema, y habríamos conseguido otro entorno que funcionaría sobre el núcleo Linux. Esta sección describirá alguna de las cosas más sorprendentes que puede hacer con Linux.

De hecho, intercambiar tanto el núcleo como las bibliotecas de sistema es posible en teoría y, en ocasiones, también en la práctica. Por ejemplo, podría ser posible reemplazar las bibliotecas GNU por las bibliotecas de sistema desarrolladas para, digamos, FreeBSD. En ese caso, conseguiría un sistema BSD/Linux. O, podría compilar las bibliotecas GNU para el núcleo de Solaris, y tendría un sistema GNU/Solaris. Mientras tanto, el proyecto GNU continúa trabajando sobre su núcleo HURD, y ya dispone de una distribución funcional basada en Debian (lea el capítulo 6) que funciona sobre HURD en lugar de sobre Linux.

Estas posibilidades pueden volverse aún más exóticas si se sale fuera del reino de las bibliotecas de sistema similares a Unix. Por ejemplo, recientemente

Apple lanzó la versión OS X de su sistema operativo para Macintosh. El corazón de este proyecto se conoce con el nombre de Darwin y está basado en BSD. Existe un grupo que trabaja en la adaptación de Darwin al núcleo Linux. Mientras tanto, Transvirtual Techonologies, Inc. produce una máquina virtual Java de código abierto conocida como Kaffe. Además, realizan un trabajo relacionado denominado PocketLinux para producir un entorno operativo Java esencialmente mediante el empleo de Kaffe como la biblioteca de sistema que se ejecuta sobre un núcleo Linux, para dispositivos embarcados como los PDA.
Las posibilidades son infinitas y de hecho hay una multitud de esfuerzos encaminados a la fusión de varios productos y la ejecución de Linux de muchas maneras especiales, como diversos tipos de entornos y bibliotecas de sistema. Algunos de estos son más teóricos que otros (por ejemplo, con toda probabilidad Linux/BSD resultaría extremadamente difícil en la práctica), pero este es sin lugar a dudas uno de los aspectos más sugerentes de Linux y el software libre.

GNU y Linux: el sistema operativo de los campeones

Volviendo a un terreno más concreto, puede que aún no esté seguro acerca de qué trata toda esta exaltación de Linux. Seguramente crea que los sistemas Linux resultan bastante útiles y que se están volviendo muy populares, de otro modo no estaría leyendo este libro ahora mismo. Pero puede que no entienda realmente por qué es cierto todo esto. ¿Qué sucede con los sistemas Linux que los hace mucho mejores que otros sabores de Unix? Bien, todo este libro es la respuesta a esa pregunta; una vez que lo haya leído, comprenderá todas las cosas apasionantes y útiles que Linux le permite hacer. Esta sección supone un primer intento de contestar a esta pregunta y describirá a grandes rasgos algunos de los atributos más positivos de Linux.

En su infancia, se enseñó a Linux la virtud de lo práctico. Ahora, a medida que Linux alcanza su madurez, aquellos valores son los que hacen a los sistemas Linux tan notoriamente versátiles y eficaces. Dejando las metáforas a un lado, los sistemas Linux son realmente los sistemas operativos más prácticos que el dinero puede comprar (¡y las mejores cosas de la vida son gratis!). Los sistemas Linux le permiten realizar todo lo que puede hacer con cualquier otro sistema similar a Unix, y muy a menudo ofrecen más características para el mismo proceso.

La frase más popular para describir cómo se inician los proyectos de código abierto es que los desarrolladores "rascan donde pica". Esto quiere decir que un desarrollador software descubre que necesita una herramienta y que no existe una

herramienta que supla su necesidad, las herramientas que existen son inadecuadas o puede que sencillamente las herramientas sean demasiado caras. Puede existir un numero cualquiera de razones por las que un desarrollador quiera escribir una pieza de software en particular, pero en la mayoría de los casos un desarrollador comienza o se une a un proyecto porque necesita el software que está escribiendo o la característica que va a añadir.

Por eso, casi cada pieza de software que se suele encontrar instalada en los sistemas Linux ha sido escrita desde la nada para cubrir una necesidad en particular. Las características y la calidad de este software reflejan estos orígenes; en general, las herramientas que se encuentran en un sistema Linux suelen ser muy robustas y no carecen de ninguna característica, especialmente cuando se comparan con las herramientas equivalentes para los sistemas comerciales.

El programa ls

Como ejemplo muy simple de este concepto de "rascar donde pica" considere el omnipresente programa ls. Cada sabor comercial de Unix tiene una versión del programa que lista directorios ls; es una parte fundamental de la interfaz de usuario.

Aún así, ls es un programa que se escribe de un modo extremadamente simple, de modo que una vez que los fabricantes de Unix consiguen una versión que funciona, la mayoría no se molesta en mejorarlo. Después de todo, el desarrollo de software es una actividad costosa, y los fabricantes de Unix pueden obtener un beneficio mucho mayor de su inversión si se centran en añadir características por las que pueden cobrar un extra en lugar de mejorar detalles menores de programas pequeños.

Aún así, la mayoría de los sistemas Linux utilizan el programa ls en color de GNU como su versión de ls. La versión de ls en color es como la versión típica pero con la capacidad de mostrar listados de directorios codificados por color. Es sólo algo atrayente; unos colores bonitos no añaden una gran cantidad de valor al conjunto de un sistema operativo. De todos modos, en su nicho de utilidad, resulta una característica muy útil; después de usarla durante un tiempo, probablemente se haga tan dependiente que cuando tenga que usar otra versión comercial de Unix que no lo tenga instalado note su falta.

gzip, bzip2, tar y vim

Por supuesto, el programa ls no es el único ejemplo de un programa Linux que "rasca donde pica". De hecho, los ejemplos son literalmente demasiado numerosos para presentar una lista. Las herramientas GNU de compresión de archivos gzip y bzip2 presentan mejores tasas de compresión que muchas herramientas comerciales equivalentes. La herramienta tar de GNU soporta de modo interno el proceso de los archivos generados con gzip y compress de Unix, de modo que los usuarios no

tienen que utilizar zcat. La versión GNU de la utilidad make tiene una sintaxis extendida que facilita el desarrollo de software. Muchos sistemas Linux vienen equipados con vim (*VI iMproved*, VI Mejorado), que es compatible con los programas estándar vi pero que ofrece numerosas extensiones muy útiles. GNU proporciona un conjunto de herramientas conocido como autoconf que simplifica notablemente la adaptación y la instalación de software en sistemas Unix dispares. Los entornos de escritorio GNOME y KDE son varios grados de magnitud (si se pudiera medir en números) más sofisticados y usables que los escritorios utilizados en muchos sistemas Unix.

El secreto del éxito

¿Comienza a entrever el patrón? Se suele escuchar que Linux es un "sabor de Unix" e inmediatamente se le echa al motón del resto de los sabores. Los expertos a veces lo desestiman como un capricho, ya que, como todo el mundo sabe, "Unix" no es amigable para el usuario. Aunque en realidad, los sistemas Linux son, casi literalmente punto por punto, tan robustos como otros sistemas Unix y poseen más características.

A lo que se reduce todo esto es que la suma de estas pequeñas características hacen de Linux uno de los mejores entornos de usuario similar a Unix. Muchas empresas utilizan Linux como su plataforma básica de desarrollo cuando escriben software para Unix, ya que las herramientas son así de buenas; el resto de las versiones para otros sabores Unix son "adaptaciones". Muchos profesionales independientes, cuando tienen que enfrentarse al uso de un sistema distinto de Linux, comienzan por la instalación de las herramientas GNU en el sistema para mejorar su entorno de trabajo. (¡Existen incluso adaptaciones de las herramientas GNU para los sistemas Windows de Microsoft!) Este es un cumplido bastante impresionante para todos aquellos desarrolladores que están "rascando donde pica".

Todo este asunto de rascar donde pica no es mano de santo, por supuesto, y un buen entorno de usuario no suministra automáticamente un mejor rendimiento y escalabilidad.

La falta de guía que sí que tienen las corporaciones (¡por no mencionar la falta de financiación!) implica que Linux ha estado históricamente rezagado con respecto a los sabores comerciales de Unix en términos de soporte a sistemas empresariales de gama alta.

Y puesto que muy pocos desarrolladores independientes disponen de sistemas de 32 procesadores con 8 GB de memoria, muy pocos desarrolladores están "rascando" para lograr que Linux se ejecute sobre ese tipo de sistemas. A pesar de todo, eso está cambiando porque las empresas relacionadas con Unix más perspicaces se están dando cuenta de que sus empleados utilizan sistemas Linux en casa y, en ocasiones, en el trabajo.

Una administración inteligente implica apoyarse sobre lo que es bueno, incluso si es mejor que tu propio producto, y los negocios también tienen pequeñas necesidades, cosas que "pican".

El zen de la plataforma

Los fabricantes de Linux producen diversos sistemas operativos basados en el núcleo Linux y otras piezas de software variadas, tales como las herramientas y bibliotecas GNU. Aún así, con frecuencia oirá a la gente de la industria de las tecnologías de la información referirse a "plataformas" de distintas formas. Puede que incluso a veces oiga a alguien referirse a las diferentes distribuciones Linux como plataformas. Antes de sumergirnos en el resto del libro, merece la pena tomarnos un momento para hablar sobre las plataformas.

El término plataforma es una de esas palabras que todo el mundo emplea, cargadas de significados, pero, en el fondo, es sólo una palabra que la gente utiliza para referirse a un sistema en su conjunto que le permite realizar su trabajo. Así, el término plataforma engloba tanto al hardware como al software. El sistema operativo Windows de Microsoft ejecutado sobre un procesador Intel es una plataforma (que a veces se denomina "Wintel"), y un sistema Linux que se ejecute sobre un procesador Intel es otra plataforma diferente, aunque podría tratarse del mismo hardware.

Algo digno de mención es que una plataforma es tanto el sistema operativo como el hardware sobre el que se ejecuta ese sistema operativo. En muchas plataformas (y en la mayoría de las plataformas Unix), el fabricante del hardware también produce el software. Por ejemplo, la plataforma Solaris típica consiste en Solaris de Sun Microsystems que se ejecuta sobre un procesador SPARC también la produce Sun.

A pesar de esto, Linux supone la excepción a esta regla. Recuerde que el núcleo Linux ha sido adaptado a un número sorprendente de arquitecturas hardware. La primera versión fue para la arquitectura Intel y es ahí donde Linux es más fuerte. Pero también se ejecuta muy bien sobre los procesadores Alpha de Compaq, PowerPC de IBM/Motorola y SPARC de Sun, así como en muchos otros.

¿Y qué?, se preguntará. Bien, ya puede haberse dado cuenta de que uno de nuestros temas recurrentes es que su ordenador debería hacer lo que se necesite; si no lo hace, entonces es sólo una pérdida de tiempo y recursos. Si sólo necesita que funcione, ¿realmente le importa cómo funciona? ¿Importa que tenga el procesador más potente, o la mejor tarjeta de video, o el disco duro más grande si todo lo que quiere hacer es comprobar su correo electrónico y navegar por la web mientras en segundo plano se ejecuta un servidor web? ¿Necesita actualizarlo con el hardware más reciente cada vez que éste cambie?

Bueno, puede que la respuesta sea afirmativa; sólo uno puede decidir por sí mismo. Aún así, para muchos (puede que para la mayoría) la respuesta es un rotundo "¡No!" Para esta gente no es importante de qué se componga el ordenador (podría ser Windows funcionando sobre un chip Intel, o bien Linux sobre un SPARC). Mientras que el cliente de correo electrónico y el servidor web se ejecuten cuando se requiera, casi nadie percibirá ninguna diferencia.

De nuevo preguntará, ¿y qué? ¿Por qué es relevante? Es relevante porque ilustra una de las cualidades más importantes (casi con el carácter de una paradoja zen) de

los sistemas Linux: que mientras que el sistema funcione, no importa cómo lo haga. Una vez que ha dominado un sistema operativo, no importa qué es la "plataforma". El equipo de trabajo principal del autor es Red Hat Linux que se ejecuta sobre un procesador Alpha 21164PCA a 533 MHz de Compaq. Con dos únicas excepciones, cada tema tratado en este libro funciona sobre dicho sistema, así como en el sistema de pruebas basado en Intel utilizado para desarrollar el contenido. Esto es así porque un sistema Linux es un sistema Linux sin importar el hardware sobre el que esté funcionando.

Una vez que haya dominado el material tratado en este libro, será autosuficiente en el manejo de los sistemas Linux. Aún más, será autosuficiente a la hora de enfrentarse con cualquier plataforma Linux, y ya nunca más volverá a suponer una diferencia. Resulta bastante refrescante poder salirse de la carrera de ratas que es el mundo dominante del PC y ser capaz de lograr que su sistema funcione, del modo en que desee, con un mínimo de preocupaciones.

El debate GNU/Linux o Linux

En términos estrictos de número total de líneas de código, el núcleo Linux es sólo una fracción del código que configura un sistema bien preparado; el código que proviene del proyecto GNU es una parte de crucial importancial para los sistemas operativos basados en Linux que se usan cada día. Por eso, sería mucho más preciso describir dichos sistemas como sistemas GNU, o sistemas GNU/Linux como compromiso. Estos términos reflejan la importante contribución que el proyecto GNU ha supuesto para el éxito de Linux.

Por desgracia, el daño ya está causado. Es difícil invertir una década de impulso y el nombre común de estos sistemas se ha convertido en, simplemente, Linux. Una vez que los medios de comunicación empiezan a utilizar un término, aunque sea inapropiado, entra en el discurso público y el insistir en el uso de un nombre más preciso se convierte en una rareza. Este fenómeno pone a libros como este ante una cuestión espinosa. El objetivo de un libro es ser preciso y detallado, aunque debe de ser preciso tanto en significado como en uso. ¿Es más preciso referirse a un sistema basado en Linux como Linux (al igual que en el uso cotidiano) o usar el término más correcto pero menos reconocido de GNU/Linux?

El debate GNU/Linux o Linux ha sido un debate muy encendido dentro de la comunidad, y también es el origen de un buen montón de sentimientos negativos. Y con todo el derecho del mundo: a nadie le gusta ver cómo la magnitud de sus contribuciones se diluye debido a una única y minúscula parte del todo, tanto como a nadie le agrada que sus contribuciones integrales sean ahogadas por el todo. Todo lo que podemos hacer es tratar de ser tan precisos como sea posible, dadas estas limitaciones.

Por eso, este libro se refiere a los sistemas GNU/Linux como sistemas basados en Linux, o sistemas Linux en pro de la brevedad. Esperemos que así se alcance un equilibrio entre la precisión y la comprensibilidad del texto.

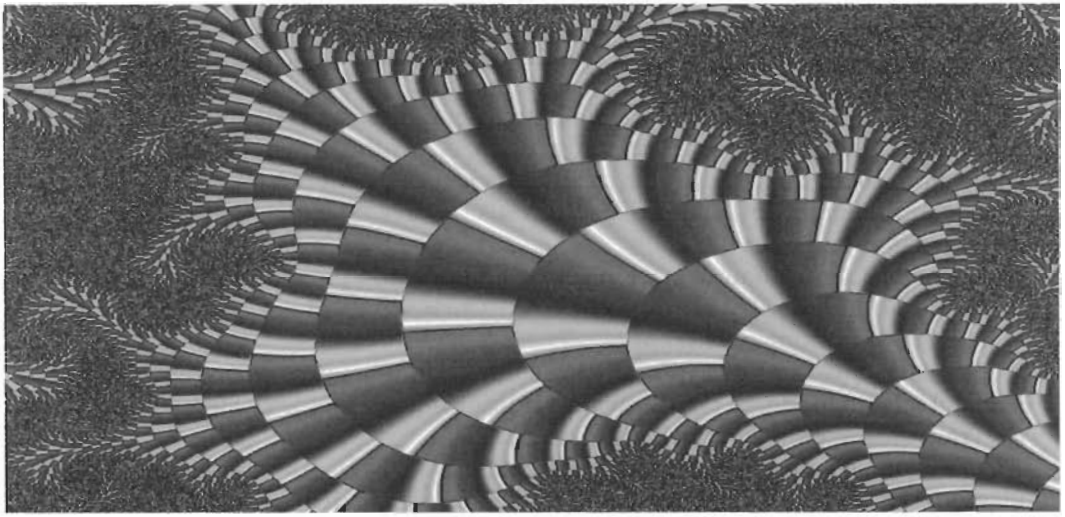
Referencias

Si está interesado en la historia de Linux, comience con el documento FAQ sobre Unix que puede encontrar en los grupos de noticias comp.unix.questions y comp.unix.shell. El capítulo seis de este documento proporciona un resumen bastante comprensible del linaje de los diversos sabores de Unix al mismo tiempo que referencias adicionales.

La mejor fuente de información sobre la Fundación para el Software Libre (Free Software Foundation) y el proyecto GNU (incluido el núcleo HURD) es su sitio web, en la dirección www.fsf.org.

Se puede encontrar más información sobre Darwin y Linux en www.darwinlinux.org; del mismo modo, se puede encontrar información sobre el sistema PocketLinux de Transvirtual en www.pocketlinux.com.

También se puede encontrar una entretenida visión sobre las motivaciones y objetivos de Linux en una discusión electrónica dentro de los foros de Usenet entre Andy Tanenbaum (el creador de Minix) y Linus Torvalds. Esta discusión se puede encontrar en diversos sitios web; simplemente busque información sobre "Linus vs. Tanenbaum" o "Linux is Obsolete" (Linux está obsoleto) en su buscador web favorito. Esta línea de discusión, vista hoy en día, es una interesante mirada retrospectiva al comienzo de toda la "revolución Linux".



2
Introducción al código abierto

Los sistemas Linux suelen ser software de código abierto, que quiere decir que su código fuente se puede conseguir fácilmente y que son redistribuibles de manera gratuita. De todos modos, mucha gente no comprende qué quiere decir realmente que un programa sea de código abierto; es más, mucha gente no entiende lo fundamental que resultan los valores del código abierto para la evolución de Linux. Este capítulo tratará acerca del software de código abierto y el software libre y explorará lo que estos conceptos significan para GNU y Linux. Los lectores que ya estén bien versados en el software de código abierto y libre pueden simplemente hojear este capítulo o saltárselo y avanzar hasta el capítulo 3.

La naturaleza del software

Existe una gran cantidad de pasión y retórica alrededor del software de código abierto y del software libre. Hay varios argumentos básicos que aparecen de diversas formas, que establecen las bases del software de código abierto y del software libre. La cuestión filosófica del software libre sigue siendo principalmente subjetiva y la razón que una persona considera válida puede no funcionar para otra persona distinta. En otras palabras, es un asunto delicado. Esta sección intentará delinear los principios básicos de los argumentos siguientes tan objetivamente como sea posible. De todos modos, los lectores deberían tomar esta lección con prevención y recordar que se trata solamente de la modesta perspectiva del autor.

El desarrollo de software no es algo sencillo. Requiere conocimiento y habilidad y, por encima de todo, tiempo. Por estas razones, el desarrollo de software lleva asociado un coste. Para las organizaciones comerciales, es costoso financieramente; incluso para los desarrolladores de software libre, es costoso en cuanto a esfuerzo se refiere. Por eso tiene mucho sentido compartir tanto como sea posible. A este respecto, el desarrollo de software es como cualquier otra disciplina científica o técnica. Los científicos comparten la información sobre su trabajo entre sí y trabajan juntos como comunidad para hacer progresar el estado del arte, construyendo sobre el trabajo de los demás.

Por el contrario, el objetivo de las empresas son los beneficios y se esfuerzan por conseguir tantos como puedan. Por supuesto, las empresas que venden productos deben llevar a cabo las labores de investigación y desarrollo que llevan a la creación de tales productos. Aún así, las empresas no tienen ningún motivo para compartir esta información con nadie más; después de todo, si lo hicieran, otra organización podría beneficiarse restándole el beneficio que podría haber obtenido para sí misma. Normalmente, este modelo bipolar funciona bien: los científicos en centros de investigación (y a veces en la industria) producen teorías y descubrimientos, realizan experimentos y comparten sus descubrimientos. Al mismo tiempo, los ingenieros toman esta información y la utilizan para construir productos reales que las empresas venden. Las empresas existen para organizar y administrar los recursos necesarios para llevar esos productos al mercado, una tarea que suele estar más allá de la capacidad de una sola persona.

Como muchos otros, este modelo falla cuando se aplica al software. La dificultad primaria del software es que en sí mismo no tiene un valor intrínseco. Básicamente, una pieza tangible de un equipo como un coche tiene un valor porque cuesta algo construirla. Aún disponiendo de los esquemas completos del coche hay que tener las materias primas necesarias para construirlo, y esos materiales no son gratuitos. Por eso el coche tiene un valor intrínseco, aparte del coste empleado en su diseño.

Un programa software, por su parte, no tiene ningún coste asociado a su construcción. Una vez que se ha escrito, se puede replicar de manera infinita, sin esfuerzo y con exactitud (siempre sin errores). Un coche tiene costes por los materiales utilizados y el proceso de fabricación. Tanto coches como programas software cuestan un dinero durante su fase de desarrollo, por supuesto; pero una vez que la investigación y desarrollo se han completado, los coches reales aún tienen que ser manufacturados, mientras que el software simplemente se copia. Ya que el software no tiene coste de duplicación, su único coste es el esfuerzo empleado en desarrollarlo. Por eso, al contrario que un coche, el software no tiene un valor intrínseco. De modo que, si se para a pensarlo, el concepto de una "compañía de software" que realmente vende software puede resultar increíblemente extraña. Es como vender aire.

Ahora bien, sigue siendo cierto que los desarrolladores de software tiene que recuperar sus costes de investigación y desarrollo. Sin embargo, esta clase de investigación y desarrollo supone sólo un coste indirecto del producto principal. Para los

fabricantes de hardware que incluye software (como los servidores web, los teléfonos móviles, o los asistentes personales digitales, los PDA) el software es sencillamente un coste necesario para el desarrollo del producto. Para las empresas orientadas a servicios, como los desarrolladores de sitios web y las empresas de soporte de redes, el software sólo es un coste indirecto de ofrecer el servicio. En ningún caso el software se vende por su propio valor. Cuando se piensa en las cosas desde este punto de vista, la idea de la venta de software parece más y más rara cuanto más se piensa en ella.

Nota

Por supuesto, todo este razonamiento es una generalización y no siempre es verdadero. En concreto, no se puede aplicar al **software horizontal**. El concepto de un producto de software horizontal se describe dentro de su propia sección, un poco más adelante en este mismo capítulo.

El resultado de todo esto es que puesto que el único coste de muchas clases de software es el coste empleado en su desarrollo, no hay un motivo racional para que exista una empresa con el propósito de organizar y administrar los recursos necesarios para fabricarlo. Esta es la razón (o al menos ideas similares) sobre la que se basan los activistas que creen en una filosofía del desarrollo de software muy distinta. Richard Stallman es un activista y un científico informático que fundó la Fundación para el Software Libre (FSF) y el proyecto GNU con la intención de fomentar una cultura de intercambio para el desarrollo de software. La FSF fomenta el intercambio y la colaboración (el sistema de valores científico) dentro del desarrollo software. Todo el software de la FSF se hace público junto con su código fuente y una licencia que asegura su transparencia continuada.

Stallman describe el software desarrollado de este modo como "software libre". En su lenguaje original, el inglés, es *free software*. La expresión se ha convertido en un viejo dicho de la comunidad de código abierto, el "*free*" de "*free software*" es "libre" como en "*free speech*" (libre discurso), no "gratis" como en "*free beer*" (cerveza gratis). Esto supone una diferencia muy importante. El software que se regala sin código fuente (como algunos navegadores web) no es software libre, sino simplemente software "sin coste".

Las razones mostradas en este capítulo se basan ampliamente en ideas vagamente económicas, filosóficas o políticas. Aunque el libro se esfuerza por alcanzar una cierta objetividad, los lectores deberían darse cuenta de que existen muchos puntos de vista distintos sobre estos temas, y desde aquí les sugerimos que lean las razones y la documentación del propio Stallman (y de la FSF) a este respecto. Una de las mejores fuentes de información de este tipo es, obviamente, el sitio web de la Fundación para el Software Libre, que se encuentra en www.fsf.org. En este sitio podrá encontrar las posturas oficiales de la FSF y sus argumentos sobre estos asuntos.

Código abierto frente a código libre

Todos estos abstractos temas económicos y éticos están bien y son correctos, pero la idea del software libre es, para muchas personas, ligeramente radical, ya que parece defender ideales que son anómalos con respecto al curso normal de la industria. Algunos incluso se oponen con vehemencia a la idea; un oponente particularmente notable en un hombre llamado Bill Gates, que construyó un monopolio del software gracias a un software con demasiada marca comercial.

Para suavizar esta impresión de radicalidad, aparece aquí el concepto del código abierto, como una alternativa al término "software libre". La distinción entre el código abierto y el software libre es sobre todo una cuestión de semántica y enfoque, más que una diferencia en la ideología.

En esencia, el software libre trata de la accesibilidad; para tener éxito, el software libre debe ser accesible a tantos grupos como sea posible. De acuerdo con esto, algunas personas creen que el software libre sólo puede triunfar si los productores de software más prolíficos practican esta ideología; como la mayoría de los productores de software más prolíficos son empresas comerciales, la idea del software libre debe orientarse a ellos. No obstante, las empresas suelen mostarse reacias a involucrarse en asuntos políticos, y algunas personas creen que probablemente las connotaciones ideológicas del término "software libre" desanimen a las empresas a participar en el proceso. Además, sigue existiendo esa confusión en torno al propio término (ese significado de "libre " frente a "gratis").

Para contribuir a la resolución de estas dificultades, mucha gente comienza a utilizar el término "código abierto" (*open source*) en lugar de "software libre" (*free software*). Sin embargo, el término "código abierto", no recoge realmente el sentido de libertad que "software libre" sí recoge, y por eso algunas personas (incluido Stallman) tienen sus objeciones ante el código abierto. En este caso, el razonamiento principal es que se supone que la idea es radical, por eso suavizar la manera de expresarla implica abandonar la idea real.

Se trata de un asunto especialmente peliagudo, y no parece que vaya a resolverse a corto plazo. Stallman ha publicado un artículo excelente en el sitio web de la FSF, titulado "Por qué 'Software Libre' es mejor que 'Código Abierto'" ("*Why 'Free Software' is Better Than 'Open-Source.'*") Los lectores interesados pueden leer las propias palabras de Stallman y la FSF en el sitio web www.fsf.org.

Mi propia opinión sobre el tema es que "código abierto" es el término más realista y con menor carga emocional, y por ello este libro utilizará el término "código abierto", sencillamente para dejar de lado el debate y centrarse en la tecnología.

Software de código abierto en acción

Pongámonos serios. Aquí todos somos más bien técnicos, ¿verdad? Dejemos a un lado toda esta charla sobre los coches, el software y la libertad durante un rato y

vayamos directos a nuestro objetivo. Esta sección explicará cómo funciona el desarrollo de software de código abierto y describirá su mayor atractivo: simplemente puros números.

Todo se trata de software. Cuando se sienta frente a su ordenador, sólo quiere que haga lo que necesita que haga. Claro, querrá que su ordenador lo haga del modo más innovador y elegante posible, pero en general, lo que persigue es la funcionalidad. De eso es de todo lo que se trata realmente el software de código abierto. El software libre es nuestra conciencia, pero en la práctica, el código abierto trata únicamente acerca de escribir software de calidad.

En última instancia, el software lo escribe gente: científicos informáticos, programadores, ingenieros de software, o como quiera que se llamen a sí mismos. Cuanta más gente esté contribuyendo en una pieza de software, mejor será ese software. Si se dispone de cinco desarrolladores dedicados a tiempo completo y cada uno soluciona diez problemas al mes, aún no será tan bueno como si se dispone de cien programadores ocasionales que solucionan un problema al mes cada uno. En una serie de influyentes artículos, Eric Raymond describe los beneficios que la metodología del código abierto puede ofrecer. En "La catedral y el bazar" ("The Cathedral and the Bazaar") popularizó la frase: "Teniendo suficientes ojos, todos los problemas son sencillos." Esta frase es un resumen conciso de los mecanismos reales del desarrollo de código abierto y un buen punto de arranque para comprender cómo funciona realmente el software de código abierto.

El desarrollo de software de código abierto da resultado mediante el proceso de conseguir que tanta gente como sea posible trabaje en una determinada pieza de software. Sin embargo, como el desarrollo de código abierto es de naturaleza fundamentalmente voluntaria, un proyecto de código abierto sólo puede atraer a un gran número de voluntarios cuando es atractivo para un gran número de usuarios. Por eso, el código abierto funciona mejor en dos casos: aplicaciones de extremado carácter horizontal y colaboraciones oficiales entre organizaciones que ponen sus recursos en común.

Aplicaciones horizontales

Las razones discutidas en la sección con que se abría este capítulo, "La naturaleza del software", incluían una aclaración, en el sentido de que sólo tenían sentido si se trataba de aplicaciones de software horizontales. Una aplicación horizontal es aquella que cubre una necesidad compartida por varios tipos de problema. (Una aplicación vertical sólo cubre una necesidad muy reducida y específica, y alguno de los razonamientos dados anteriormente no repercuten sobre las aplicaciones verticales.) Estos son algunos dominios de aplicación horizontal típicos:

- Sistemas operativos (que todo el mundo necesita para hacer lo que sea en su ordenador).
- Compiladores de lenguajes de programación (que todo el mundo que quiere desarrollar software necesita).

- Servidores web (que cualquiera que quiera ofrecer contenidos web necesita).

Como mucha gente necesita la aplicación, una versión de código abierto dispone de una amplia base desde la que atraer a colaboradores.

Debido a esto, las aplicaciones de código abierto tradicionales más exitosas están generalmente dentro de dominios horizontales: Linux y el sistema GNU conforman un sistema operativo; el servidor web Apache se encarga del funcionamiento de más sitios web que ningún otro servidor web en la superficie del planeta; el Entorno de Escritorio K (*K Desktop Environment*, KDE) es una interfaz de usuario, que es algo casi tan general como un sistema operativo; el grupo de compiladores GNU (*GNU Compiler Collection*) es una herramienta básica de los desarrolladores de software, y hay muchas más aplicaciones. Cada una de estas aplicaciones es un ejemplo de software que "todo el mundo" que se corresponda con un perfil particular necesita; cuando "todo el mundo" necesita algo, una gran parte de ellos deseará contribuir en su desarrollo.

Una vez que hay un montón de desarrolladores contribuyendo en un proyecto, comienza a convertirse en algo más que el conjunto de su código. Puede que Luis no necesite una característica en particular, y podría suceder que Patricia si la necesite. Sin embargo, si Patricia sigue adelante e implementa esa característica, puede que tenga que añadir algunos cambios en la arquitectura de los que más adelante Luis podría beneficiarse. Un ejemplo concreto de esto es el sistema virtual de archivos (*virtual filesystem*, VFS) de Linux, que define una API común entre varios tipos de sistemas de archivos. Esta característica fue añadida para facilitar la escritura de controladores de dispositivos para sistemas de archivos, pero también permitió a los desarrolladores implementar avanzados sistemas de archivos "falsos", como controladores de dispositivos, dispositivos USB, terminales de entrada del usuario al sistema e información de procesos. Estas características tan útiles habrían sido mucho más difíciles o incluso imposibles de implementar si los desarrolladores no hubieran añadido en primer lugar la arquitectura VFS.

Desarrollo cooperativo de software

La otra vía principal para que el modelo de código abierto pueda salir triunfante se da cuando las organizaciones ponen en común sus recursos. En este caso, múltiples organizaciones colaboran en alguna pieza de software en particular, compartiendo tanto el código fuente como los resultados.

Mediante la colaboración, empresas potencialmente competidoras producen software que beneficia a todos los implicados. Por ejemplo, durante un gran periodo de tiempo, el núcleo Linux carecía de buenas características orientadas a la empresa, como el soporte para múltiples procesadores, grandes cantidades de memoria y sistemas de archivos con *journaling* (un método de lectura/escritura de disco mediante transacciones). Una vez que corporaciones importantes como SGI, IBM y Compaq comenzaron a colaborar con Linus Torvalds y el resto de los desarrolladores

principales del núcleo, estas características comenzaron a mejorar rápidamente y, de hecho, en algunos casos ahora son soluciones que pueden competir con otras; actualmente hay al menos tres opciones distintas para un sistema de archivos de tipo *journaling* para Linux, por ejemplo.

La razón por la que el código abierto ofrece un beneficio a las empresas es que al final todos los participantes están en las mismas condiciones. La licencia del código abierto impide que ninguna entidad tome un puesto dominante en el proyecto; cada uno de los participantes tienen el mismo derecho sobre el código fuente (junto con cualquier otro desarrollador independiente o nueva organización que quiera participar o utilizar el software). A una empresa puede darle un pronto y volverse a casa, pero no puede llevarse su pelota consigo como un niño enfurruñado.

Por todo esto, el software de código abierto está haciendo avances en empresas donde el software que se desarrolla puede ser utilizado por todos los participantes. Por ejemplo, la razón fundamental por la que la mayoría de las grandes compañías de hardware (como IBM, Compaq y HP) están respaldando a Linux es porque obtienen el máximo de sus beneficios de su hardware y de los servicios que ofrecen; sus sistemas operativos sólo están ahí para vender el hardware. Se ha especulado con que algunas empresas incluso pierden dinero al desarrollar sus sistemas operativos.

Para este tipo de empresas, el coste de desarrollo de un sistema operativo es realmente un handicap, y mediante la puesta en común de sus recursos pueden producir colectivamente un sistema operativo mejor de lo que podrían individualmente, permitiéndose centrarse en su hardware y los servicios ofrecidos (que son su corazón y su motor). Por ejemplo, si cinco empresas gastaran cinco millones de euros al año cada una en el desarrollo de un sistema operativo, con el modelo de código abierto podrían aunar su financiación de desarrollo y producir colectivamente un sistema de igual o mejor calidad, con menor coste para cada una de las empresas participantes.

Mientras tanto, por supuesto, los usuarios privados independientes obtienen los mismos beneficios de las mejoras. Es esta sinergia la que hace que el modelo de software libre sea beneficioso para los usuarios individuales y el modelo de código abierto tan convincente para los negocios.

Licencias de código abierto

Hasta ahora, hemos visto cómo funciona el software de código abierto y cómo beneficia a los grupos implicados Pero, ¿qué es exactamente lo que lo mueve? La respuesta está en la licencia. El software de código abierto se aprovecha de las leyes existentes sobre el copyright para promulgar un marco de autorización de licencias en el que esencialmente se pide a los usuarios del software que jueguen limpio. Como todos se encuentran jugando en un terreno del mismo nivel, no hay ningún riesgo de que un colaborador pueda contaminar el campo común y usurpar el duro trabajo de todos los demás; esto convierte al software de código abierto en seguro

para todos los involucrados y anima a la gente a colaborar. En esta sección se discutirá el proceso de obtención de licencias y se describirá alguno de los ejemplos más comunes.

Contrariamente a una creencia demasiado común, el software de código abierto no supone lo mismo que si fuera de dominio público. Algunas personas utilizan los términos "dominio público" y "código abierto" (o "software libre") como si fueran intercambiables, pero no lo son. El software de dominio público es precisamente eso: pertenece al público. La persona o grupo que escribió ese software ha decidido renunciar a su propiedad y darlo al dominio público. Cualquier otra persona o grupo puede tomar una pieza de software de dominio público y hacer lo que desee con ella. Esto quiere decir que nada impide que el software de dominio público vuelva a ser utilizado dentro del software de marca comercial.

El software libre, en contraste, no renuncia a la propiedad. El modelo más frecuente en la venta de software es en la actualidad la autorización. Cuando compra (o recibe) una pieza de software, lo que realmente se posee es el permiso para utilizar el software, no el software propiamente dicho. La entidad que le suministró el software aún lo posee, y lo que puede y no puede hacer con el software queda determinado por los términos de la licencia que aceptó cuando recibió el software.

Típicamente, las empresas venden el software a los usuarios y se usa la licencia para impedir que el usuario pueda dar una copia a alguien más (de este modo, otros usuarios que deseen utilizar el software deben comprarlo por sí mismos a la compañía que lo posee). El software libre, sin embargo, sencillamente hace uso de la licencia para asegurar que el software y su código fuente no están obstaculizados por restricciones sobre la utilización o la modificación. Básicamente se reduce a un hecho clave: cada vez que alguien modifica el software y distribuye esa versión modificada, también tiene que distribuir el código fuente de las modificaciones realizadas, sin ningún tipo de restricción adicional sobre los permisos.

Por supuesto, a veces no es tan sencillo como eso. Por ejemplo, algunas licencias no necesitan que los usuarios que contribuyen con modificaciones del software cedan el copyright de sus contribuciones al poseedor de la licencia original. Esto quiere decir que si Luis hace público software bajo una licencia libre y Patricia le envía un cambio, Patricia aún posee el copyright de su envío. Incluso aunque se encuentre bajo la misma licencia que el software original, Luis no puede decidir otorgar un permiso sobre el software a Ricardo amparándose en una licencia diferente sin la aprobación de Patricia, porque ahora Patricia posee una parte del software y Luis no puede hacer caso omiso de ello. Luis podrá, por supuesto, decidir no incluir la contribución de Patricia en la versión que autorice para Ricardo, pero a su vez Patricia siempre podrá simplemente redistribuir su propia versión derivada del software de Luis a Ricardo. Para evitar este problema, algunos proyectos y licencias toman sus medidas para impedir que se produzcan todos estos asuntos de licencias mezcladas.

El resultado es que el software libre y el software de código abierto poseen licencias y copyright reales y legítimos, las cuales pueden resultar tan simples y tan complejas como las leyes del copyright lo permitan. En el momento de la elabora-

ción de este libro, ninguna de las licencias de código abierto o software libre más comunes han sido llevadas ante un juzgado, así que nadie sabe realmente con seguridad si se mantendrían en pie.

En general, el software libre y el software de código abierto describe lo mismo. Aún así, varían ligeramente en sus definiciones, y a veces una licencia que clasifica como de código abierto según la definición especificada por la Iniciativa de Código Abierto (OSI) no lo hace tan bien como software libre de acuerdo con la definición que especifica la FSF. No obstante, este tipo de licencias son relativamente poco comunes.

Nota

La Iniciativa de Código Abierto (*Open Source Initiative*, OSI), cuya dirección web es www.opensource.org, es una organización sin ánimo de lucro que fomenta el software de código abierto y ofrece servicios de certificación para licencias de código abierto.

Aquí hay unos cuantos ejemplos de licencias de código abierto reales, con breves comparativas y resúmenes. Esos ejemplos explicarán algunas de las variaciones principales entre las licencias de código abierto; la mayor parte de las licencias se pueden comparar con alguno de estos ejemplos o seleccionarán características de ellos.

La Licencia Pública General (GPL) de GNU

La Licencia Pública General (*General Public License*, GPL) es la licencia con la que se lanzan la mayoría de los proyectos GNU. Algunas veces se refieren a ella como "*copyleft*" (para hacer énfasis en su diferencia con respecto al copyright). La licencia GPL fue realmente la primera licencia de software libre y, fiel a su herencia, sigue siendo una de las licencias más estrictas que se puedan encontrar.

El texto completo de la licencia GPL se puede encontrar en www.gnu.org/copyleft/gpl.html, pero, a modo de resumen, dice esto:

- No hay restricciones en la utilización o redistribución del software (aunque se pueda cobrar una suma de dinero debido a los costes de distribución y manipulación).
- Si el usuario redistribuye el software, el código fuente debe estar disponible.
- El usuario puede modificar el software de cara a su uso privado.
- Si el usuario modifica el software y distribuye la versión alterada, el código fuente debe ser accesible.
- Si el usuario modifica el software y distribuye la versión alterada, dicha versión debe ser accesible bajo los mismos términos.

- Si el usuario incorpora el software en un trabajo más amplio, el trabajo más amplio también debe ampararse bajo esta licencia o una más rígida.

Por supuesto, antes de tomar cualquier decisión importante, debería leer la licencia GPL por sí mismo en lugar de creer a pies juntillas lo que dice este libro al respecto.

El último apartado acerca de los trabajos más amplios quiere decir que si Patricia incorpora el Software de Luis en su propio SuperSoftware (incluso si no modifica en absoluto el Software de Luis sino que se limita a utilizarlo), éste también tiene que estar disponible con la licencia GPL o una licencia más estricta. A veces se denomina a esta una licencia "vírica", ya que "infecta" el software de Patricia, al exigir que también utilice la licencia GPL.

Por otro lado, esto anima a los desarrolladores que están construyendo su software basándose en software libre a que su software también sea libre. Sin embargo, reduce el campo de uso; por ejemplo, una empresa puede querer utilizar software GPL como parte de su producto, pero no podrán porque quieren vender su producto. Esto hace que GPL sea probablemente la licencia más fuerte (es decir, más libre), pero su hostilidad ante un uso comercial es un motivo importante para la creación de otras licencias.

La licencia BSD

La licencia de distribución de software de Berkeley (*Berkeley Software Distribution*, BSD) presenta un marcado contraste con la GPL. Donde la licencia GPL exige un uso gratuito, pide que el código se distribuya junto con el software e "infecta" trabajos derivados, la licencia BSD no hace nada de esto.

La licencia BSD es muy simple: sencillamente dice que si el software se redistribuye en forma de código fuente, o que si una versión modificada o derivada se distribuye en forma de código fuente, entonces los nombres de los autores originales y su aviso sobre el copyright deberán aparecer en el código que se redistribuye. Si el software (o una versión modificada o derivada) se distribuye en formato binario, entonces los autores deberán recibir el reconocimiento debido en la documentación o en el software mismo.

La licencia BSD es extremada y deliberadamente amigable con el uso comercial. Permite al desarrollador incorporar el software dentro de otro trabajo más amplio, con el único requisito de otorgar a los autores originales el debido reconocimiento. La filosofía de la licencia BSD es un poco más académicamente pura que la de GPL; los desarrolladores de BSD simplemente están interesados en producir el mejor código posible, sin importar en qué se emplea.

La licencia BSD está muchísimo más alejada del ideal del software libre que la licencia GPL puesto que permite a los titulares de la licencia incorporar el software dentro de trabajos de marcas comerciales (aunque se deban reconocer los méritos apropiados).. No obstante, sigue siendo una licencia de código abierto, y es más amigable con el mundo de los negocios.

La Licencia Pública Menos General (LGPL) de GNU

La Licencia Pública Menos General (*Lesser General Public License*, LGPL) es una versión ligeramente relajada de la licencia GPL y se encuentra en algún punto a medio camino entre los dos extremos que acabamos de mostrar. La licencia LGPL se creó como respuesta ante los problemas relacionados con las bibliotecas de enlace estático que aparecían con la licencia GPL. Una biblioteca de enlace estático es aquella en que el código está físicamente incrustado en un programa. La otra posibilidad es una biblioteca de enlace dinámico, en la que un determinado programa únicamente hace referencia a esa biblioteca, de la que únicamente existe una copia que se comparte entre todos los programas que la utilizan. Hay puntos a favor y en contra para ambos tipos de aproximación al problema.

Una biblioteca de enlace estático (como las bibliotecas C de sistema o las API de gráficos) se copia físicamente dentro del código binario generado y es necesaria para que los programas se ejecuten, de modo que cualquier programa que utilice estas bibliotecas implícitamente se funde con la biblioteca y se convierte en un "trabajo más amplio". Como la licencia GPL es de tipo vírico (como se ha mencionado anteriormente en este capítulo), cualquier programa que se enlace estáticamente contra una librería cubierta por la licencia GPL deberá hacerse público bajo la misma licencia (o una más estricta), ¡aunque la biblioteca se haya utilizado sin haberla modificado en absoluto!

Se juzgó que este efecto era demasiado extremo, así que se creó una licencia denominada Licencia Pública Menos General. Esta licencia es idéntica a la GPL, excepto en que no presenta su carácter vírico; relaja la cláusula por la que se exige que los trabajos más amplios también hagan uso de la licencia bajo los mismo términos.

Al no ser vírica, la licencia LGPL tiene más en común con la licencia BSD que con la GPL. Sin embargo, supone una ligera debilitación del ideal del software libre, por lo que la FSF generalmente recomienda el uso de la licencia GPL en lugar de la LGPL, excepto en los casos en que resulte estrictamente necesario.

La Licencia Pública Mozilla

Cuando Netscape Communications Corporation decidió lanzar su producto Communicator como código abierto, echaron un vistazo a las licencias de código abierto existentes. Sin embargo, la licencia GPL resultaba demasiado estricta (ya que querían que su código se pudiera utilizar dentro de trabajos más amplios) y la licencia BSD demasiado débil (ya que deseaban preservar el requisito de que las versiones derivadas y modificadas incluyeran el código fuente). La licencia que Netscape produjo para paliar esta situación se denominó Licencia Pública Mozilla (*Mozilla Public License*, MPL), y es muy parecida a la LGPL, excepto en que es un poco más liberal a la hora de determinar qué constituye un trabajo más amplio. La licencia MPL incluye cláusulas adicionales que establecen más protección contra colaboradores deshonestos; por ejemplo, una cláusula impide que nadie contribuya

con código patentado y después intente utilizar la patente contra los usuarios del software o contra otros colaboradores o titulares de la licencia. (La licencia GPL tiene cláusulas similares para la protección frente a colaboradores deshonestos, pero es menos explícita en los detalles.)

Otras licencias

La mayoría de las demás licencias son más o menos similares a alguna de las licencias tratadas aquí. La licencia Apache, por ejemplo, es muy similar a la licencia BSD, aunque restringe a los titulares de la licencia el uso del nombre "Apache" en el nombre de su producto (la licencia BSD únicamente restringe el uso del nombre en publicidad y promociones).Existen numerosas licencias ahí afuera, demasiadas para seguir la pista de todas ellas. De todos modos, por lo general, la mayor parte son variantes o muy similares a alguna de las licencias aquí comentadas. Si la OSI ha certificado que una determinada licencia es "código abierto", entonces ese es un buen punto de referencia; en cualquier caso, todo lo que tiene que hacer es leer la licencia y decidir si le gusta.

Licencia para programar

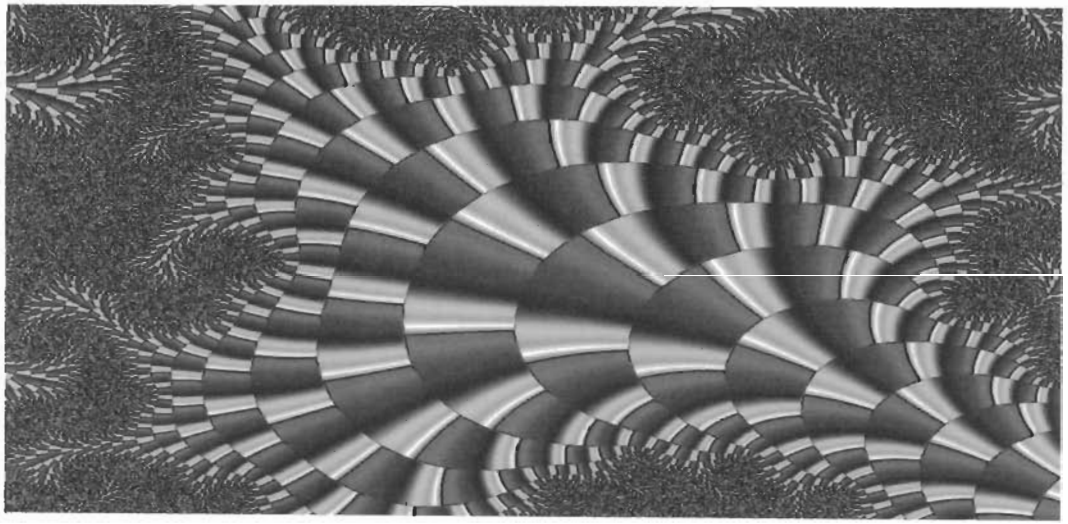
Así que, ¿qué significa todo esto para las personas únicamente interesadas en usar (y ajustar y personalizar) un sistema Linux? Bien, lo que todas las licencias de código abierto y software libre tienen en común es que su uso es libre para cualquier propósito. La mayoría de los problemas que surgen cuando se trata con licencias de software abierto suelen estar relacionados con la redistribución y la modificación. Los grupos que simplemente utilizan el software están a salvo.

Pero, con todas estas cuestiones y esta retórica sobre el software libre frente al software propietario, ¿deberíamos estar realmente tan despreocupados sobre las licencias que utilizamos? La respuesta es que depende de cada cual. Hay quien pone sus objeciones a las pruebas de toxicidad de cosméticos sobre animales. Aparte de denunciar a estas empresas si violan la ley, el único recurso disponible para activistas de este tipo es boicotear los productos de las empresas de las que están en contra. Del mismo modo, si no está de acuerdo con una licencia de software en particular, sencillamente puede escoger no utilizar ese software. Si no puede trabajar sin él, entonces puede escribir (o encargar) su propia versión bajo la licencia que prefiera.

Animamos a los usuarios que de verdad se preocupan (o quizás sólo sienten curiosidad) por las licencias de software libre a que visiten los sitios web de la Fundación para el Software Libre y la Iniciativa de Código Abierto, lean sus respectivas posturas y definiciones y hagan, si quieren, una donación. La mejor opción es siempre la que escoge uno mismo.

Parte II

Distribuciones Linux



3

La naturaleza de una distribución

En el capítulo 1 aprendió lo que es un sistema Linux, de qué partes se compone, y cómo encajan entre sí. Esto debería suponer bastante trecho recorrido en el camino hacia la comprensión de los sistemas Linux en general, ¿cierto?

Bien, así es, pero hay más cosas que tienen que ver. Existen diversas versiones para cada uno de los componentes de un sistema Linux que hemos visto. El núcleo, las bibliotecas y otras partes del sistema suelen tener diferentes versiones "activas"; algunas son estables y seguras pero ya no son "actuales", mientras que las versiones más nuevas pueden estar plagadas de errores o ser inestables. Además, a veces existen opciones rivales para el mismo programa, como en el caso del software de escritorio, donde tanto GNOME como KDE son excelentes elecciones.

La responsabilidad de un distribuidor de Linux es producir un sistema que funcione. Esto quiere decir seleccionar las mejores versiones y productos de software posibles para cumplir las tareas que vayan surgiendo. Una configuración particular de una colección de software de este estilo se conoce con el nombre de distribución. Este capítulo tratará acerca de la idea de una distribución y qué es lo que la forma, y describirá el modo en que se crean las distribuciones.

Una instantánea del Continuo Linux

La expresión "Continuo Linux" no proviene de un libro de ciencia ficción ni se trata de un rito de algún culto extraño. Es sencillamente un modo de describir el

estado global del software que colectivamente se convierte en un sistema Linux. Todo este software no se encuentra en un árbol del que crezca, ni cae del cielo como el maná. Realmente crece en un millón de árboles diferentes a la vez y el verdadero maná a veces se encuentra oculto entre el lodo y la mugre. Puede resultar muy duro recolectar todo este software y construir con todo ello un sistema utilizable. Esto es lo que hacen los distribuidores Linux y el resultado final es una distribución.

La gran mayoría de los paquetes de software no se ha escrito explícitamente para formar parte de una distribución Linux en particular. Debido a esto, en un determinado momento, cada uno de estos paquetes de software tiene su propia versión, e incluso su propio esquema de versiones. Por ejemplo, en el momento de escribir esto, la versión actual del editor emacs es la 21.2, mientras que la versión estable actual de Xfree86 es la 4.2.0.

La figura 3.1 muestra que lo que realmente hace una distribución es tomar una muestra (o una "instantánea") del más amplio continuo global de software Linux. En esta figura se puede ver cómo Red Hat Linux 7.3 se compone de una serie de distintos paquetes, cada cual con su propia versión. Obviamente, Red Hat Linux consiste en mucho más paquetes que los que muestra la figura 3.1; a pesar de eso, en esta figura se pueden ver tres paquetes de software importantes:

- La implementación Xfree86 del sistema X Window.
- El núcleo Linux.
- El escritorio KDE.

Cada uno de esto paquetes tiene su propia historia de versiones, ciclo de lanzamientos, etc. Mientras tanto, Red Hat Linux, construye su distribución a partir de versiones particulares del software. Los tres paquetes representan un continuo de ejemplo, mientras que Red Hat Linux representa una instantánea de ese continuo. En este caso, Red Hat Linux 7.3 incluye Xfree86 4.2.0, Linux 2.4.18 y KDE 3.0. Recuerde que esta es una instantánea del continuo y no cambia a lo largo del tiempo (no suele hacerlo); Red Hat puede escoger la última versión disponible de un paquete en el instante de su lanzamiento, pero pueden aparecer versiones posteriores tras la toma de esa instantánea.

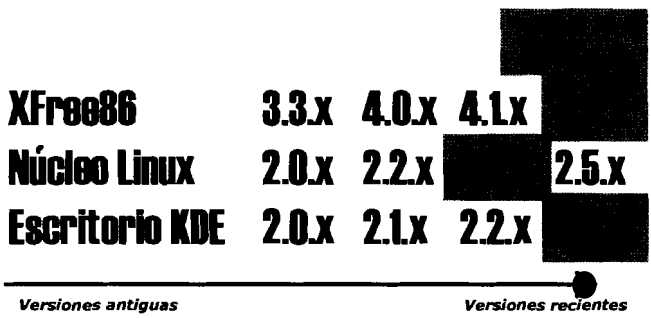


Figura 3.1. Ejemplo del continuo software.

Nota

Una distribución Linux es una colección de software que forma un sistema operativo basado en Linux. Como cada uno de los paquetes de software en una distribución tiene su propia versión independiente, el número de versión que el distribuidor asigna a su distribución no mantiene ninguna relación con las versiones de los paquetes de software de que está compuesta.

Si todos estos paquetes de software y versiones comienzan a confundirse, borrosos en su mente, es genial, porque esta es la manera perfecta de ver las cosas. Realmente puede ver el estado del software de código abierto como un gran "espacio" de software y números de versión: un continuo. Este continuo está en constante evolución y cambio según se desarrollan los paquetes de software individuales y cambian las versiones. Lo que un fabricante llama un lanzamiento con calidad de producción es en realidad una instantánea tomada a este continuo en un momento determinado.

La responsabilidad de un fabricante de distribuciones es trabajar con esta instantánea para que el sistema global resulte tan útil, flexible y esté tan libre de errores como sea posible. Los fabricantes hacen esto identificando en primer lugar los paquetes de software que van a ir dentro de la distribución y seleccionando después la versión estable más reciente de ese software que satisfaga sus objetivos.

Los objetivos de una distribución

Los sistemas Linux son extremadamente flexibles. Linux (el núcleo) puede ampliarse hasta grandes sistemas multiprocesador, o reducirse para funcionar en los PDA, con sus limitaciones de memoria. Obviamente, los requisitos software para una distribución que se ejecute en el sistema multiprocesador será bastante diferente de los requisitos software para un PDA, a no ser, por supuesto, que exista alguien tan afortunado como para tener un PDA con 32 procesadores y 8 GB de memoria (¡que no se desintegre debido a sufrir una combustión espontánea!).

Por ello, la primera decisión que el productor de una distribución debe tomar es cuáles son, exactamente, los objetivos de una distribución. ¿Es una distribución que se va a dedicar a servidores o a estaciones de trabajo? ¿Es el objetivo un entorno de usuario muy amigable que se adecue a la utilización en un entorno de oficinas, o su objetivo es un entorno Unix tradicional de altas prestaciones para ingenieros? ¿Se supone que la distribución se ejecutará en un ordenador Intel, Alpha, PowerPC, SPARC, MIPS, ARM o IBM S/390?

Esta es, por supuesto, una decisión de mercado (ya sea el esfuerzo de una entidad comercial o de un grupo de voluntarios), y distintos productores tienen como objetivo distintos mercados. Red Hat Software, por ejemplo, produce una distribución bien acabada válida para el uso tanto como servidor que como sistema de

escritorio. El Proyecto Debian produce una distribución similar en el ámbito de Red Hat, pero que consiste enteramente en software libre. Mandrake Software produce una distribución enfocada al uso de escritorio, y TurboLinux una distribución de alta calidad para el proceso en paralelo y distribuido. La distribución Slackware apunta hacia la creación de un sistema tradicional similar a Unix. En otras palabras, el mercado que cada fabricante tiene en el punto de mira determina los requisitos globales de la distribución y la selección particular de software que incluirá.

Al mismo tiempo, cada usuario tiene sus necesidades específicas. Algunos usuarios necesitan preparar una estación de trabajo para su uso personal, mientras que otros necesitan montar un servidor local de archivos, y aún habrá otros que necesiten componer una gran granja de servidores para que se encargue del funcionamiento de un sitio web.

Algunas distribuciones de Linux, por supuesto, son mejores en determinado tipo de tareas y otras son mejores en otros campos. A veces, el usuario tiene el lujo de poder escoger la distribución que mejor se adapte a la tarea. Sin embargo, frecuentemente el usuario prefiere emplear una distribución específica (quizás por preferencias personales o por mandato corporativo) que puede no haber sido optimizada para la tarea tal y como está conformada. En estos casos, una gran ayuda para el usuario es comprender cómo se ha construido la distribución y cuáles son sus objetivos para poder personalizar el sistema para la tarea deseada. Este libro ayudará al lector a comprender y personalizar el sistema, pero depende del lector el saber cuáles son sus propios objetivos y necesidades.

Este libro comenta tres distribuciones de ejemplo:

- Red Hat Linux
- Slackware Linux
- Debian GNU/Linux

Hemos escogido estas distribuciones debido a la inmensa influencia que tiene en la evolución de los sistema Linux; casi cualquier otra distribución que exista es un derivado suyo, se ha inspirado en ellas, o está relacionada con una de las tres. A pesar de sus diversos enfoques, cada una de estas distribuciones consiste en el mismo conjunto general de software; son instantáneas del mismo continuo, sólo que desde distintos ángulos. El resto de la segunda parte (del capítulo 4 al 6) se dedica a comentar cada una de estas distribuciones en detalle, pero antes debe entender exactamente lo que es una distribución; que será en lo que nos centremos en el resto de este capítulo.

Disección de una distribución Linux

Esta sección describirá los componentes claves de una distribución Linux, que son:

- Versión del núcleo Linux
- Formato de empaquetado
- Estructura del sistema de archivos
- Guiones de arranque del sistema
- Versiones de las bibliotecas de sistema
- Escritorio X Window
- Software de usuario

Tras la lectura de esta sección, los lectores comprenderán las bases de lo que se incluye en una distribución; en capítulos posteriores se tratarán detalladamente distribuciones específicas de ejemplo, pero esta sección proporciona una introducción. Los usuarios que ya conozcan una distribución Linux o sean usuarios experimentados de Unix deberían poder simplemente hojear o saltarse este capítulo y avanzar hasta las explicaciones detalladas de las distribuciones de ejemplo, comenzando por la distribución Red Hat Linux en el capítulo 4.

Núcleo Linux

El núcleo Linux es el corazón del sistema operativo (véase el capítulo 1 para obtener una explicación sobre el sistema operativo Linux, si es que no lo ha hecho ya). En general, el núcleo Linux tiene dos "árboles": la versión estable y la versión de desarrollo. La versión estable es un núcleo listo para la producción, está libre de errores (tanto como sea posible) y ha sido probado por millones de usuarios a lo ancho y largo del mundo. La versión de desarrollo, es, en cambio, dónde los desarrolladores principales del núcleo realizan sus ajustes y mejoras de características. El árbol de desarrollo es por ello muy inestable y no sirve para los sistemas de producción, el uso diario o personas con tendencia a la apoplejía.

El núcleo Linux tiene un esquema de numeración de X.Y.Z, que se describe a continuación:

- X es la versión principal (*major version*), que representa cambios en la arquitectura importantes.
- Y es la versión secundaria (*minor version*), que indica mejoras funcionales pero pocos cambios de arquitectura.
- Z es la versión de construcción (*build version*), que representa niveles de eliminación de errores o de parches de desarrollo.

Si Y es un número par, el núcleo es estable; si es impar, el núcleo se encuentra en desarrollo. La serie de núcleos estables que se corresponde con el momento de esta escritura es la 2.4.x, y la serie de desarrollo es la 2.5.x. (Una "serie" es sencillamente un conjunto de versiones similares.)

Hablando de un modo general, los fabricantes de distribuciones están interesados en producir sistemas estables y libres de errores. Por este motivo cualquier lanzamiento real debería utilizar exclusivamente núcleos Linux estables; en otro caso, los fabricantes se arriesgan a exponer a sus usuarios a una cantidad excesiva de errores en el núcleo, la parte más sensible y crítica del sistema.

Por otra parte, el núcleo Linux, como muchos otros proyectos de software abierto, tiene un ciclo de lanzamiento muy largo si se se compara con muchos de los esfuerzos comerciales. Esto quiere decir que muchos fabricantes de distribuciones suelen descubrir que necesitan características que son nuevas o aún están en fase experimental, y que sólo existen en los últimos núcleos de desarrollo. Los fabricantes de distribuciones suelen tener que decidir si escogerán un núcleo estable pero con menos características, o una versión mucho mejor equipada pero probablemente menos estable.

Muchos fabricantes de distribuciones adoptan un compromiso, mediante la retroadaptación de características de los núcleos de desarrollo a núcleos de aparición previa. (El proceso de añadir este tipo de características avanzadas a un núcleo antiguo se conoce como retroadaptación, o, en inglés, *backporting*.) El núcleo de desarrollo que se va a convertir en el próximo núcleo estable deriva originalmente de la versión estable actual, de modo que aún comparte muchas similitudes con el núcleo antiguo. Así, muchas de las características y resolución de errores que se han añadido a un núcleo de desarrollo típico no son estrictamente incompatibles con el núcleo estable previo.

Los desarrolladores principales del núcleo (liderados por Linus Torvalds) no encuentran suficiente tiempo como para probar y encargarse de estas retroadaptaciones a núcleos antiguos; ya emplean todo el tiempo de que pueden disponer en mejorar el nuevo núcleo. Por eso muchos fabricantes de distribuciones se encargan de ese trabajo y lanzan núcleos personalizados junto con sus productos. Por ejemplo, el soporte al bus serie universal (USB) se añadió al núcleo Linux en la serie de desarrollo 2.3, pero Red Hat quería ofrecer soporte a USB en su distribución Red Hat Linux 7.0 que se basa en la serie 2.2. Decidieron realizar el trabajo de la retroadaptación de la funcionalidad USB de la versión 2.3 e incluyeron en Red Hat 7.0 este núcleo personalizado. A veces los fabricantes de distribuciones añaden incluso funcionalidades que no se encuentran en ningún núcleo, ni en el árbol de desarrollo ni en el árbol estable; por ejemplo, Mandrake Software tiene parches para el núcleo que hacen que los dispositivos extraíbles se administren más fácilmente. Esto no es un comportamiento aislado, y muchas distribuciones aparecen con núcleos particularizados.

No siempre es posible la retroadaptación de características de los núcleos de desarrollo, por supuesto; A veces las nuevas características de un núcleo de desarrollo dependen de cambios en la arquitectura que se hicieron en el núcleo y que no existen en versiones estables previas, y por eso no pueden ser retroadaptados. Además, en muchos casos las características de desarrollo se consideran inestables únicamente porque son nuevas y no se han comprobado de manera intensiva. Algunos fabricantes pueden no estar de acuerdo y considerar estas características

como válidas para producción o pueden disponer de características propias que quisieran incluir. En estos casos, los fabricantes pueden escoger incluir un núcleo particularizado con su distribución.

Hay quien desaprueba este comportamiento, haciendo referencia al peligro de fragmentar el núcleo en muchos derivados particulares y a los problemas de compatibilidad que pueden surgir cuando los fabricantes particularizan un núcleo tanto que difiere de los núcleos "estándar" de un modo sutil. Sin embargo, se puede ver el núcleo "oficial" como una especie de punto de partida para que los fabricantes de distribuciones comiencen a desarrollar sus sistemas. Después de todo, ¿realmente le preocupa si las características USB que utiliza provienen del núcleo 2.3 o del núcleo 2.2 si su cámara digital con conexión USB puede funcionar?

Formato de empaquetado

El formato de empaquetado de una distribución es el mecanismo que se emplea para instalar y administrar el software que viene junto con la distribución. Muchas de estas aplicaciones de software están situadas de manera contigua en los mismos directorios, de modo que resulta muy útil disponer de un método para determinar qué archivos en particular fueron instalados por qué paquetes. Los gestores de paquetes también proporcionan la habilidad de desinstalar o actualizar el software y de preguntar al sistema acerca de qué software se instaló.

Las herramientas de administración de paquetes también proporcionan en general información sobre las versiones y las dependencias. Por ejemplo, una versión del escritorio X Windows KDE en particular necesita una versión particular de la biblioteca de componentes Qt. Para que KDE se pueda ejecutar adecuadamente, la versión apropiada de Qt también tiene que estar instalada, pero determinar información sobre las versiones no siempre resulta sencillo. Los gestores de paquetes pueden seguirle la pista a esta información, ayudando a los usuarios y a los administradores del sistema a mantener su software y sus versiones en sincronía.

Cada sabor de Unix, incluyendo los sabores comerciales, tiene su propio formato de empaquetado o su mecanismo de mantenimiento del software. No obstante, como en muchas otras áreas, los formatos de paquetes de Linux son mucho más funcionales y sencillos de utilizar que la mayoría de sus equivalentes comerciales de Unix. Es más, algunos fabricantes comerciales de Unix incluso han anunciando su intención de incluir el formato de empaquetado RPM dentro de sus propios sistemas.

Gestor de paquetes de Red Hat

El primer gestor de empaquetado sofisticado para los sistemas Linux en aparecer fue el gestor de paquetes de Red Hat (*Red Hat Package Manager*, RPM), desarrollado por Red Hat Software para sus distribuciones. RPM soporta todas las tareas de empaquetado estándar y tiene una amplia lista de características. Instala, desinstala y actualiza paquetes, ejecuta guiones durante las instalaciones y actua-

lizaciones que realizan un trabajo adicional de configuración del software, y soporta el seguimiento de dependencia de paquetes y el respeto de versiones. RPM también soporta la investigación extensa sobre contenidos y documentación de paquetes, ofrece servicios de verificación de la integridad de los paquetes instalados e incluso puede generar archivos "RPM fuentes" como ayuda a los usuarios que quieren o necesitan personalizar los RPM. La mayor contribución de RPM al software de empaquetado probablemente sea su habilidad para realizar actualizaciones de paquetes y amplias comprobaciones de dependencia de manera fiable.

Archivos Deb

El proyecto Debian tiene su propio formato alternativo de empaquetado. Como estos archivos tienen las extensión .deb, frecuentemente se conocen como "archivos Deb". El formato de empaquetado de Debian es similar en prestaciones a RPM, pero dispone de algunas características adicionales, como una capacidad mucho más amplia de utilización de guiones y la habilidad de crear metapaquetes para compendiar otros paquetes. Los sistemas Debian también disponen de herramientas de interfaz de usuario adicionales pensadas para hacer que el sistema de empaquetado sea más fácil de usar.

RPM y el formato Debian son los dos formatos de empaquetado más populares que se puedan encontrar. La mayor parte de otras distribuciones utilizan uno de estos dos formatos. Una excepción notable es Slackware, que hace uso de su propio formato basado en archivos .tar tradicionales y en bases de datos de búsqueda de texto. También existen herramientas de soporte para estos sistemas de empaquetado que proporcionan características adicionales; por ejemplo, tanto RPM como el sistema Debian disponen de herramientas que inspeccionan diversos servidores en busca de paquetes concretos y los descargan e instalan de manera automática.

Formato de empaquetado frente a distribución

Es importante no confundir el formato de empaquetado con la distribución. Dos distribuciones pueden utilizar el mismo formato de empaquetado y a pesar de ello tener versiones incompatibles del mismo software. Por ejemplo, tanto Caldera como Red Hat utilizan el formato RPM para sus distribuciones. Sin embargo, un RPM creado para un sistema Red Hat no tiene por qué funcionar en un sistema Caldera OpenLinux, que puede tener versiones distintas de las bibliotecas que el paquete necesita. En otras palabras, un formato de empaquetado sólo gestiona la instalación y configuración del software, no el software mismo. El formato de empaquetado no afecta al hecho de que una pieza de software necesite otro componente, sólo le permite instalarlo.

Funcionalmente, no hay tanta diferencia entre los diversos formatos de empaquetado. Todos hacen las mismas cosas. Debido a esto, la elección del formato a utilizar no se realiza según criterios técnicos o de calidad, sino, realmente, de acuerdo con la disponibilidad de paquetes. Esto es, ¿con quién quiere que su paquete sea compatible? En la actualidad, la mayoría del software que se empaqueta para Linux

utiliza el formato RPM. Esto no quiere decir que los usuarios de otros formatos (como Debian) no puedan utilizar el software; para una gran cantidad de software, se pueden encontrar paquetes .deb además de archivos RPM, y siempre dispone de la posibilidad de instalar el software manualmente (o construirlo a partir de su código fuente) sin tener que utilizar ninguna herramienta de empaquetado. No obstante, si se encuentra navegando por la web y descubre al azar un componente software, es más probable que encuentre la versión para Linux empaquetada como RPM que en el formato de Debian.

Incluso esta distinción es discutible, ya que la distribución de Debian también incluye herramientas que pueden convertir archivos RPM en paquetes compatibles con Debian, y viceversa. Como usuario, es mucho más probable que encuentre problemas de compatibilidad con respecto a las bibliotecas necesarias que sobre el formato utilizado para empaquetar el software. Después de todo, sea cual sea el formato que se instale, se acabarán extrayendo los mismos archivos.

Dicho esto, actualmente parece que el formato RPM tiene un impulso más fuerte. Lo utilizan más distribuciones y parece ser el estándar de facto de formato de empaquetado de software para sistemas Linux. De todos modos, si hay algo por lo que se distingue la comunidad de código abierto es porque es dinámica, y esta situación podría cambiar a la semana siguiente de la publicación de este libro.

Estructura del sistema de archivos

Incluso la mejor distribución Linux no sirve para nada si no puede encontrar el archivo que está buscando. Desde los más antiguos días de Linux, las convenciones sobre qué archivos y programas van en qué lugar dentro del disco han supuesto una importante reflexión. Un sistema promedio similar a Unix es mucho más complejo que muchos otros sistemas operativos, y por eso es muy importante tener estándares sobre dónde encontrar las cosas. La convención seguida por un fabricante para un sistema de archivos determinado tiene un gran impacto sobre el resto del sistema.

Existe un número tan grande de modos de estructurar los contenidos de una distribución en un disco que realmente no hay un número fijo de opciones entre las que escoger. El legado Unix ha dejado como patrimonio un conjunto básico de reglas tradicionales acerca de qué programas van en qué directorios, pero como mucho sólo ha sido a grandes rasgos, y tanto los diversos fabricantes de sabores comerciales de Unix tanto como los de sistemas libres han seguido estas imprecisas reglas con un grado variable de rigor.

Directorios

El directorio /etc de los sistemas Unix contiene generalmente archivos de configuración, guiones y contenidos similares. Aún así, los sistemas Solaris también colocan algunos programas de administración en este directorio, mientras que los sistemas Linux y BSD sitúan típicamente dichos programas en /sbin. Además,

algunas veces es el software mismo el que determina el lugar en que se situarán sus archivos: en los sistemas BSD, algún software de terceros como Apache pone sus archivos de configuración en /usr/local/etc, mientras que en la mayoría de los sistemas Linux incluso el software de terceros coloca sus archivos de configuración en /etc.

Estándar de jerarquía del sistema de archivos

Existe una convención ostentosa en la estructura de directorios de los sistemas Linux. El estándar de jerarquía del sistema de archivos (*Filesystem Hierarchy Standard*, FHS) encara la confusión que rodea a las estructuras de directorios en los sistemas Unix y establece un conjunto de directrices estándar sobre qué tipo de archivos deberían estar presentes en los directorios. El FHS tiene como objetivo ser aplicable a cualquier sistema parecido a Unix, aunque se creó y adecuó para los sistemas Linux. La mayor parte de las distribuciones Linux siguen el FHS razonablemente bien, aunque existen algunas excepciones. Los sistemas de código abierto BSD también siguen en líneas generales el FHS, pero los fabricantes comerciales están demostrando ser algo más lentos en la adopción del FHS, probablemente debido a temas de compatibilidad hacia atrás.

La tabla 3.1 resume los aspectos principales del FHS, mostrando los directorios más importantes identificados por el FHS. Este es un buen punto de arranque, pero los usuarios que verdaderamente quieran dominar el "Zen de Linux" deberían leer el documento completo. El FHS se puede encontrar en www.pathname.com/fhs/. Una vez que haya digerido el FHS, habrá dado el primer paso hacia la comprensión de un sistema Linux, que se trata sencillamente de saber dónde encontrar las cosas.

Tabla 3.1. Resumen del estándar de jerarquía del sistema de archivos.

Directorio	Nombre	Contenidos típicos	Ejemplos
/etc	"etcetera"	Archivos de configuración	/etc/passwd: información de la cuenta de usuario; /etc/resolv.conf: información de búsqueda de red
/boot	Archivos de arranque	Archivos necesarios para que arranque el sistema	/boot/vmlinuz: imagen del núcleo Linux
/dev	Nodos de dispositivos	Archivos que representan dispositivos hardware del sistema	/dev/sda: el primer disco SCSI del sistema; /dev/audio: el dispositivo de sonido
/lib	Bibliotecas de sistema	Archivos de bibliotecas compartidas por los programas de usuario	/lib/libc.so: la biblioteca compartida (.so) para programas C; /lib/libpthread.so.0: la biblioteca de hilos POSIX
/bin	Binarios de programas	Programas de aplicaciones de usuario	/bin/ls: la utilidad de listado de archivos; /bin/bash: la shell de comandos "*Bourne Again Shell*"

(Continuación)

Directorio	Nombre	Contenidos típicos	Ejemplos
/sbin	Administrativo	Programas utilizados principalmente para tareas administrativas	/sbin/ifconfig: herramienta de configuración de dispositivos de red; /sbin/mkfs: herramienta de formato del sistema de archivos
/home	Casa del usuario	Directorios locales para los usuarios	(Opcional) /home/pgomez: el directorio local para el usuario "pgomez"
/mnt	Puntos de montaje	Contiene subdirectorios donde se pueden montar sistemas de archivo temporales	/mnt/cdrom: punto de montaje para el lector de CD-ROM; /mnt/floppy: punto de montaje para la disquetera
/usr	Partición del usuario	Archivos útiles para los usuarios una vez que el sistema se está ejecutando	/usr/X11R6: archivos relacionados con el sistema X Windows; /usr/bin: binarios de programa para el usuario final
/var	Programas en tiempo de ejecución	Archivos y directorios para almacenar datos o información de control para programas y servicios	/var/spool/mail: archivos relacionados con el correo electrónico /var/log: archivos de anotaciones del sistema
/tmp	Archivos temporales	Archivos que no son persistente o importantes	

Una pequeña explicación de la tabla 3.1 no está de más, especialmente en lo que respecta al directorio "/" y al directorio "/usr". Las rutas de Unix comienzan con un carácter "/", y el directorio de más alto nivel es el mismo "/". Se conoce como directorio o partición raíz (o directorio o partición *root*); si alguien le pide "haz cd hacia el directorio raíz", realmente le están pidiendo que ejecute el comando `cd /`. Del mimo modo, el directorio /usr se pronuncia "barra usuario" (*slash user*).

Cuando lea el FHS, probablemente se dará cuenta de que define dos copias de la mayoría de los directorios, por ejemplo, /bin y /usr/bin. El motivo para esto está relacionado con el modo en que se arranca el sistema. Durante el arranque, los sistemas Unix montan en primer lugar la partición raíz, y sólo montan las particiones de usuario mucho más adelante en este proceso. También un administrador arrancará en ocasiones el sistema en el modo monousuario o de mantenimiento para llevar a cabo actualizaciones importantes, el mantenimiento de la configuración, o análisis de seguridad. En este caso, el sistema podría detenerse en la partición raíz y no montar ninguna de las otras. En ambos casos, es necesario un conjunto de programas vitales (como ls, mount, cd, vi, echo, etc...) para que funcione el sistema.

Ya que el sistema necesita acceso a estos programas y a archivos vitales antes de montar el resto de los volúmenes, obviamente deben poder encontrarse en la partición raíz.

Por eso, los directorios de más alto nivel como /bin, /lib, y /sbin se encuentran ahí para contener un conjunto básico de comandos necesarios para que el sistema complete su proceso de arranque, mientras que /usr/bin, /usr/lib, y /usr/sbin

contienen programas adicionales que son usados más habitualmente por los administradores en un sistema que ya ha arrancado.

Por ejemplo, el comando `ls` resulta necesario, por eso está en /bin. Sin embargo, el comando `find` no es estrictamente necesario (aunque resulta conveniente), así que se encuentra en /usr/bin. Como regla sencilla, si trata de encontrar un programa o intenta decidir dónde situar un programa nuevo, en primer lugar decida si resulta imprescindible para que funcione el sistema. Si lo es, utilice /bin, /lib, o /sbin; si no lo es, utilice /usr/bin, /usr/lib, o /usr/sbin.

De hecho, hay incluso un tercer caso de este tipo de directorios: el directorio /usr/local se utiliza para el mismo tipo de archivos que /usr (esto es, archivos que no son estrictamente necesarios para que se arranque o funcione el sistema), pero para archivos que son locales al sistema particular. Por ejemplo, el software X Windows es bastante común y estándar a lo largo de todas las instalaciones, así que debería de estar en /usr. Sin embargo, una instalación del servidor web Apache que haya sido particularizada para un sistema específico no es estándar y debería colocarse en /usr/local, en lugar de en /usr.

Hay varias reglas simples que se pueden aplicar cuando se intenta decidir dónde se colocará un archivo en particular. Una aproximación es que cualquier cosa que venía incluida con la distribución debería colocarse en /usr y todo lo demás en /usr/local. Alternativamente, si es gestionado por las herramientas de empaquetado del sistema, se corresponde con /usr; en otro caso, pertenece a /usr/local. El asunto es que el FHS es un estándar ciertamente flexible y puede situar un archivo en cualquier ubicación que quiera, siempre y cuando sea consistente.

El modo en que el FHS divide el sistema de archivos en estos directorios paralelos, casi gemelos, puede parecer excesivamente complicado. No obstante, hay una buena razón para esta complejidad añadida: hacer la vida más fácil a los administradores del sistema y más recuperable en caso de errores de disco. Al dividir el sistema de archivos en directorios vitales (raíz), no vitales (/usr), y locales (/usr/local), el FHS establece un método sencillo para que los administradores coloquen diferentes tipos de archivo en discos diferentes.

Nota: Hacer particiones pensando en las actualizaciones

Recuerde que la colocación de archivos vitales en el directorio raíz permite al sistema arrancar y manejarse sin tener que molestarse con otras particiones del sistema durante el proceso de arranque. Sin embargo, esta separación del directorio raíz y /usr significa que los archivos y programas del sistema central o los del nivel de usuario podrán actualizarse independientemente entre sí. Quizás sea más importante que esta separación también significa que el sistema central (los volúmenes raíz y /usr) se puede actualizar independientemente de /usr/local y /opt. Esto permite a los administradores actualizar un sistema sin interferir con el software personalizado instalado en el sistema, y también significa que el software personalizado puede protegerse si se estropea el disco del sistema principal que contiene los volúmenes raíz y /usr.

(Es sencillo reinstalar un sistema operativo, pero muy complicado recuperar meses de trabajo en el sitio web Apache que estaba funcionando. Es incluso peor si el sitio web sigue intacto pero tiene que sobrescribirlo para reinstalar el sistema operativo porque comparten la misma partición o el mismo disco.)

Guiones de arranque del sistema

¿Para qué vale un ordenador si no puede arrancarse? Siguiendo con esta línea de pensamiento, ¿para qué vale un guión de arranque si no puede hacerse una idea de cómo utilizarlo para arrancar su propio software? Los sistemas Linux, como todos los sistemas Unix, configuran sus actividades de arranque y detención mediante una serie de guiones (*scripts*). El sistema por sí mismo podría quedarse tan tranquilo si simplemente se arrancara y cerrara sin hacer nada más por el usuario, pero resultaría muy aburrido (y peligroso, si quisiera decir que las particiones de disco no se administraran de manera correcta). Los guiones de arranque y detención, denominados en ocasiones guiones de comandos de nivel de ejecución (*runlevel command scripts* o *rc scripts*) realizan el pesado trabajo de arrancar y detener su sistema Linux.

Al ejecutar un sistema Linux , el núcleo es la primera cosa que se carga. Después inicia un programa conocido como init, que es el responsable de ejecutar el resto del sistema. Este programa init se encarga básicamente de gestionar los niveles de ejecución. Un nivel de ejecución es sencillamente una configuración de los programas y servicios que se ejecutarán; normalmente, estas configuraciones están orientadas hacia un objetivo específico, como la operación multiusuario normal, el cierre del sistema, su funcionamiento, cambiar al modo de mantenimiento, etc. Cada nivel de ejecución corresponde a un estado distinto; por ejemplo, el nivel de ejecución 0 podría significar "detener el sistema", mientras que el nivel de ejecución 4 podría corresponder a "modo normal de servidor". El programa init es el responsable de iniciar cualquier servicio (como servidores web, servidores de DNS, bases de datos y otros parecidos) y de preparar el entorno de usuario (como montar todas las particiones, permitir que el usuario entre en el sistema e iniciar X Windows si es necesario).

Al núcleo no le importa cómo init se encarga de esto; init podría realizar el trabajo por sí mismo, por supuesto, pero eso no resulta muy personalizable, por eso en su lugar init delega la mayoría de sus tareas de ejecución en un programa externo. Todo lo que init suele hacer es buscar un archivo de configuración en el directorio /etc (llamado /etc/inittab) que le dice en qué nivel de ejecución debería entrar por defecto y después invoca al programa (casi siempre un guión de shell) apropiado para ese nivel. De hecho, el programa init es sencillamente un parámetro de configuración para el núcleo. El núcleo puede inciar con la misma facilidad otro programa en lugar de init que manifieste un comportamiento distinto para preparar el sistema. Por ejemplo, si el sistema ha sido dañado y necesita recuperarse o

pasar por un proceso de mantenimiento, el núcleo puede sencillamente iniciar una shell de comandos interactiva en lugar de init.

Aún así, normalmente el núcleo inicia init, que ejecuta los guiones para el nivel de ejecución actual. Actualmente hay siete de estos niveles de ejecución del sistema, resumidos en la tabla 3.2. Siempre que init conmuta a un nivel de ejecución nuevo, llama a los guiones para ese nivel, y los guiones se encargan de ejecutar o detener programas según sea necesario. En realidad, init ni siquiera distingue de qué nivel es cada cual; sencillamente tiene seis "modos" en los que puede estar. Por convención de Unix, los niveles de ejecución suelen tener los significados expresados en la tabla 3.2. A veces los niveles varían según la distribución o el sabor de Unix; para descubrir exactamente que niveles soporta su sistema, eche un vistazo al archivo de configuración de init, que suele ser /etc/inittab.

Tabla 3.2. Niveles de ejecución del sistema.

Nivel de ejecución	Significado	Comportamiento
0	Parada	Finaliza todos los programas y servicios, desmonta los sistemas de archivos, y detiene la CPU.
1	Modo monousuario	Finaliza la mayoría de los programas y servicios, permitiendo sólo entradas del superusuario; frecuentemente se utiliza para el mantenimiento del sistema.
2	Multiusuario sin red	Inicia todos los programas y servicios excepto los relacionados con la red; permite a todos los usuarios locales entrar en el sistema.
3	Multiusuario normal	Inicia todos los programas y servicios, excepto X.
4	Multiusuario normal	No se suele usar; excepto eso, es como el nivel 3.
5	Multiusuario gráfico	Como el nivel de ejecución 3, pero también ejecuta el sistema X Window para entrada de usuarios.
6	Reinicio	Finaliza todos los programas y servicios y reinicia el sistema.

Cada nivel de ejecución tiene un significado distinto, que refleja qué programas y servicios iniciarán en ese nivel. Un concepto importante es que se puede entrar a cualquier nivel desde cualquier otro nivel. Por ejemplo, en el nivel de ejecución 1 (modo monousuario) se puede entrar tanto en el arranque del sistema, o desde otro

nivel de ejecución en un sistema activo, como al nivel 3. Desde la perspectiva de init (y la perspectiva de los guiones de shell que init ejecuta), los dos casos son idénticos, y el nivel de ejecución 1 tiene el mismo significado y ejecuta los mismos guiones. En otras palabras, los términos "finaliza" e "inicia" son relativos; si el sistema acaba de iniciar, no hay nada que finalizar, y viceversa. Las maneras en que se indica a init que conmute entre niveles de ejecución varía según el sistema; el método más utilizado es usar el comando `telinit`, pero la mayoría de los sistemas ofrecen comando alternativos como `shutdown`.

Nota

Recuerde, init no realiza por sí mismo ningún trabajo, sino que sencillamente llama al guión adecuado para el nivel de ejecución actual. Lo que realmente suceda cuando ese guión se ejecute depende del guión. El guión es como cualquier otro programa, y puede ser tan complejo o tan simple como el autor decida.

El formato de los guiones de inicio es una de las áreas donde se puede seguir la pista desde los orígenes de un sabor Unix hasta el original Unix SysV de AT&T o los sistemas operativos BSD. Cada uno de estos sistemas, a pesar de compartir el mismo comportamiento y la misma utilización de init, tienen arquitecturas distintas para los guiones reales que init ordena ejecutar. Aunque cumplan los mismos propósitos, son bastante diferentes en los detalles.

La mayoría de los sistemas similares a Unix, incluidos los sistemas Linux, escogen entre uno u otro de los modelos de guiones de arranque de SysV o BSD y lo imitan con diversos grados de precisión. Red Hat Linux, por ejemplo, utiliza el modelo de SysV, mientras que Slackware utiliza el modelo BSD. Ya que vamos a cubrir ambas distribuciones en profundidad en los capítulos 4 y 5, explicaremos los detalles de los mecanismos de SysV y BSD allí en lugar de duplicar aquí esa información.

Como muchas actividades relacionadas con los guiones, el que un usuario en particular prefiera el modelo de guiones de inicio de SysV o el de BSD es en realidad un tema de gusto personal. El modelo SysV se puede describir como más "modular", que es lo que algunas personas prefieren, mientras que el modelo BSD es más simple, que es lo que prefieren otros. Es complicado decidir si alguno es fundamentalmente mejor que otro, así que el modelo de guiones de inicio que utilizará una distribución Linux en particular es realmente algo inmaterial, aunque probablemente refleje cuál es el sabor Unix que la distribución intenta imitar.

Versiones de las bibliotecas de sistema

Como se mencionó en el capítulo 1, son realmente las bibliotecas de sistema las que definen el carácter de un sistema operativo. Si decide emplear tiempo y esfuer-

zo, por ejemplo, podría escribir un conjunto de bibliotecas de sistema que ofrecieran un clon de Windows, ejecutándose directamente sobre un núcleo Unix (como, por ejemplo, Linux). Así se proporcionaría una interfaz de usuario y una API completamente diferente, aunque seguiría existiendo un núcleo compatible con el estándar POSIX ejecutándose bajo la superficie. En otras palabras, mediante la modificación de las bibliotecas de sistema puede alterar completamente la naturaleza de un sistema operativo.

Se estará preguntando qué son estas bibliotecas de sistema. Las bibliotecas de sistema vuelven el desarrollo y uso de aplicaciones más sencillo ya que condensan un bloque de prestaciones dentro de, bueno, una biblioteca. Esto permite a los programas de usuario finales reutilizar funcionalidades mediante el empleo de la biblioteca en lugar de tener que interactuar con el núcleo directamente. Hay bibliotecas de sistema para la mayor parte de los aspectos del sistema operativo, incluyendo el acceso a redes, la ejecución y administración de procesos paralelos, el enlace y carga de programas, y muchos más.

Realmente, el término "bibliotecas de sistema" es un poco borroso. Bibliotecas parecidas a las que se acaban de mencionar son muy necesarias para que el sistema funcione. Sin embargo, hay bibliotecas adicionales que no son estrictamente necesarias, pero aún así pueden ser consideradas bibliotecas de sistema. El ejemplo más obvio lo ponen las bibliotecas que el sistema X Window necesita para funcionar; el sistema no necesita estas bibliotecas para funcionar, pero son necesarias para que el sistema pueda ejecutar una GUI. Otros ejemplos incluyen las bibliotecas de base de datos ldbm que utilizan muchos programas, o la biblioteca tcp_wrappers que muchos programas de red utilizan para la seguridad. Estas bibliotecas no son necesarias, pero siguen siendo bibliotecas de sistema.

En la amplia mayoría de los casos, la única posibilidad real como bibliotecas de sistema para un sistema Linux es el paquete GNU C Library, conocido como glibc. El paquete glibc proporciona la mayor parte de las bibliotecas que absolutamente imprescindibles para que el sistema funcione. glibc no incluye algunas de las bibliotecas de sistema opcionales; estas otras bibliotecas se suelen instalar individualmente, como paquetes separados.

Nota

Tradicionalmente, los sistemas Unix nombran sus bibliotecas como libfoo.o, donde "*foo*" es el nombre de la biblioteca. Según esta convención, libc hace referencia a las bibliotecas C. El paquete glibc proporciona muchas otras bibliotecas como complemento al libc.o básico, así que su nombre está pobremente escogido.

Las bibliotecas de sistema son probablemente una de las tres piezas de software más difíciles de escribir, junto con el núcleo y un compilador. Por ello, una vez que se consigue un conjunto funcional bibliotecas de sistema (o un núcleo o un

compilador), hay que pensárselo mucho antes de intentar crear uno propio. Por este motivo nadie ha escrito ningún competidor serio de glibc para los sistemas Linux, y cada distribución principal existente utiliza alguna versión de glibc como bibliotecas de sistema básicas.

Al menos, nadie ha escrito un competidor serio para las distribuciones de escritorio tradicionales o para servidores. De hecho hay varios esfuerzos enfocados hacia la producción de un conjunto de bibliotecas de sistema para otros entornos hardware, como los PDA, los dispositivos de red interactivos (*set-top network appliances*) y muchos más. Muchos de estos esfuerzos resultan bastante innovadores. PocketLinux de Transvirtual, por ejemplo, consiste en un núcleo Linux que esencialmente hace uso de una máquina virtual Java (llamada Kaffe) como bibliotecas de sistema; el proyecto uCLinux, en cambio, tiene como objetivo un sistema "Linux-en-chip", donde un núcleo Linux reducido pero funcional y una rudimentaria biblioteca de sistema puedan insertarse en una pieza de equipo.

De nuevo, sin embargo, siguen existiendo excepciones. A pesar de lo interesantes que puedan resultar estos esfuerzos, sólo demuestran la versatilidad del núcleo Linux, mientras que estamos interesados en sistemas más tradicionales como ordenadores de oficina, estaciones de trabajo o servidores basados en Linux. Para estos sistemas, glibc está donde tiene que estar.

Escritorio X Window

Ah, X. El sistema X Window puede ser uno de los componentes de software más amados y odiados de la historia. X se ha convertido en el estándar de facto (y algunos añadirían que más que de facto) de sistema de ventanas para sistemas operativos similares a Unix. X puede resultar extraño para los que se enfrentan a él por primera vez, puede parecer primitivo a observadores casuales, y puede parecer increíblemente poderoso para sus usuarios experimentados.

El sistema X Windows se hizo público por primera vez en 1984 de la mano del Laboratory for Computer Science del Massachusetts Institute of Technology, y ahora está dirigido por un consorcio de la industria y otros grupos públicos y privados. Desde 1984, se ha convertido en la GUI escogida para los sistemas Unix. Comúnmente denominado simplemente "X", tiene legiones de celosos seguidores, al mismo tiempo que enemigos acérrimos. Aún más, el programa de entretenimiento `fortune` incorporado en Red Hat Linux (y que simula una galleta de la fortuna) incluye muchas citas anti-X; "X Windows: desearás que estemos bromeando" y "X Windows: cuerda más que suficiente". Sin embargo, se ame o se deteste, todo el mundo admite que X introdujo varias características arquitectónicas importantes en el escenario de las GUI, incluida la transparencia de red y el diseño en capas para las interfaces de usuario.

Transparencia de red

Transparencia de red significa que se puede ver la salida de un programa X que se ejecuta en una máquina distinta al otro lado de la red. Esa máquina, de hecho, no

necesita que X se esté ejecutando, ni siquiera que tenga un monitor. Esta transparencia de red se estandarizó en el protocolo X, lo que permite a cualquier GUI soportar la visión de aplicaciones X sencillamente mediante la implementación del protocolo; por ejemplo, puede adquirir servidores X para Windows que le permitirán ver aplicaciones X en su máquina Windows. Esta transparencia de red es una de las razones primarias por las que X ha sobrevivido tanto como lo ha hecho; sin ella, X habría sido abandonado entre el polvo pixelado de sus competidores.

Modelo de GUI en capas

Otro de los motivos importantes para la fama de X es el modelo de GUI en capas. Básicamente, el sistema X Windows por sí mismo es sólo un marco de eventos para la gestión de ventanas y eventos de ventana. Gestionar ventanas implica seguir la pista de dónde se encuentran ubicadas en la pantalla, sus dimensiones y demás. Gestionar eventos de ventana significa enviar notificaciones cuando se pulsan botones, se redimensionan ventanas, se vuelven a pintar o dibujar y cosas así. Estos eventos de gestión de ventanas, junto con la funcionalidad de la transparencia de red, forman el protocolo X-Window. Este protocolo no sabe nada acerca de tarjetas de video, memoria, aceleración 3D, ni ningún otro aspecto que realmente represente gráficos sobre un dispositivo. Estas tareas de bajo nivel son la responsabilidad del servidor X, que es un programa que interactúa con la tarjeta de video, implementando el protocolo X.

Sin embargo, ni el protocolo X ni el servidor X tienen conocimiento de los que sucede en una ventana. Es decir, X no ofrece prestaciones para dibujar botones, cuadros de comprobación, ni ninguno de los otros componentes que son familiares a los usuarios. El protocolo X sólo administra eventos para estos componentes, y el servidor X sólo maneja el proceso de representación de esos componentes en la tarjeta de video. X tampoco ofrece ningún componente de la interfaz de usuarios como iconos, escritorios, menús de inicio y demás. El propio X está bastante desprovisto (cuando se compara con otras GUI como MacOS o Windows) y confía en otras dos capas para hacer el trabajo real: el conjunto de componentes y el gestor de ventanas. En otras palabras, el sistema X Window consta de tres partes:

- Un servidor X
- Un conjunto de componentes
- Un gestor de ventanas

Estas capas se comentarán con más detalle en el resto de esta sección.

Conjunto de componentes (*widgets*)

El conjunto de componentes (*widget set*) es una biblioteca o una especie de paleta de elementos comúnmente utilizados en las GUI. Ejemplos de componentes incluyen los botones y cuadros de comprobación mencionados en la sección anterior, al igual que listas de selección, botones de opción, y muchos otros. X no impone ninguna restricción sobre el aspecto de estos componentes, de modo que los

desarrolladores y artistas son libres para crear conjuntos de componentes con el aspecto que ellos decidan. Hay una gran cantidad de conjuntos de componentes, con una amplia variedad de aspectos. Motif (desarrollado por la Fundación de Software Abierto -*Open Software Foundation*), sin ninguna relación con el FSF) fue el primer conjunto de componentes muy popular, y se convirtió en el estándar de referencia durante muchos años. Hoy en día, hay muchos conjuntos de componentes disponibles para elegir; tantos, de hecho, que es difícil seguirle la pista a todos ellos.

Recientemente, los conjuntos de herramientas GTK+ de GNU y QT de TrollTech están tomando puestos como bibliotecas modernas, repletas de características y de código abierto. Estos conjuntos son realmente algo más que simples conjuntos de componentes, ya que también ofrecen marcos bastante amplios para construir aplicaciones. GTK+ se popularizó gracias al escritorio GNOME, y Qt gracias a KDE.

GNOME y KDE

GNOME y KDE, en cambio, son ejemplos de la última capa de una GUI X: el gestor de ventanas. Un gestor de ventanas es responsable de dibujar las decoraciones de ventanas como los bordes, las barras de título y los ubicuos botones de minimizar y maximizar, al mismo tiempo que los aspectos más agradables, como iconos, escritorios, menús, etc... que realmente caracterizan una interfaz de usuario. Hay probablemente tantos o más gestores de ventanas que conjuntos de componentes, y frecuentemente hay un gestor de ventanas para cada conjunto. Por ejemplo, además de las parejas KDE/Qt y GNOME/GTK+, el gestor de ventanas mwm se basa en Motif, mientras que los gestores fvwm y twm tienen sus propios conjuntos de componentes internos.

Nota: Gestor de ventanas frente a entorno de escritorio

Aunque describo KDE y GNOME como gestores de ventanas, realmente es una pequeña mentira piadosa. En realidad, ambos son ejemplos de juegos de software que conforman un entorno de escritorio. Los juegos incluyen varios programas distintos, de los que un gestor de ventanas sólo es uno. Normalmente, KDE y GNOME incluyen:

- Gestor de ventanas
- Programa escritorio (para colocar iconos en el fondo)
- Panel de botones (que contiene botones para iniciar programas)
- Barra de tareas o lista de ventanas (para conmutar entre aplicaciones en funcionamiento)
- Paginador de escritorios (para conmutar entre escritorios virtuales)
- Programas de utilidades (para controlar y configurar todo lo demás)

Distintos escritorios y gestores de ventanas (incluyendo no sólo a KDE y GNOME sino también a otros) tendrán algún conjunto más pequeño o más amplio de componentes de esta lista. Para liberar realmente su productividad cuando utilice su escritorio, debería aprender qué características se encuentran disponibles en su entorno. A lo largo del resto de este libro, nos referiremos a estos juegos como a entornos de escritorio.

En esencia, el gestor de ventanas se encarga de mover, redimensionar y cerrar las ventanas, mientras que el conjunto de componentes se encarga del aspecto y el comportamiento de las ventanas. Mientras tanto, X ofrece el marco en que funciona todo esto. Este modelo orientado a componentes de la GUI da a los desarrolladores y usuarios un asombroso, e incluso desalentador, grado de elección con respecto a las GUI. Algunas de estas GUI son muy simples y livianas, como twm (abreviatura para *"Tom's window manager"*, el gestor de ventanas de Tom). Otros, como KDE y GNOME, son más que gestores de ventanas y quedan descritos mejor como entornos de escritorio.

Escoger un servidor X, conjunto de componentes, y gestor de ventanas

Puesto que en un entorno X Window hay tres "capas", básicamente existen tres decisiones que necesita tomar atendiendo a lo siguiente:

- Qué servidor X usar
- Qué conjunto de componentes usar
- Qué gestor de ventanas usar

Para los sistemas Linux y la mayoría de los sistemas BSD, el paquete XFree86 es el estándar de facto como servidor X y conjunto de bibliotecas, así que esta es una elección sencilla, de modo que sólo queda determinar el conjunto de componentes y el gestor de ventanas. No obstante, ya que el conjunto de componentes y el gestor de ventanas suelen estar íntimamente relacionados, sólo queda tomar una decisión. En otras palabras, la única elección que realmente tiene que realizar es qué gestor de ventanas o escritorio quiere utilizar.

La elección de gestor de ventanas depende en gran medida de lo que quiera hacer con el sistema. (¿De qué me suena a mí esta frase?) Si está leyendo este libro, probablemente esté más interesado en utilizarlo como escritorio, servidor o estación de trabajo. En estos casos, sus otras alternativas son Windows o Macintosh, así que la competición se promete dura. Si esto es lo que está buscando, sus mejores opciones son KDE y GNOME.

KDE y GNOME son ambos entornos muy completos. No sólo proporcionan un gestor de ventanas, sino también diversas aplicaciones como paneles de iconos (para iniciar aplicaciones), barras de tareas y otras características que conseguirán

que la mayoría de los usuarios de Windows se sientan como en casa. KDE y GNOME también disponen de características que sus competidores no pueden alcanzar (incluso Windows y Macintosh) como la habilidad de soportar "temas" o "pieles" particularizadas que permiten al usuario personalizar el escritorio hasta el nivel del conjunto de componentes. Las figuras 3.2 y 3.3 muestran el mismo escritorio KDE ejecutando dos temas completamente diferentes. La figura 3.2 muestra el tema "Acqua GRAPHITE" de Alessandro Rossini, mientras que la figura 3.3 muestra el tema de Martin Doege, "LCARS ACCESS 411". Ambos temas pueden conseguirse en el sitio web kde.themes.org, o a través de su buscador web preferido.

KDE y GNOME generalmente se rezagan con respecto a Windows sólo en algunos aspectos de la infraestructura. Por ejemplo, Windows posee un modelo de objetos componentes (*component object model*, COM) más extenso, presente y robusto para incluir prestaciones en las aplicaciones y comunicación entre las mismas, que es hacia donde KDE y GNOME están comenzando a dirigirse. Esto resulta particularmente evidente en el campo de las aplicaciones de productividad en oficinas, donde COM se utiliza de un modo muy amplio. Sin embargo, tanto KDE como GNOME tienen en camino esfuerzos orientados a la creación de un armazón de componentes incrustados, y están ganando terreno rápidamente. La versión 2.1 de KDE, por ejemplo, ofrece algunas características que incluso Windows no tiene. De manera adicional, consorcios de la industria se han alineado recientemente detrás de KDE y GNOME, incluyendo apoyo proveniente de todos los fabricantes comerciales de Unix principales. Este apoyo podría acelerar el progreso de estos entornos incluso más.

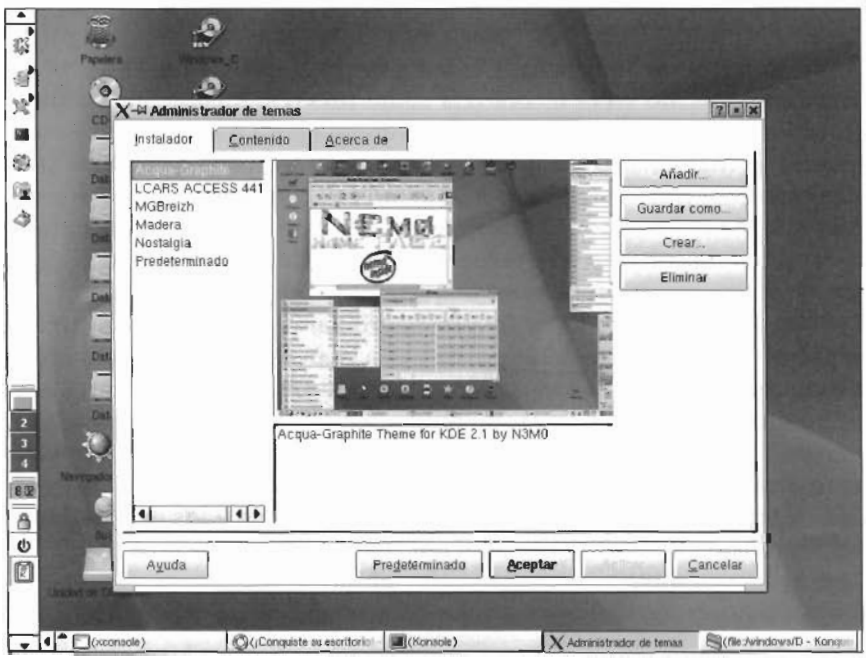


Figura 3.2. Ejemplo de tema de KDE: Acqua GRAPHITE.

Figura 3.3. Ejemplo de tema de KDE: LCARS ACCESS 411.

Software de aplicación

Hasta ahora, he hablado mucho acerca del propio sistema. Eso está bien si es un entusiasta interesado en la tecnología como fin en sí misma. Sin embargo, el resto de nosotros estamos interesados en la tecnología para nuestros fines propios : ¿Qué nos puede ayudar a hacer? (Espero que se haya percatado de este tema recurrente.) A este respecto, un sistema Linux es sólo tan útil como lo sea el software de productividad que tiene instalado. El objetivo real de una distribución Linux es ofrecer tanta cantidad de software útil como sea posible; todo lo visto hasta ahora sólo nos permite instalar y hacer funcionar lo que realmente nos importa.

Por eso, ¿qué clases de software puede utilizar realmente con Linux? La respuesta es: Todas las clases. Sin embargo, hay una gran cantidad de software hay afuera, y no todo resulta útil para todos. Al igual que no querrá dejar fuera un paquete de software que los usuarios quieran, tampoco querrá añadir demasiados que no vayan a necesitar. Por eso, para determinar qué clase de software instalar, en primer lugar tendrá que determinar que clase de sistema va a instalar.

Generalmente, hay tres modos en que la gente utiliza los ordenadores:

- Escritorios (u ordenadores personales)
- Servidores
- Estaciones de trabajo

Un escritorio generalmente lo utiliza gente no muy técnica que sólo quiere realizar su trabajo. Se encargan de los servidores personas o empresas que necesitan ofrecerse servicios entre sí o al público. Una estación de trabajo es un híbrido: es parte servidor y parte escritorio. Una estación de trabajo es generalmente un escritorio que suele ejecutar aplicaciones más pesadas (como aplicaciones de diseño para ingenierías o científicas) y en ocasiones unos pocos servicios (como servidores web de poco volumen). Microsoft Windows ME y el OS de Macintosh son buenos sistemas operativos de escritorio; Windows NT es un buen sistema operativo para estaciones de trabajo; Solaris y Linux son buenos sistemas operativos para estaciones de trabajo y servidores. (Estos ejemplos sólo son generalizaciones, por supuesto; no hay nada que diga que MacOS no puede ser el sistema operativo de una estación de trabajo.)

¿Por qué Linux no es un buen sistema operativo para escritorio? Bueno, hay gente que dice que lo es. No obstante, hay una complejidad mucho mayor relacionada con un sistema Linux que con el típico sistema operativo para escritorio, y los usuarios de sistemas operativos de escritorio generalmente evitan la complejidad en todas sus formas. La prueba está en la mezcla: Linux ha explotado en el campo de los servidores y las estaciones de trabajo, así que si no lo ha hecho en los escritorios debe de haber una razón. Sin embargo, es importante darse cuenta de la diferencia entre que no lo haya hecho y que no pueda: los sistemas Linux no han experimentado una amplia penetración en los escritorios aún; que no quiere decir que no puedan. KDE y GNOME tiene mucho que decir al respecto, y sólo el tiempo dirá lo que sucede.

¿Por qué KDE y GNOME son tan fuertes? Bien, porque al contrario de la creencia popular, hay software para los sistemas Linux. Desde completos conjuntos de software para oficinas a software para cámaras digitales, pasando por editores de imágenes, navegadores web y clientes de correo electrónico, todo está ahí, en el escritorio. Desde servidores web a bases de datos, pasando por directorios LDAP o software para compartir archivos en red, todo está ahí, en el servidor. Desde programas informáticos científicos y de simulaciones a aplicaciones de ingeniería, pasando por el desarrollo de software, todo está ahí, en las estaciones de trabajo profesionales. El problema principal es que este software puede ser complicado de localizar e instalar; a veces es necesario un desarrollador de software sólo para conseguir que un programa funcione. Bueno, un desarrollador de software o alguien que esté leyendo este libro, eso quería decir.

Base Estándar de Linux

Probablemente, llegados a este punto, vaya entendiendo que hay una gran cantidad de variables que considerar en la creación de una distribución Linux. Cada distribución es independiente, y el hecho de que dos distribuciones diferentes se basen en el mismo núcleo Linux no implica en absoluto que sean compatibles entre sí. De hecho, frecuentemente no lo son. El software que se compila para una distri-

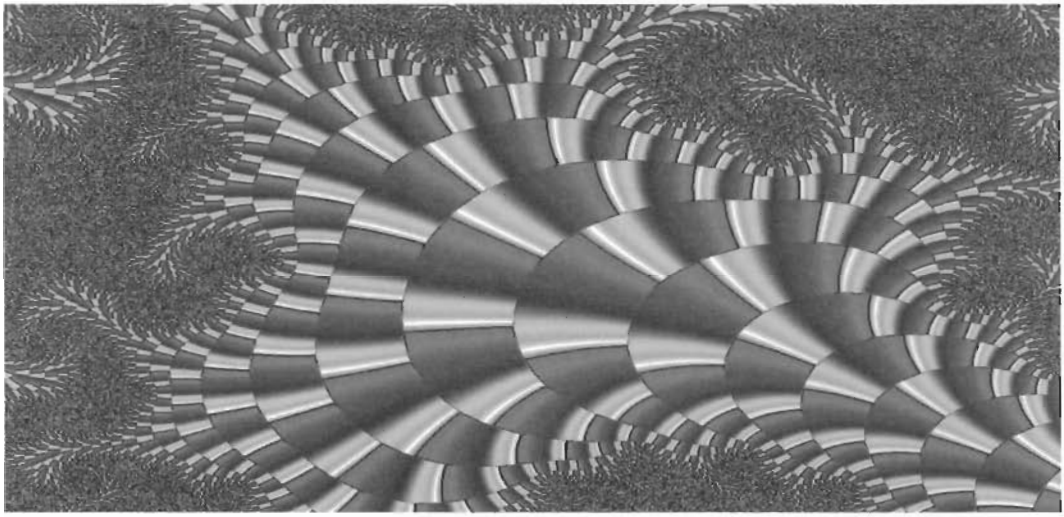
bución puede no funcionar sobre otra, y esto incrementa el trabajo de los desarrolladores de software, que acabarán teniendo que adaptar su software para múltiples distribuciones y mantener un seguimiento para todas esas adaptaciones.

Para enfrentarse a este hecho, se creó el proyecto de la Base Estándar de Linux (*Linux Standards Base*, LSB). Este proyecto es una colaboración entre empresas y desarrolladores individuales relacionados con sistemas Linux y otras actividades de software de código abierto o libre. EL objetivo de la organización es establecer un conjunto básico de software vital que los fabricantes de distribuciones puedan incluir en sus productos (en esencia, una especie de "minidistribución" que los fabricantes pueden usar como base de sus propios sistemas). Mediante la estandarización de acuerdo con la LSB, las diversas distribuciones pueden mejorar la interoperatibilidad y compatibilidad entre sistemas. En el momento de la elaboración de este libro, la LSB sólo hizo pública recientemente una especificación 1.0.0 formal, y por ello ningún fabricante de distribuciones importante la ha apoyado aún. Sin embargo, la LSB (y la organización que la subvenciona, el Grupo para los Estándares Libres -*Free Standards Group*) puede convertirse en una influencia importante sobre los sistemas Linux en el futuro. El sitio web de la LSB se puede encontrar en www.linuxbase.org.

Resumen

En este capítulo, ha leído acerca de los elementos básicos de una distribución. Ha aprendido que a pesar de lo diferentes que sean entre sí diversas distribuciones, al final todas deben de resolver el mismo conjunto de problemas, como la decisión sobre qué software incluir y cómo estructurar el sistema de archivos. Una vez que ha comprendido esto, la tarea de dominar una nueva distribución será tan simple (¡o tan complicada!) como entender de qué modo la distribución da respuesta a cada uno de estos problemas.

En los tres capítulos siguientes, leerá acerca de tres distribuciones reales. Estos capítulos comentan las distribuciones en detalle y las diseccionan para mostrar el modo en que cada una de ellas resuelve estos mismos problemas básicos. Tras la lectura de estos capítulos, poseerá un amplio conjunto de detalles de uso práctico para darle cuerpo a la comprensión general que consiguió con este capítulo



4
Red Hat Linux

Uno de los distribuidores comerciales de Linux con más éxito es Red Hat, Inc., creadores de la distribución Red Hat Linux que le da nombre. En este capítulo, leerá acerca de la versión 7.3 de Red Hat Linux y se centrará en las características que lo convierten en algo único y útil. Ya que Red Hat Linux es una de las distribuciones más populares y mejor conocidas, es un buen punto de partida para aprender acerca de los sistemas Linux en general.

Este capítulo es el primero de los tres capítulos centrados en las distribuciones que forman la segunda parte de este libro. El objetivo de estos capítulos sobre distribuciones de ejemplo es proporcionar tres perspectivas distintas sobre los sistemas Linux, ofrecer una visión perspicaz sobre su construcción e ilustrar los conceptos generales del diseño de distribuciones. Más importante aún, examinar estas distribuciones de ejemplo le armará con una más amplia base de experiencias sobre la que apoyarse cuando trabaje con cualquier otra distribución, presente o futura.

Antecedentes y filosofía de Red Hat

Bob Young y Marc Ewing fundaron Red Hat, Inc. en 1994. Estaban interesados en el software de código abierto y pensaron que sería un gran modelo de negocio. Su actividad principal al principio fue la producción y venta de Red Hat Linux, su producto bandera. Desde el comienzo de Red Hat, la empresa ha expandido su campo de acción a varios mercados adicionales, incluidos los sistemas embarcados

y servicios profesionales. Como empresa, sin embargo, Red Hat permanece alineada con y dedicada al modelo de código abierto. Casi todo su propio software se lanza bajo una licencia de código abierto, y financian diversos esfuerzos de desarrollo de código abierto que no son necesariamente esenciales para la distribución Red Hat Linux.

Nota

Véase el capítulo 2 acerca del modelo de código abierto.

Sin embargo, Red Hat Software es un negocio, así que tienen que conseguir dinero. Como Red Hat Linux es gratuito en sí mismo, Red Hat consigue su beneficio al añadir valor a la distribución a través de contratos de soporte, subscripciones para acceso prioritario y muchos otros métodos. De hecho, estas otras actividades son la principal fuente de financiación de Red Hat, y su distribución Linux puede ser vista simplemente como una plataforma que posibilita estas otras fuentes de ingreso.

A pesar de que sea o no su fuente principal de dinero principal, la distribución Linux de Red Hat aún tiene que tener calidad, o de otro modo no triunfará. Para garantizar la mejor calidad global, Red Hat considera su distribución como un sistema holístico que debe de tener todas las características posibles y estar tan libre de errores como se pueda en cuanto agregado, en lugar de ser el más avanzado en términos de cada componente individual (como el núcleo o el compilador C del sistema). En consecuencia, Red Hat toma un enfoque más conservador en cuanto a su software que muchas otras organizaciones de software libre y código abierto. Red Hat frecuentemente incorpora versiones de software y parches que no son técnicamente "estables" pero que Red Hat considera listas para producción. A la inversa, con la misma frecuencia Red Hat elige no incluir software que otros podrían decir que es estable si no cumple los estándares propios de calidad de Red Hat. En otras palabras, para mejorar la calidad y funcionalidad de la distribución Red Hat, la empresa acepta diversos compromisos con ese objetivo en mente.

Núcleo Linux

Red Hat Linux 7.3 incluye un núcleo basado en Linux 2.4.18 y en general está dirigida al uso de la serie 2.4 de núcleos. Cuando la siguiente serie se haga pública (que será probablemente la 2.6 o la 3.0), la serie Red Hat 7.x podrá o no trabajar con los nuevos núcleos; Red Hat generalmente diseña sus distribuciones para que trabajen con una serie específica de núcleos. Aunque puedan luchar con el objetivo de la compatibilidad hacia delante, el objetivo último de Red Hat es conseguir un sistema global robusto y funcional, así que en ocasiones se deja de lado esa compatibilidad hacia delante cuando llegan al convencimiento de que los cambios necesa-

rios reducirían la calidad de su distribución. (Red Hat tiene esta actitud con respecto a todas las partes de su sistema, no únicamente al núcleo, como podrá ver cuando continúe leyendo.)

Como Red Hat está concentrado en producir una distribución de alta calidad, raras veces incluyen un núcleo Linux tal cual en su sistema operativo. En su lugar, Red Hat somete el núcleo a una serie de extensas pruebas para garantizar la calidad, y suelen aplicar particularizaciones y corrección de errores dirigidas a mejorar la calidad de la distribución o a añadir alguna funcionalidad que consideran útil. Red Hat no está solo en esta práctica; la mayor parte de los fabricantes de distribuciones comerciales siguen prácticas similares, a pesar de que algunas otras distribuciones (como Slackware, que se comentará en el capítulo 5) se centren en los núcleos estándar.

Por supuesto, Red Hat envía cualquier cambio que realicen al núcleo Linux de vuelta a la comunidad de desarrolladores del núcleo. De hecho, Red Hat emplea desarrolladores que son colaboradores regulares y cruciales del núcleo Linux estándar. Generalmente, cualquier cambio que Red Hat realiza será finalmente incluido en las fuentes del núcleo estándar, pero en ocasiones no será así, lo que hace despertar temores acerca de una "bifurcación del núcleo" o incompatibilidades. Hasta la fecha, Red Hat nunca ha incluido un núcleo que suponga un cambio verdaderamente significativo con respecto al núcleo estándar.

Formato de empaquetado

Red Hat Linux utiliza el formato Administrador de Paquetes Red Hat (*Red Hat Package Manager*, RPM) como su administrador de paquetes. RPM se desarrolló explícitamente para Red Hat Linux. Esta sección comentará el formato RPM en detalle, describiendo en primer lugar alguna de las propiedades generales del formato y cubriendo después las características más importantes en detalle. Una vez que haya leído esta sección, dispondrá de una comprensión que podrá utilizar para administrar software en un sistema basado en RPM y sabrá dónde buscar para descubrir cómo realizar las tareas más esotéricas.

RPM fue uno de las primeras herramientas de administración de paquetes para sistemas Linux, pero actualmente se encuentra en la versión 4, lo que quiere decir que ha sido revisada significativamente y puesta a punto tres veces desde la versión original. RPM es consecuentemente una herramienta de administración de paquetes relativamente madura y se puede comparar favorablemente con la mayoría de las herramientas comerciales de administración de paquetes. De hecho, es un producto tan sólido que incluso fabricantes comerciales de Unix están comenzando a incluir instalaciones suyas a medida. El único rival real que se equipara o incluso supera a RPM es el sistema de empaquetado de Debian.

RPM es un ejemplo bastante típico de herramienta de administración de paquetes para sistemas Unix, aunque sobrepasa de lejos a la mayoría con sus prestaciones actuales. Este tipo de utilidades típicamente le permiten instalar y desinstalar pa-

quetes de software. RPM proporciona grandes prestaciones para estas tareas básicas, pero también va más allá ofreciendo una semántica de validación de dependencias y actualización, al mismo tiempo que una manera de generar listas con los paquetes listados e incluso realizar preguntas a los paquetes (tanto instalados como no instalados) en busca de documentación. Adicionalmente, RPM resulta funcional con el soporte de archivos RPM fuente, que son paquetes que contienen el código fuente para un paquete de software particular. Los usuarios pueden utilizar un RPM fuente para construir el software con particularizaciones diferentes de las que se dan por defecto en un determinado paquete o para construir el software para una arquitectura o plataforma no soportada por el paquete.

Puede pensar en RPM en cierto sentido como en una especie de base de datos para información sobre software. Las bases de datos relacionales (como aquellas utilizadas en sitios web u otras aplicaciones) almacenan sus datos de tal modo que pueden ser interrogadas o actualizadas de acuerdo con una amplia variedad de modos. De manera similar, el formato RPM permite a los administradores y usuarios preguntar a los paquetes instalados en busca de una gran variedad de información y "añadir" o "eliminar" aplicaciones mediante su instalación o desinstalación. También, del mismo modo en que las bases de datos relacionales fuerzan transacciones y operaciones atómicas, RPM obliga a comprobar los requisitos de dependencia del software e intenta garantizar que las operaciones sean atómicas.

Nota

Una operación atómica es una que no puede ser interrumpida y que se garantiza que se completará con éxito o no se realizará en absoluto. Una operación atómica no puede realizarse parcialmente y dejar todo el sistema revuelto.

Resumen técnico

Esta sección cubre algunos de los detalles básicos de un paquete RPM. En particular, se comentarán las convenciones adoptadas para dar nombre a los archivos RPM y las propiedades del propio formato de archivos.

Convenciones de nombrado de archivos

Por convención, los nombres de archivo RPM tienen este formato:

```
paquete-versión-construcción.arq.rpm
```

donde "paquete" es el nombre del software, "versión" es la versión del software hecho público, "construcción" es la versión de un paquete RPM en particular, y "arq" es la arquitectura para la que se compiló el RPM.

Por ejemplo, un archivo RPM para glibc v2.2.5 incluido en Red Hat Linux 7.3 y construido para sistemas basados en Intel podría tener el nombre de archivo glibc-2.25-.i386.rpm. La versión de este archivo actualmente incluida en Red Hat 7.3 es

glibc-2.2.5-34.i386.rpm; en este caso, "-34" indica que es la versión trigésimo cuarta del RPM que Red Hat creó antes de lanzar Red Hat 7.3. (Generalmente, estas miniversiones incluyen cambios menores en la estructura del paquete u opciones de configuración, y ocasionalmente correcciones menores de errores.)

Arquitecturas

Como se mencionó en la sección anterior, los RPM se construyen para arquitecturas específicas. Normalmente, una arquitectura es simplemente un tipo de microprocesador. La mayoría de las instalaciones de Linux se realizan en sistemas con microprocesadores Intel de la familia i386, pero Linux se puede ejecutar en muchos otros procesadores, y también podrá encontrar RPM para estas plataformas. La tabla 4.1 da información sobre algunas arquitecturas comunes y el nombre de esa representación en un nombre de archivo RPM.

Tabla 4.1. Arquitecturas RPM.

Arquitectura	Fabricante	Nombre en RPM	Comentario
familia Intel 386	Intel	i386	Formato RPM más común.
Intel Pentium	Intel	i586	Utiliza optimizaciones de rendimiento especificas para Pentium.
Intel Pentium Pro	Intel	i686	Utiliza optimizaciones para Pentium Pro, Pentium III o Pentium IV.
Alpha	Compaq	alpha	Procesador RISC de alto rendimiento de Compaq. ¡Es una arquitectura real y no indica una versión de prueba del software!
PowerPC	IBM, Motorola, Apple	ppc	Arquitectura PowerPC de IBM utilizada en los Apple Macintosh y algunas estaciones de trabajo de IBM.
SPARC	Sun Microsystems	sparc	Procesador utilizado por Sun en estaciones de trabajo y servidores.
Genérica	No aplicable	noarch	El RPM no contiene ningún archivo binario específico para un procesador (por ejemplo, podría consistir enteramente en guiones de shell).

(Continúa)

Arquitectura	Fabricante	Nombre en RPM	Comentario
Código fuente	No aplicable	src	Sólo contiene código fuente. Cuando se compile para una arquitectura específica, la salida será un RPM binario para esa plataforma.

Formato del archivo del paquete

El formato de archivo RPM se basa en el de cpio; esto es, un archivo rpm es simplemente un archivo cpio con algunos bytes adicionales de cabecera que lo preceden. El paquete RPM en sí mismo viene con una herramienta (rpm2cpio) que puede convertir realmente un archivo RPM en el archivo cpio subyacente. El comando `cpio` en un sistema Unix es similar a `tar`, pero tiene un uso ligeramente diferente.

Sin embrago, RPM es más que sencillamente un elaborado formato de almacenamiento de archivos. Los archivos RPM (a veces llamados paquetes) incluyen información adicional sobre las dependencias que tiene el software. RPM también permite que se ejecuten guiones cuando se instala o desinstala el paquete, y esos guiones pueden realizar trabajo adicional que RPM no puede hacer automáticamente por sí mismo. Una instalación RPM forzará estas dependencias y rechazará intentos del administrador de instalar un archivo si las dependencias no se cumplen (a no ser que el usuario anule explícitamente este comportamiento).

Nota: Los comandos `cpio` y `tar`

Dos formatos comunes para el empaquetado y distribución de archivos en sistemas similares a Unix son cpio y tar. Estos formatos reciben su nombre de los programas utilizados para crearlos y manipularlos. El comando `tar` es la abreviatura de "*tape archive*" (archivo de cinta) y se usaba originalmente para mover datos hacia y desde cintas magnéticas, normalmente con el propósito de obtener y restaurar copias de seguridad. Sin embargo, la herramienta también puede utilizarse para crear y manipular archivos que agrupen archivos ordinarios. El comando `cpio` es muy similar a `tar`, y su nombre significa "*I/O (input/output) copy*" (copia entrada/salida).

Estos dos comandos se utilizan para crear archivos que contienen otros archivos. Son muy similares al formato de archivo ZIP comúnmente utilizado en otros sistemas operativos como Windows y OS/2. No obstante, al contrario que el formato ZIP estos comandos no proporcionan ninguna compresión de los datos por sí mismos, sino que son simplemente "contenedores" de otros archivos. Típicamente, uno de estos archivos se comprime mediante un

comando como el programa Zip de GNU (`gzip`). Por ejemplo, un archivo tar comprimido con `gzip` se convertirá en el omnipresente archivo .tar.gz.
Estos formatos de archivo difieren de un formato de empaquetado más sofisticado como RPM. Donde un archivo tar o cpio simplemente contiene a otros archivos para su "transporte", un archivo RPM añade a los mismos archivos información adicional (como dependencias de otros archivos no incluidos en el archivo o guiones de instalación) necesaria para la correcta instalación o utilización de los archivos.

Uso de RPM

En esta sección se comentan los mecanismos básicos del uso de RPM para administrar instalaciones de software. La documentación de Red Hat sobre RPM es bastante buena, por eso, en lugar de reproducir esa información, esta sección y las siguientes tratarán únicamente acerca de los conceptos básicos. Los lectores que necesiten información más detallada deberían consultar la página del manual de RPM (mediante el comando `man rpm`) o la excelente documentación existente en el sitio web de Red Hat (www.redhat.com/docs/books/max-rpm/index.html).

Modos del programa

El comando para administrar paquetes RPM es `rpm`, así de simple. Este sencillo programa dispone de los siguientes cinco modos de operación para realizar sus diversas tareas:

- Preguntar
- Instalar
- Actualizar
- Verificar
- Eliminar
- Reconstruir

Cada uno de estos modos tiene opciones adicionales que permiten a los usuarios realizar tareas específicas. Una vez más, si quiere la documentación completa, consulte la información disponible procedente de Red Hat. Las siguientes secciones dan detalles sobre cómo llevar a cabo varias tareas útiles y comunes.

Permisos de acceso

La base de datos RPM es una herramienta para administrar el software instalado en el sistema. Obviamente, si un sistema se comparte entre muchos usuarios, sólo

un administrador debería poder instalar software. Sin embargo, los usuarios aún podrían desear ser capaces de comprobar qué paquetes están instalados; por este motivo, la base de datos RPM se instala normalmente con permiso de lectura para todo el mundo y permiso de escritura sólo para superusuarios (usuarios *root*). Por ello, como regla general, los usuarios normales podrán realizar cualquier pregunta, pero sólo el superusuario podrá ejecutar cualquier comando que modifique el contenido del disco, como una instalación o eliminación.

Comprobaciones de dependencia

Muchos de los comandos descritos en esta sección confían en comprobaciones de dependencia. Esto es, RPM determinará si puede o no realizar una operación según si la operación puede provocar que se destruya el software o no se cumplan las dependencias. Por conveniencia, RPM también incluirá todos los nombres de paquete o archivo que reciba en esta comprobación de dependencia. Por ejemplo, el comando `rpm -i foo.i386.rpm bar.i386.rpm` servirá para instalar ambos paquetes RPM al mismo tiempo. Además de ahorrarle al usuario el esfuerzo de introducir dos comandos `rpm`, esto permite a los usuarios instalar al mismo tiempo múltiples RPM que podrían ser mutuamente dependientes. La mayoría de las operaciones RPM manifiestan este comportamiento.

Nota: Comprensión de las dependencias

Un paquete tiene una dependencia de otro paquete o archivo si necesita de la presencia de ese paquete o archivo para funcionar adecuadamente. Estas dependencias pueden tomar cualquier forma. Un programa puede necesitar una biblioteca compartida en particular para funcionar, o podría necesitar que otro programa se instale en primer lugar. (Este programa podría utilizarse en la configuración o en el uso del nuevo programa.) Alternativamente, un paquete de archivos de documentación podría necesitar la presencia de un programa lector para ese tipo de archivos.

Como un ejemplo concreto, el gestor de ventanas y sistema de escritorio de KDE necesita que se haya instalado Xfree86, al igual que la biblioteca de herramientas de ventanas Qt. Del mismo modo, el servidor FTP de la Washington University (wuftpd) necesita que un programa inetd (como xinetd) esté presente. Seguir la pista de todas estas dependencias puede convertirse en una carga, y por eso el sistema RPM (y otros formatos de empaquetado) se crearon para automatizar estas tareas.

Información sobre paquetes no instalados

RPM es un formato de distribución tanto como un formato de administración de paquetes. Esto es, un objetivo importante de RPM es proporcionar a los distribuido-

res de software una manera canónica y estandarizada de empaquetar sus aplicaciones de tal modo que los usuarios finales puedan instalar fácilmente esas aplicaciones. Con esta finalidad, el formato RPM y el programa rpm proporcionan al usuario la capacidad de preguntar a los paquetes RPM no instalados para obtener información sobre el software. Esto permite a los usuarios descubrir qué hará el RPM y que instalará antes de instalarlo realmente.

En general la forma de un comando de pregunta RPM es esta:

```
rpm -q<opciones> <objetivo>
```

<objetivo> puede ser tanto el nombre de un paquete instalado como la ruta de un archivo RPM, y <opciones> son las letras adicionales que especifican el tipo de información que se debe obtener y si el paquete deseado ya está instalado.

Para preguntar a un paquete que no se ha instalado aún, la forma es

```
rpm -qp<opciones> <nombre de archivo>
```

<opciones> debe indicar en este caso qué información se debe recuperar, y <nombre de archivo> debe ser la ruta hacia el archivo RPM al que se pregunta. Existen numerosas opciones para obtener información, pero estas cuatro son particularmente útiles:

- Información general
- Lista de archivos
- Requisitos
- Capacidades proporcionadas

La tabla 4.2 muestra la lista de los comandos RPM necesarios para realizar estos tipos de preguntas y describe qué detalles proporciona cada uno.

Tabla 4.2. Información sobre paquetes RPM.

Tipo de pregunta	Comando RPM	Información proporcionada
Información general	`rpm -qpi`	Información general sobre el paquete, como tamaño en bytes y una descripción textual del paquete.
Lista de archivos	`rpm -qpl`	Una lista de todos los archivos que el paquete instalará.
Requisitos	`rpm -qpR`	Una lista de todas las capacidades (como bibliotecas compartidas) necesitadas por el software.
Capacidades proporcionadas	`rpm -qp --provides`	Una lista de todas las prestaciones proporcionadas por el software.

Nota

Cada uno de estos comandos debe estar seguido por el nombre del archivo del paquete RPM sobre el que se pregunta (por ejemplo, `rpm -qpi glibc-2.2.5-34.i386.rpm`).

Entre las preguntas adicionales que puede realizar sobre paquetes no instalados se incluye generar listas de archivos de configuración y documentación, listas de los guiones utilizados por el paquete y demás. Algunas de estas preguntas RPM más avanzadas son realmente bastante potentes, pero pueden convertirse en algo un poco críptico y misterioso. Los lectores interesados deberían consultar la documentación de Red Hat sobre RPM.

Instalación de paquetes

Instalar paquetes mediante RPM es bastante sencillo. El comando que se utiliza es sencillamente:

```
rpm -i <nombre de archivo>
```

Dos opciones muy útiles son `--force` y `--nodeps`, que respectivamente obligan a que el paquete se instale incluso si entra en conflicto con archivos ya existentes e indican a RPM que ignore las comprobaciones de dependencia cuando instale el paquete.

Aviso

Utilice las opciones `--force` y `--nodeps` con mucha precaución, ya que lo que hacen es anular el comportamiento de RPM por defecto.

Los usuarios pueden aprovechar las opciones adicionales para indicar a RPM que particularice la instalación de un paquete, pero generalmente el proceso de las instalaciones es tan simple como ejecutar los comandos anteriores.

RPM sigue la siguiente rutina cuando realiza la instalación de un paquete:

1. Extrae una lista de archivos a partir del paquete y la compara con la base de datos de archivos instalados en busca de conflictos.
2. Comprueba las dependencias que se indican en el paquete para asegurarse de que se cumplen. Si todo va bien, entonces RPM ejecuta cualquier guión de preinstalación identificado en el paquete.
3. Los archivos se extraen del archivo y se colocan en los sitios apropiados del disco.
4. Finalmente, RPM ejecuta cualquier guión de postinstalación identificado en el paquete.

Una vez que se han completado estos pasos, el paquete queda instalado. Sin embargo, a veces un determinado paquete que ya se ha instalado puede necesitar que se le informe de que un paquete relacionado está instalado. Por ejemplo, el paquete xinetd puede necesitar que se le informe cada vez que cualquier aplicación que utilice xinetd se instale. En casos como este, RPM permite a un paquete ha designado guiones "gatillo" que se ejecuten cuando unas determinadas condiciones se cumplan durante una instalación posterior de paquetes. Ejecutar cualquier guión gatillo relevante es el último paso en la instalación de un paquete RPM. En la práctica, proporcionalmente pocos paquetes RPM utilizan realmente gatillos, pero esta funcionalidad está disponible.

Actualización de paquetes

Si hay algo que sea cierto con respecto al software, es que nunca es algo que quede terminado. Cuando una nueva versión del software se hace pública, los usuarios probablemente querrán actualizar sus copias. RPM permite a los usuarios actualizar los paquetes instalados en su sistema con una versión posterior.

El comando para una actualización RPM es

```
rpm -u <nombre de archivo>
```

El proceso de actualización de un RPM es parecido a su instalación, sólo que RPM comprueba primero que el software ya esté instalado. (RPM anulará el proceso si el software no se encuentra presente.) La mayoría de los comandos disponibles durante la instalación de un RPM también funcionará para una actualización. Utilizar una actualización RPM esencialmente ahorra al usuario el tener que borrar manualmente la instalación previa e instalar a continuación la nueva.

Nota: ¡Atención con las mayúsculas!

La sintaxis para una actualización RPM es `rpm -U <nombre de archivo>`; sin embargo, ¡versiones antiguas de RPM utilizaban una "u" minúscula con el significado de desinstalar, *uninstall*! Las versiones actuales de RPM utilizan la opción `-e` (por *erase*, borrar) en lugar de `-u`, pero debería tener cuidado a pesar de todo, especialmente cuando utilice sistemas Red Hat Linux más antiguos.

Información sobre paquetes instalados

Una de las más potentes características de RPM es su capacidad para preguntar al sistema acerca de los paquetes instalados. Esto permite a los usuarios comprobar con facilidad si una pieza de software está instalada (y si lo está, dónde se encuen-

tra ubicada), en lugar de tener que seguirle el rastro en el sistema para localizarla del modo complicado. También permite a los administradores mantener el sistema bajo control, ya que pueden generar una lista de todo el software instalado. Como la base de datos de RPM también sigue la pista de la información adicional descrita anteriormente en este capítulo dentro de la sección "Información sobre paquetes no instalados", RPM proporciona acceso a la documentación y a la información de dependencia de los paquetes instalados.

Preguntar a un paquete instalado resulta casi idéntico a preguntar a un paquete que aún no se ha instalado. La única diferencia es que en lugar de ejecutar el comando:

```
rpm -qp<opciones> <nombre de archivo>
```

los usuarios ejecutan el comando:

```
rpm -q<opciones> <paquete>
```

y substituyen el nombre del archivo de un paquete no instalado por el nombre de un paquete instalado en su lugar. Por lo demás, toda las opciones de preguntas que se muestran en la tabla 4.2 se pueden continuar utilizando.

Una variación bastante notable del comando de preguntas sobre paquetes es la opción de preguntar a todos los paquetes. El comando:

```
rpm -qa
```

permite a los usuarios realizar una pregunta sobre todos lo paquetes. Normalmente, se utiliza para conseguir una lista de todos los paquetes instalados, pero los usuarios también pueden añadir las opciones de pregunta de la tabla 4.2. Aunque, lo advertimos: comandos como `rpm -qai` generarán una gran cantidad de información de salida.

Verificación de los paquetes instalados

A lo largo del uso normal de un sistema, los archivos cambian. Esto puede resultar tan inocente como sencillamente cambiar un archivo de configuración en el directorio /etc o tan problemático como que un archivo se corrompa debido a un fallo de energía del sistema. Para ayudar en la detección de estos cambios, RPM permite a los usuarios y administradores verificar la integridad de un paquete instalado. El comando:

```
rpm -V <paquete>
```

o

```
rpm -Va
```

comprobará todos los paquetes instalados. Después RPM realiza el cálculo de la suma de comprobación MD5 de todos los archivos del disco y los compara con las

Una vez que se han completado estos pasos, el paquete queda instalado. Sin embargo, a veces un determinado paquete que ya se ha instalado puede necesitar que se le informe de que un paquete relacionado está instalado. Por ejemplo, el paquete xinetd puede necesitar que se le informe cada vez que cualquier aplicación que utilice xinetd se instale. En casos como este, RPM permite a un paquete ha designado guiones "gatillo" que se ejecuten cuando unas determinadas condiciones se cumplan durante una instalación posterior de paquetes. Ejecutar cualquier guión gatillo relevante es el último paso en la instalación de un paquete RPM. En la práctica, proporcionalmente pocos paquetes RPM utilizan realmente gatillos, pero esta funcionalidad está disponible.

Actualización de paquetes

Si hay algo que sea cierto con respecto al software, es que nunca es algo que quede terminado. Cuando una nueva versión del software se hace pública, los usuarios probablemente querrán actualizar sus copias. RPM permite a los usuarios actualizar los paquetes instalados en su sistema con una versión posterior.

El comando para una actualización RPM es

```
rpm -u <nombre de archivo>
```

El proceso de actualización de un RPM es parecido a su instalación, sólo que RPM comprueba primero que el software ya esté instalado. (RPM anulará el proceso si el software no se encuentra presente.) La mayoría de los comandos disponibles durante la instalación de un RPM también funcionará para una actualización. Utilizar una actualización RPM esencialmente ahorra al usuario el tener que borrar manualmente la instalación previa e instalar a continuación la nueva.

Nota: ¡Atención con las mayúsculas!

La sintaxis para una actualización RPM es `rpm -U <nombre de archivo>`; sin embargo, ¡versiones antiguas de RPM utilizaban una "u" minúscula con el significado de desinstalar, *uninstall*! Las versiones actuales de RPM utilizan la opción `-e` (por *erase*, borrar) en lugar de `-u`, pero debería tener cuidado a pesar de todo, especialmente cuando utilice sistemas Red Hat Linux más antiguos.

Información sobre paquetes instalados

Una de las más potentes características de RPM es su capacidad para preguntar al sistema acerca de los paquetes instalados. Esto permite a los usuarios comprobar con facilidad si una pieza de software está instalada (y si lo está, dónde se encuen-

tra ubicada), en lugar de tener que seguirle el rastro en el sistema para localizarla del modo complicado. También permite a los administradores mantener el sistema bajo control, ya que pueden generar una lista de todo el software instalado. Como la base de datos de RPM también sigue la pista de la información adicional descrita anteriormente en este capítulo dentro de la sección "Información sobre paquetes no instalados", RPM proporciona acceso a la documentación y a la información de dependencia de los paquetes instalados.

Preguntar a un paquete instalado resulta casi idéntico a preguntar a un paquete que aún no se ha instalado. La única diferencia es que en lugar de ejecutar el comando:

```
rpm -qp<opciones> <nombre de archivo>
```

los usuarios ejecutan el comando:

```
rpm -q<opciones> <paquete>
```

y substituyen el nombre del archivo de un paquete no instalado por el nombre de un paquete instalado en su lugar. Por lo demás, toda las opciones de preguntas que se muestran en la tabla 4.2 se pueden continuar utilizando.

Una variación bastante notable del comando de preguntas sobre paquetes es la opción de preguntar a todos los paquetes. El comando:

```
rpm -qa
```

permite a los usuarios realizar una pregunta sobre todos lo paquetes. Normalmente, se utiliza para conseguir una lista de todos los paquetes instalados, pero los usuarios también pueden añadir las opciones de pregunta de la tabla 4.2. Aunque, lo advertimos: comandos como `rpm -qai` generarán una gran cantidad de información de salida.

Verificación de los paquetes instalados

A lo largo del uso normal de un sistema, los archivos cambian. Esto puede resultar tan inocente como sencillamente cambiar un archivo de configuración en el directorio /etc o tan problemático como que un archivo se corrompa debido a un fallo de energía del sistema. Para ayudar en la detección de estos cambios, RPM permite a los usuarios y administradores verificar la integridad de un paquete instalado. El comando:

```
rpm -V <paquete>
```

o

```
rpm -Va
```

comprobará todos los paquetes instalados. Después RPM realiza el cálculo de la suma de comprobación MD5 de todos los archivos del disco y los compara con las

sumas MD5 almacenadas, procedentes de los RPM originales. La salida de este comando es una línea que indica cualquier archivo que se diferencie de las versiones que se instalaron originalmente y un código que indica cómo ha cambiado el archivo.

La verificación RPM es útil para detectar los archivos accidentalmente dañados y para mantener una lista de archivos que se han modificado manualmente. Es importante notar que muchos archivos (como los archivos nodo de dispositivo que se encuentran en /dev) se modifican casi inmediatamente tras la instalación, por lo que casi siempre habrá archivos modificados en cualquier sistema.

Además, aunque la verificación RPM puede ser una primera línea de defensa en la detección de una intrusión no autorizada (como un hacker que consiga entrar como si fuera el usuario root en la máquina), existen mejores herramientas disponibles, como Tripwire, que viene junto con muchas distribuciones, incluida Red Hat 7.3. La funcionalidad de verificación de paquetes RPM debería utilizarse como rudimentario detector de daños, y no se debería confiar en ella como herramienta de seguridad. Podrá encontrar la versión de código abierto de Tripwire en www.tripwire.org.

Reconstrucción de paquetes

A veces un usuario querrá instalar una pieza de software con particularizaciones distintas de aquellas contenidas en el RPM por defecto tal y como se distribuye, o estará utilizando un sistema Linux en una arquitectura para la que no existe un archivo binario RPM. A veces un fabricante de software sencillamente no puede seguir la pista de todas las posibles configuraciones objetivo de su software y desea proporcionar por conveniencia un paquete de código fuente para su producto. Para enfrentarse a estos casos, el formato RPM de Red Hat incluye el soporte de RPM fuente que contiene código fuente para el software. Estos RPM tienen la extensión .src.rpm y se pueden reconstruir como RPM binarios.

Recuerde que un archivo RPM es realmente un archivo cpio con alguna información adicional. Esta información adicional está contenida en un archivo denominado archivo spec (*spec file*), que es una abreviatura para "archivo de especificación" (*specification file*). Un archivo spec contiene toda la información necesaria para construir un RPM, como la lista de dependencias, la lista de capacidades requeridas y demás. Los archivos spec los escriben desarrolladores del software o el empaquetador del archivo RPM. Todo lo necesario para construir un archivo RPM binario es el archivo spec y el código fuente.

Hay dos maneras de reconstruir un paquete RPM. Si se dispone de un RPM fuente, el comando:

```
rpm --rebuild <nombre de archivo>
```

reconstruirá ese archivo como un RPM binario (suponiendo que no se produzcan errores durante la creación). Este es el caso más simple, y funcionará para casi todos los RPM fuente. No obstante, el comando RPM soporta opciones adicionales

para reconstruir convenientemente archivos RPM a partir de otros formatos. En ocasiones, los desarrolladores de una pieza de software harán pública su aplicación en un formato neutral, como un *tarball* (archivo comprimido con la extensión .tar), pero incluirán un archivo spec con él. En este caso, el *tarball* se puede colocar en el directorio especial /usr/src/redhat/SOURCES, y el archivo spec en /usr/src/redhat/SPECS. Entonces, el comando:

```
rpm -ba /usr/src/redhat/SPECS/<nombre de archivo>
```

reconstruirá el *tarball* de acuerdo con las instrucciones dadas en el archivo spec. El resultado de este comando será un archivo RPM en /usr/src/redhat/RPMS/<arq> (donde <arq> es la arquitectura de su sistema, como i386, como muestra la tabla 4.1). A continuación, este archivo RPM se puede instalar normalmente como cualquier otro archivo RPM. También se puede copiar a otro sistema e instalarse allí (asumiendo que las dependencias se cumplen); de hecho, este es generalmente el modo en que la mayoría de los RPM se crean de cara a su distribución.

Desinstalación de paquetes

De vez en cuando hay que desinstalar software de un sistema. Esta es históricamente una de las tareas más difíciles en la administración de un sistema informático. A los usuarios de Windows les resulta familiar el infierno DLL, que hace referencia al problema de las bibliotecas compartidas "huérfanas" y las múltiples (incompatibles) versiones de bibliotecas instaladas o abandonadas por diversos programas. En los sistemas basados en RPM, sin embargo, este problema queda muy reducido.

Debido a que la base de datos de RPM sigue la pista de todos los archivos instalados por un determinado paquete y se asegura de que un paquete no reemplace un archivo que pertenezca a otro, RPM siempre sabe qué archivos pertenecen a un determinado paquete, simplificando la desinstalación de un paquete. La capacidad de RPM de mantener la información de dependencia de los paquetes también le permite asegurarse de que un usuario no borre un paquete que proporcione una prestación necesitada por otro paquete. Estos comportamientos pueden anularse o ignorarse, pero generalmente RPM tiene un gran éxito en la desinstalación de software.

El comando para desinstalar un paquete RPM es simplemente:

```
rpm -e <paquete>
```

La mayoría de las opciones para instalar y actualizar (por ejemplo, `--nodeps` y `--force`) también se soportan en la operación de borrado. Si RPM determina que es seguro desinstalar el software, entonces el comando simplemente acabará. Sin embargo, si una comprobación de dependencia falla o ocurre algún otro error, RPM indicará cuál es el error y no eliminará el paquete. Si el problema es que otro paquete depende del paquete que se va a eliminar, el usuario debe decidir si se pueden eliminar ambos programas.

Creación de nuevos paquetes

Crear un nuevo paquete RPM generalmente implica crear un archivo spec y utilizar el programa RPM para construir un archivo RPM a partir de un archivo de código abierto de acuerdo con las instrucciones indicadas en el archivo spec. Crear un archivo spec requiere un conocimiento exhaustivo del software que se va a distribuir, y por eso generalmente debe ser creado por un desarrollador u otro colaborador familiarizado con el software. Hay múltiples fuentes de documentación sobre cómo crear un archivo spec y un RPM, así que los lectores y desarrolladores interesados deberían consultar estas fuentes. Una fuente excelente es el libro de Red Hat "Maximum RPM", que podrá encontrar en www.rpm.org/max-rpm.

Prestaciones adicionales

RPM dispone de algunas características extra para facilitar su uso. Estas características no incluyen ninguna funcionalidad extra más allá de las operaciones básicas tratadas con anterioridad en este capítulo, pero en cambio hacen más fácil y seguro para el usuario la utilización de RPM en esas operaciones. Específicamente, RPM soporta archivos RPM firmados criptográficamente y operaciones con archivos RPM a través de una red.

La mayoría de los usuarios de Red Hat Linux instalan el sistema a partir de algún medio físico como un CD o a través de la red desde un servidor público. En cualquier caso, es extremadamente común para los usuarios el descargar archivos RPM de Internet e instalarlos en sus sistemas. Después de todo, la conectividad de red es una parte importante del éxito de los sistemas Linux. Como conveniencia para estas operaciones, RPM soporta interiormente la descarga automática de archivos RPM desde servidores FTP o HTTP. Esto permite a los usuarios instalar un paquete RPM con un único comando en lugar de tener que descargar manualmente un archivo. Por ejemplo, el agobiante comando:

```
rpm -i ftp://ftp.redhat.com/pub/redhat/redhat-7.3-en/os/i386/RedHat/RPMS/
zsh-4.0.4-5.i386.rpm
```

instalará el paquete zsh para Red Hat Linux 7.3 desde el servidor público FTP de Red Hat.

Sin embargo, descargar archivos a través de la red puede resultar peligroso, ya que es posible que usuarios hostiles pueden colocar paquetes que hagan de "caballo de Troya" que parezcan ser auténticos pero de hecho contengan código hostil que podría dañar el sistema de un usuario o dejarlo expuesto a un ataque. Para minimizar este riesgo, RPM puede utilizar el programa Protección de Privacidad de GNU (*GNU Privacy Guard*, GPG) para verificar la autenticidad de los archivos RPM que han sido firmados criptográficamente. Puede acceder al programa GPG mediante el comando `gpg`. Si quiere más información, visite el sitio web de Protección de Privacidad de GNU (*GNU Privacy Guard*) en www.gnupg.org.

Mecanismo frente a interfaz de usuario

Hasta aquí, sólo hemos hablado de RPM como de una herramienta para gestionar paquetes que el usuario ya tiene. RPM mismo es únicamente una herramienta de línea de comandos y no dispone de una interfaz gráfica de usuario. El motivo de estas limitaciones es que RPM está orientado estrictamente a la administración de paquetes. La filosofía de Red Hat es mantener el mecanismo separado de la interfaz de usuario, y por eso RPM confía en herramientas adicionales para proporcionar una interfaz de usuario atractiva.

El programa up2date de Red Hat

La herramienta de interfaz de usuario preferida de Red Hat es el programa up2date. up2date es tanto un programa de línea de comandos como un programa gráfico que se comunica con un servidor para obtener actualizaciones de software e instalarlas. up2date sigue la pista del perfil de un sistema (que es esencialmente una lista de qué RPM tiene instalados un sistema) y actualizará cualquier paquete para los que exista una versión posterior en el servidor. Esto permite a los usuarios y administradores seguir la pista de cerca a correcciones de seguridad y errores hechas públicas por los administradores del servidor. En una distribución por defecto de Red Hat Linux, up2date se configura para utilizar los servidores de Red Hat. Red Hat permite acceso básico a sus servidores gratuitamente pero tiene un plan de cuotas de suscripción basado en up2date. Los usuarios que no deseen pagar por este servicio de Red Hat (como aquellos que administran una base de instalaciones de Red Hat Linux en un negocio) pueden preparar su propio servidor o escoger otro proveedor del servicio si es que hay alguno disponible.

Otros programas GUI

Una plétora de otros programas GUI actúan como interfaz de RPM. Por ejemplo, los proyectos GNOME y KDE tienen cada uno sus propias herramientas de ese tipo, llamadas gnoRPM y kpackage, respectivamente. Estos programas son bastante similares en prestaciones y proporcionan una interfaz en la que señalar y hacer clic, arrastrar y soltar para gestionar paquetes RPM. Generalmente, estas aplicaciones también están integradas con sus escritorios de diversas maneras. Fundamentalmente, RPM es realmente una simple especificación de un formato de archivo y una base de datos de información sobre los RPM instalados en un sistema. Otro sistema de empaquetado puede ser compatible con el formato RPM si simplemente soporta el formato de archivo. El formato de empaquetamiento de Debian ofrece soporte para archivos RPM en cierta medida en su propia herramienta, apt-get.

Ejemplos de uso de RPM

Como muchas herramientas en sistemas Unix, RPM es bastante sofisticada e incluso (me atrevo a decirlo) misteriosa. Típicamente, mucha de la auténtica poten-

cia de RPM proviene de su uso junto con otros comandos. Esta sección muestra algunos ejemplos útiles de RPM para realizar tareas comunes. La tabla 4.3 muestra una lista de algunos comandos comunes y describe el resultado de su ejecución. Excepto donde se indique, estos comandos son generalmente seguros (los usuarios pueden experimentar libremente con ellos).

Tabla 4.3. Comandos RPM útiles.

Comando	Descripción
`rpm -qf 'which <comando>'`	Muestra el paquete que instaló `<comando>`.
`rpm -ql <paquete> | grep /bin`	Encuentra los binarios del programa (por ejemplo, archivos en /usr/bin) instalados por `<paquete>`.
`rpm -qfi 'which <comando>'`	Muestra una descripción del paquete que instaló `<comando>`; puede ser útil para comandos sin página de manual.
`rpm -qfl 'locate <archivo>'`	Muestra todos los archivos provenientes del mismo paquete que `<archivo>`. (¡Cuidado, `locate` puede devolver más de un resultado!)
`rpm -qa | grep <paquete>`	Muestra el nombre de paquete exacto para `<paquete>`; es útil cuando sabe que un paquete está instalado pero desconoce su nombre exacto.
`rpm -qi 'rpm -qa | grep <paquete>'`	Muestra la descripción de un paquete cuyo nombre no se conoce completamente.
`rpm -e 'rpm -qa | grep <paquete>'`	Útil para eliminar un paquete cuyo nombre no se conoce completamente. Este comando puede ser muy peligroso, úselo con precaución.

Si ha leído completamente esta sección, debería de tener en este momento un conocimiento útil del formato de empaquetado de Red Hat, RPM. Con este conocimiento, debería de poder gestionar el software instalado en su sistema, y con una ligera exploración de la documentación detallada disponible procedente de Red Hat, usará RPM como un profesional. El resto de este capítulo le proporcionará una comprensión similar de otros aspectos fundamentales de Red Hat Linux.

Estructura del sistema de archivos

Quizás el rasgo identificativo inmediatamente reconocible de cualquier sistema similar a Unix (incluidos los sistemas Linux) sea la estructura de su sistema de

archivos. En el capítulo 3 hablé del estándar de jerarquía del sistema de archivos de Linux (FHS). En esta sección aprenderá con qué cercanía sigue Red Hat Linux la FHS. Cuando concluya esta sección, será capaz de encontrar cualquier cosa en su sistema Red Hat Linux. Si aún no lo ha hecho, resultaría útil que revisara el FHS y mi explicación sobre él en el capítulo 3 en este momento. En general, Red Hat sigue de cerca el FHS. La tabla 4.4 resume los puntos clave de la estructura del sistema de archivos de Red Hat Linux.

Tabla 4.4. Red Hat Linux y el FHS Linux.

Directorio	Contenidos FHS	Comentarios sobre Red Hat
/etc	Archivos de configuración	Sigue el FHS; la estrategia general es tener un subdirectorio separado para cada componente.
/boot	Archivos necesarios para que arranque el sistema	Sigue el FHS.
/dev	Archivos que representan dispositivos hardware en el sistema	Sigue el FHS.
/lib	Bibliotecas compartidas necesarias por los programas de usuario	Sigue el FHS.
/bin	Programas de aplicaciones de usuario	Sigue el FHS.
/sbin	Programas usados principalmente para tareas administrativas	Sigue el FHS.
/home	Directorios locales propios de los usuarios	Sigue el FHS.
/mnt	Contiene subdirectorios donde se montarán los sistemas de archivos temporales	Sigue el FHS.
/usr	Archivos útiles para usuarios una vez que el sistema está en funcionamiento	Generalmente sigue el FHS, pero en ocasiones se desvía para casos especiales específicos.
/var	Archivos y directorios para el almacenamiento de la información de control o datos de programas y servicios	Generalmente sigue el FHS, pero en ocasiones se desvía para casos especiales específicos.
/usr/local	Archivos específicos de instalación	Presente debido al FHS, pero sin contenidos por defecto.
/opt	Software "opcional" específico de instalación	Presente debido al FHS, pero sin contenidos por defecto.
/tmp	Archivos que no son persistentes ni importantes	Sigue el FHS.

Tener un estándar de sistema de archivos para la instalación base de un sistema operativo está bien, pero no tiene ningún sentido si se ignora durante las siguientes instalaciones de software y el mantenimiento del sistema. Esto es, es tan importante seguir la pista de dónde sitúan los RPM los archivos como mantener la pista de dónde sitúa los archivos la instalación base. Por fortuna, la convención para los RPM es seguir el FHS; la mayoría de los RPM que se pueden descargar desde Internet (y todos los RPM propios de Red Hat) colocarán sus archivos en /usr o /, según sea apropiado.

Otro punto que es importante resaltar es cómo trata Red Hat Linux la partición /usr. El estándar FHS indica que /usr debería contener únicamente archivos "estáticos" que no necesitan cambiar frecuentemente. Cualquier archivo "variable" debería estar en otro directorio, como /var. Si esta regla se sigue de cerca, entonces el directorio /usr jamás debería ser modificado, porque cualquier archivo no estático pertenecería a otro directorio. Por ello, si el directorio /usr se sitúa en su propia partición, debería ser posible montar esta partición como de sólo lectura; esto ayuda a prevenir intrusiones y daños accidentales. Esta característica es soportada por muchos otros sistemas similares a Unix e incluso algunas otras distribuciones Linux. Versiones anteriores de Red Hat Linux no eran tan capaces de soportar un directorio /usr de sólo lectura sin tener que realizar ninguna modificación, pero Red Hat 7.3 soporta esta técnica.

Como ha visto, Red Hat Linux sigue el FHS relativamente de cerca, con unas pocas excepciones. Como usuario de un sistema Red Hat Linux, debería de estar al tanto de las convenciones establecidas por el FHS. Cuando quiera que necesite encontrar un archivo en particular o hacerse a la idea de cómo se instalará una pieza de software, puede seguir esta regla básica: en primer lugar compruebe el lugar en que el FHS dice que estará y, después, compruebe el lugar en que esperaría encontrarlo en un sistema Unix distinto. Si todo lo demás falla, puede intentar usar el comando `locate` si conoce el nombre del archivo que está buscando.

Guiones de arranque del sistema

En el capítulo 1 se trató acerca de los orígenes e historia de Linux, incluyendo una breve historia de los sistemas Unix. El capítulo 3 mostró los componentes básicos de una distribución Linux, incluyendo una breve mención a los guiones de arranque ejecutados por el superproceso init. En esta sección, aprenderá acerca de la arquitectura de guiones de arranque utilizada por Red Hat Linux, y también aprenderá cómo personalizar el comportamiento de un sistema en el arranque, tanto manualmente como mediante las herramientas proporcionadas por el sistema.

Uno de los dos sistemas Unix arquetípicos era System V de AT&T (SysV). Este sistema introdujo una técnica particular para estructurar los guiones que se ejecutan por parte de init en el arranque. La otra aproximación importante fue presentada por Berkeley Software Distribution (BSD). Estos dos modelos sirven al mismo

propósito final pero difieren significativamente en cómo lo llevan a cabo. La aproximación de SysV es un marco más complicado que es bastante sencillo automatizar (pero más complicado de modificar para los humanos), mientras que la aproximación BSD es comparativamente más simple y fácil de gestionar para los humanos (aunque sea más costoso automatizarla). Red Hat Linux utiliza el enfoque de SysV, y el resto de esta sección describe los detalles de este mecanismo.

Red Hat Linux no es el único sistema similar a Unix que utiliza el modelo de guiones de inicio de SysV; algunos sabores comerciales de Unix también lo utilizan. Una vez que haya leído esta sección, dispondrá de un conocimiento operativo del modelo de inicio de SysV en general, de modo que debería ser capaz de aplicar su conocimiento a cualquier sistema que utilice ese modelo, no únicamente a Red Hat Linux. A la inversa, si ha usado uno de estos otros sistemas previamente, puede que ya le resulte familiar el modelo de inicio de SysV. En este caso, puede limitarse ahojear o bien saltarse este capítulo; sin embargo, la sección "Uso de las herramientas de Red Hat" merece ser leída ya que puede ofrecerle alguna información adicional.

Comprensión de los niveles de ejecución iniciales

El proceso init es el padre de todos los demás procesos, es el superproceso. Consecuentemente, puede pensar en init como en el proceso maestro que gobierna el sistema. Como tal proceso maestro, init controla el ciclo de vida de todo otro proceso y por ello del mismo sistema.

Sin embargo, generalmente init no está directamente implicado en las actividades de sus procesos hijo. En su lugar, una vez realizado su trabajo, init generalmente permanece inactivo en un segundo plano y sólo despierta cuando el usuario solicita unas determinadas acciones o cuando un proceso crucial (como la consola tty que se utiliza para entrar en el sistema) necesita ser "regenerado".

En un momento determinado, init es uno de los siete estados generales; estos estados son los niveles de ejecución del sistema. Los niveles de ejecución están numerados de 0 a 6, y cada nivel se corresponde típicamente con un cierto modo de uso del sistema (como el modo de mantenimiento, el modo monousuario, o incluso el modo de reinicio del sistema). Init puede configurarse para realizar diversas acciones basadas en el nivel de ejecución en que se entra, y el superusuario puede indicarle que cambie de un nivel de ejecución a otro.

El estándar POSIX define el comportamiento general de init, pero no especifica qué estados representan los siete niveles de ejecución. Tradicionalmente, los sistemas Unix utilizan el nivel de ejecución 0 para detener el sistema, el nivel 1 para un único usuario y el nivel 6 para reiniciar el sistema.

El resto de los niveles (del 2 al 5) son generalmente niveles de ejecución multiusuario, pero lo que realmente sucede en estos niveles de ejecución depende del fabricante del sistema operativo. La tabla 4.5 resume los significados que Red Hat Linux asigna a estos niveles de ejecución; esta información está contenida en el archivo /etc/inittab.

Tabla 4.5. Niveles de ejecución de Red Hat Linux.

Nivel de ejecución	Significado	Comportamiento
0	Parada del sistema	Detiene todos los procesos, vacía las memorias caché del sistema de archivos, y cierra el sistema.
1	Modo monousuario	Abre una única consola con nivel de superusuario; se utiliza para el mantenimiento del sistema.
2	Multiusuario, sin NFS	Modo multiusuario normal, excepto en que el sistema de archivos de red (*Network Filesystem*, NFS) está deshabilitado. Suele utilizarse para servidores o estaciones de trabajo que sólo tienen usuarios locales.
3	Multiusuario normal	El modo típico para los sistemas (como servidores) que no necesitan normalmente un escritorio gráfico.
4	Sin uso	
5	Multiusuario con X11	Modo multiusuario normal con una pantalla gráfica X11 de entrada al sistema. La única diferencia entre este nivel de ejecución y el nivel 3 es que el gestor de entrada X11 inicie automáticamente.
6	Reinicio	Es como el nivel 0, pero reinicia el sistema en lugar de cerrarlo.

Se puede indicar a init que cambien de nivel de ejecución en un sistema que se encuentre activo mediante los comandos `init` o `telinit`. Por ejemplo, el método canónico para reiniciar un sistema Unix (heredado desde la antigüedad) es mediante la secuencia de comandos:

```
sync; sync; sync; init 6
```

que vacía manualmente la caché del sistema de archivos y después indica a init que reinicie el sistema. Siempre que init cambia de nivel de ejecución, en primer lugar finaliza o acaba con los procesos que no se corresponden con el nuevo nivel de ejecución, y después arranca todos los procesos que deberían estar ejecutándose. Mediante este comportamiento, detener y reiniciar un sistema mediante los niveles de ejecución 0 y 6 se lleva a cabo simplemente al no definir procesos que deban ejecutarse en esos niveles. Entonces, init finaliza todos los procesos y no inicia ninguno, a excepción de un simple guión de reinició o parada del sistema.

Utilizar init para cambiar entre niveles de ejecución del sistema resuelve sólo la mitad del problema de gestionar el software que se ejecuta en el sistema. Las acciones que init lleva a cabo (que mayoritariamente consisten en los procesos que init

ejecuta) también deben establecerse. Es aquí donde los guiones de shell mencionados en el capítulo 3 entran en acción. Para cada nivel de ejecución, se puede configurar init para que ejecute un programa correspondiente o realice alguna otra acción. La siguiente sección trata sobre los archivos importantes del paquete de guiones de arranque de Red Hat.

Estructura de archivos y directorios

La instalación de Red Hat de init y los guiones viene en dos paquetes RPM: SysVinit e initscripts. En Red Hat 7.3, los RPM que se utilizan son (para sistemas i386) SysVinit-2.84-2.i386.rpm e initscripts-6.67-1.i386.rpm. SysVinit instala el propio programa init y los programas auxiliares relacionados; initscripts instala los guiones de shell reales que init iniciará.

El paquete SysVinit

SysVinit es simplemente una implementación del programa init y no establece nada acerca de cómo deben de ser los guiones de shell (es el paquete initscripts el que establece el marco real que Red Hat Linux utiliza). De hecho, la mayoría de las distribuciones Linux utilizan SysVinit como programa init, aunque sus guiones de shell sean como los de SysV o los de BSD. La tabla 4.6 muestra la lista de los archivos que SysVinit instala. Esta lista no está completa; debería utilizar el comando `rpm -ql` para obtener una lista completa.

Tabla 4.6. Archivos importantes del paquete SysVinit.

Archivo	Propósito
/sbin/init	El programa init.
/sbin/telinit	Se usa para indicar a init que cambie de nivel de ejecución.
/usr/bin/wall	Se usa para enviar mensajes de advertencia a todos los usuarios de un sistema cuando init modifica el nivel de ejecución.
/sbin/pidof	Herramienta útil que se usa para obtener el identificador de proceso (*process ID*, PID) de un programa a partir de su nombre; pruebe a ejecutar `pidof init`.

La mayor parte de los archivos del paquete SysVinit pueden encontrarse un cualquier sistema similar a Unix que utilice un programa init compatible con SysV. La principal excepción es probablemente el programa pidof, que se utiliza para obtener el identificador de proceso (PID) de un proceso a partir de su nombre. Por ejemplo, el comando

```
pidof init
```

Devolverá 1 porque init siempre tiene un PID de 1. Este programa es útil en cualquier guión que necesite recuperar el PID de un proceso en particular y su uso resulta un poco más conveniente que toda una serie de comandos encadenados. `pidof miproceso` es más fácil de escribir que `ps aux | grep miproceso | cut -f 1 -d " "` y tiende bastante menos a producir errores, ya que el último comando devolvería cualquier proceso con "miproceso" en el nombre (como, por ejemplo, "noesmiproceso"). El comando `pidof` es obviamente un comando no necesario para que el sistema funcione correctamente, pero es un ejemplo típico de las mejoras incrementales que el software de código abierto realiza en la calidad de un paquete global (init, en este caso). Generalmente, los usuarios no interactuarán directamente con la mayoría de los archivos de SysVinit; si necesita más información que la contenida en la tabla 4.6, debería consultar la página man para el comando que necesite. Sin embargo, el paquete initscripts contiene los guiones reales que controlan el sistema y por eso se merece que hablemos sobre él.

El paquete initscripts

La tabla 4.7 muestra la lista de los archivos instalados por initscripts. Esta lista no es completa; utilice el comando `rpq -ql` para conseguir una lista completa.

Tabla 4.7. Archivos importantes del paquete initscripts.

Archivo	Propósito
/etc/inittab	Configura init; especifica qué guiones de shell invocará init en cada nivel de ejecución.
/etc/sysconfig	Contiene archivos que especifican valores para variables de configuración de los guiones. Cada archivo da valores a un aspecto particular del sistema, como, por ejemplo, la red.
/etc/rc.d/rc.sysinit	Es invocado por init una vez cuando arranca el sistema.
/etc/rc.d/rc	Es invocado por init cuando se cambia de nivel de ejecución.
/etc/rc.d/rc.local	Un guión que se invoca en último lugar del proceso; configura detalles específicos de la máquina.
/etc/rc.d/rc{X}.d	Contiene enlaces a guiones de control de servicios que se utilizan para arrancar servicios para un determinado nivel de ejecución X (por ejemplo, /etc/rc.d/rc5.d).
/etc/rc.d/init.d	Contiene los guiones de shell reales referidos en los directorios rc.{X}.
/sbin/service	Una utilidad que se utiliza para arrancar y detener servicios manualmente de forma conveniente mientras que el sistema está ejecutándose.
/etc/X11/prefdm	Se usa para indicar qué gestor de escritorio X11 utilizar; es específico para el nivel de ejecución 5 en los sistemas Red Hat.

Nota

Con la versión 8, Red Hat probablemente moverá los directorios /etc/rc.d/rc{X}.d a /etc/rc{X}.d; de hecho, los enlaces simbólicos a estos directorios ya están presentes. Debería de estar al tanto de este inminente cambio de cara a la compatibilidad hacia delante.

/etc/inittab

El archivo /etc/inittab contiene la tabla de inicio (*init table*, inittab) para el sistema. Este archivo especifica qué guiones se invocarán para qué niveles de ejecución.

También especifica qué guiones se invocarán globalmente, como por ejemplo cuando arranque el sistema o cuando el usuario indique el reinicio mediante el "saludo de los tres dedos" (la combinación **Control-Alt-Supr**). El archivo /etc/inittab también indica el nivel de ejecución por defecto en que entra init durante el arranque.

/etc/sysconfig

El directorio /etc/sysconfig contiene datos básicos de configuración específicos para un sistema. Los guiones para un determinado nivel de ejecución configuran los servicios y el propio sistema para su operación. Estos guiones deben ser genéricos ya que se escriben para ejecutarse sobre cualquier sistema; no obstante, muchos servicios necesitan valores específicos para funcionar correctamente. Por ejemplo, el guión para configurar la red debería ser genérico para que pueda reutilizarse en cualquier sistema.

Sin embargo, la dirección IP de un sistema es obviamente específica a ese sistema. Para mantener los guiones en una forma reutilizable, datos como las direcciones IP se colocan en el directorio /etc/sysconfig y son leídos por los guiones. El archivo /etc/sysconfig/network, por ejemplo, contiene datos para el guión que se encarga de la red.

/etc/rc.d

El directorio /etc/rc.d contiene todos los guiones de shell y el árbol de directorios utilizado por init. Hay un subdirectorio por cada nivel de ejecución dentro de /etc/rc.d, nombrado de acuerdo con el número del nivel. Por ejemplo, el "directorio rc" para el nivel de ejecución 5 es /etc/rc.d/rc5.d. También hay un directorio adicional llamado /etc/rc.d/init.d que contiene los archivos físicos de guiones de shell; los contenidos de los directorios rc individuales son enlaces simbólicos a los guiones del directorio init.d. Fíjese en que hay enlaces simbólicos a cada uno de estos directorios dentro del directorio base /etc; sin embargo, esos enlaces apuntan a los directorios correspondientes en /etc/rc.d y están presentes por conveniencia y por compatibilidad hacia delante. (Una versión posterior de Red Hat probablemente desplazará los directorios rc a /etc permanentemente.)

rc.sysinit

El archivo rc.sysinit es un guión que se ejecuta una vez que se inicia el sistema, antes de entrar en ningún nivel de ejecución. Este guión está especificado en /etc/inittab y realiza tareas como la comprobación de discos y del sistema de archivos en busca de errores, la carga de cualquier módulo del núcleo o de controladores necesarios, restaurar el reloj del sistema y las semillas del generador de números aleatorios o cualquier otro trabajo que el sistema necesite, sin importar el nivel de ejecución en que se vaya a entrar. Si observa un sistema Red Hat Linux reciente durante su proceso de inicio, tiene dos "fases": una fase que siempre se realiza del mismo modo, sin importar el nivel de ejecución por defecto y una segunda fase en la que funcionan los servicios actuales para el nivel de ejecución. La primera fase (donde se le muestra la frase "*Press I to enter interactive startup*") se corresponde con el momento de la ejecución de rc.sysinit; la siguiente fase comienza cuando rc.sysinit termina su proceso e init entra en el nivel de ejecución por defecto.

Cuando init entra en un nivel de ejecución particular, ejecuta el guión que se corresponde con ese nivel de ejecución en /etc/inittab. (De hecho, init podría configurarse para ejecutar más de un guión siguiendo una secuencia en particular, pero Red Hat emplea una aproximación diferente.) Este guión es entonces responsable de ejecutar cualquiera servicios que sean apropiados para ese nivel de ejecución. La siguiente sección describe este proceso con más detalle.

Seguir la pista a una ejecución

Quizás el mejor modo de mostrar el comportamiento de los guiones de inicio de Red Hat sea examinar la copia de /etc/inittab que viene con Red Hat Linux y seguir su rastro a través de una secuencia de arranque de ejemplo. El resto de esta sección analiza minuciosamente el archivo inittab y asocia sus contenidos con eventos individuales que ocurren mientras arranca un sistema Red Hat Linux. Para una documentación completa sobre el archivo inittab, debería consultar la página man. Todo lo que necesita saber de cara a los propósitos de esta explicación es que la primera columna (antes de los primeros dos puntos) en las líneas que vienen a continuación es implemente un identificador y que puede ser arbitrario.

La primera línea no comentada en inittab especifica el nivel de ejecución por defecto. En el caso de Red Hat Linux, este es normalmente 3 o 5. Es 5 si el usuario seleccionó la opción de iniciar en modo gráfico durante la instalación, y 3 en cualquier otro caso. En la tabla 4.5 (y en el propio archivo /etc/inittab), el nivel de ejecución 3 está indicado como modo multiusuario normal, mientras que el nivel 5 es modo multiusuario con entrada gráfica. La diferencia está en los servicios que se inician en cada nivel de ejecución. La línea que indica el nivel de ejecución por defecto es esta:

```
id:5:initdefault:
```

La siguiente línea especifica el guión global rc.sysinit que init invocará antes de entrar en el nivel de ejecución por defecto. Este guión, tal y como se describió en la

sección anterior, es el responsable de comprobar los discos en busca de errores, limpiar el directorio /tmp, cargar cualquier módulo del núcleo o controlador necesario, y otras tareas administrativas similares. Este archivo es también el que genera el mensaje "*Press I to enter interactive startup*" durante el arranque. Este archivo no es demasiado complicado, así que los lectores interesados en los detalles deberían simplemente analizar este guión. Es posible definir más de uno de estos guiones globales para que se ejecuten; no obstante, Red Hat escoge utilizar un solo guión de este tipo. La línea que especifica este guión global de arranque es esta:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

El siguiente grupo de líneas asigna niveles de ejecución (en la segunda columna) a guiones (en la última columna). Como es inmediatamente obvio, Red Hat Linux utiliza el mismo guión para todos los niveles de ejecución y simplemente pasa al guión un argumento diferente que depende del nivel de ejecución en que se ha entrado. El comportamiento de este guión es bastante simple: mira en el directorio rc correspondiente para el nivel de ejecución que se pasó como argumento y luego invoca cualquier programa que encuentre en ese directorio. El guión /etc/rc.d/rc se tratará con un poco más de detalle en la sección "Guión de configuración del nivel de ejecución", así que por ahora aquí están las líneas más relevantes:

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
...
l6:6:wait:/etc/rc.d/rc 6
```

Las siguientes cuatro líneas determinan el comportamiento de init cuando conmuta entre niveles de ejecución o cuando recibe ciertos eventos significativos. La primera línea arranca el demonio de actualización del núcleo para sincronizar periódicamente la memoria caché con el disco, la segunda línea indica a init que ejecute el comando de parada del sistema cuando el usuario pulse la combinación **Control-Alt-Supr**, y las dos últimas líneas especifican los comandos que se ejecutarán si al sistema se le avisa de un fallo de energía. (Estos dos últimos eventos sólo pueden ser disparados si el sistema está conectado a una fuente ininterrumpida de energía [UPS] adecuadamente configurada.)

```
ud::once:/sbin/update
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
```

El siguiente grupo de líneas indica a init que cree varias consolas de texto virtuales (conocidas como tty), pero sólo para los cuatro niveles de ejecución multiusuario. Estas consolas tty son las seis consolas de texto que los usuarios pueden utilizar en lugar de una interfaz gráfica X Window; los usuarios pueden conmutar entre ellas mediante las combinaciones de teclas **Alt-F1** a **Alt-F6**. Se definen que las tty se regeneren, significando esto que init las reiniciará cuando termine su ejecución. Una vez que el usuario sale del sistema, la tty termina su ejecución e init ejecutará

otra en su lugar, que volverá a mostrar el mensaje de entrada al sistema de nuevo. Aquí están las líneas que definen las tty:

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
...
6:2345:respawn:/sbin/mingetty tty6
```

La última línea del archivo indica a init que ejecute el programa de entrada al sistema gráfico prefdm. La segunda columna, sin embargo, restringe este comportamiento al nivel de ejecución 5, de modo que no se invocará para ningún otro nivel de ejecución. Esta es la línea que provoca que la pantalla gráfica de entrada al sistema de KDE o GNOME aparezca en el monitor tras el arranque del sistema. (Como esta línea está tras la línea anterior del guión rc, se invoca una vez que init haya entrado en el nivel de ejecución 5.)

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Truco

Si indica a Red Hat Linux que inicie el sistema gráfico de entrada en el sistema en el arranque pero con posterioridad decide que no le gusta esto (o viceversa), puede solucionar este comportamiento mediante la edición del archivo /etc/inittab y la modificación del nivel de ejecución por defecto "initdefault" a 3 (para deshabilitar la entrada gráfica) o a 5 (para activarla).

Toda esta información puede reunirse en una rastro a través del arranque de un sistema Red Hat de acuerdo con los siguientes pasos:

1. El núcleo arranca e inicia init, que después lee /etc/inittab.
2. El archivo /etc/inittab indica a init que invoque al guión /etc/rc.d/rc.sysinit; si ese guión se completa con éxito, init avanza hasta la siguiente línea. Si ese guión devuelve un error (debido a un error en el disco, por ejemplo), entonces init se detiene y se entra en modo monousuario.
3. Una vez que rc.sysinit completa su ejecución con éxito, init comprueba el nivel de ejecución por defecto (que en este ejemplo será 5) y después invoca al guión que está definido para ese nivel de ejecución.
4. El guión definido arrancará cualquier servicio que se necesite para ese nivel de ejecución. Una vez que se complete esa ejecución, init registrará las siguientes líneas del archivo, definiendo cómo debería reaccionar ante diversos eventos, y después alcanzará las líneas en que se definen los tty. Si el nivel de ejecución es uno de los definidos en estas líneas (como sería para cualquier nivel de ejecución excepto el modo monousuario y la parada o el reinicio del sistema), init arranca /sbin/mingetty en los tty para permitir al usuario entrar en el sistema y después espera que terminen su ejecución.

5. Finalmente, init realiza la misma comprobación para la última línea, que determina si debería iniciar el programa "gestor de escritorio favorito" (*preferred desktop manager*) /etc/X11/prefdm (que a su vez invoca a los programas gráficos de entrada en el sistema de KDE o GNOME, si es que se encuentran presentes).

Llegados a este punto, el sistema ha arrancado completamente y está listo para su uso. Una secuencia muy similar de eventos ocurre si el superusuario indica a init que cambie a un nivel de ejecución diferente; la única diferencia es que el guión rc.sysinit no se ejecuta, ya que se ha definido que se invoca una única vez, durante el arranque del sistema. La única parte de esta secuencia aún no descrita en detalle es el comportamiento del guión /etc/rc.d/rc que se invoca cuando init entra en un nivel de ejecución, este guión se comentará en la siguiente sección.

Guión de configuración del nivel de ejecución

Todo lo descrito en la sección anterior es algo automático para un sistema Red Hat Linux. Los usuarios finales (y en la mayoría de los casos, los administradores del sistema) generalmente no necesitan modificar ninguno de los guiones que init invoca cuando arranca. De hecho, es uno de los objetivos generales de diseño de Red Hat para el sistema: los usuarios deberían ser capaces de configurar el sistema sin tener que editar guiones. La particularización más común que los usuarios realizan en un sistema es simplemente modificar qué servicios se arrancan para un determinado nivel de ejecución. Esta sección describe el mecanismo que se utiliza para automatizar esto en Red Hat Linux.

Recuerde que el archivo de configuración /etc/inittab de Red Hat Linux utiliza el mismo guión de shell (/etc/rc.d/rc) para configurar todos los niveles de ejecución. Este guión de shell toma el número del nivel de ejecución como entra y configura los servicios como sea apropiado para dicho nivel de ejecución. Lleva a cabo esto buscando archivos en el directorio rc correspondiente al nivel de ejecución (/etc/rc.d/rc5.d en nuestro ejemplo). Este directorio debería contener archivos cuyos nombres se ajusten a un cierto patrón; el guión rc utiliza este patrón para arrancar y detener el servicio según resulte apropiado. Los archivos propiamente dichos deberían ser, pero no necesitan ser, enlaces simbólicos a guiones de shell físicos que se encuentren dentro del directorio /etc/rc.d/init.d. La tabla 4.8 resume el patrón de nombrado.

Esencialmente, si un nombre de archivo comienza con "K", el servicio correspondiente es "matado" (*killed*, aunque realmente se finaliza, no se mata) cuando init entra en ese nivel de ejecución. Si el nombre del archivo comienza con "S", es "iniciado" (*started*) siempre que init entre en ese nivel de ejecución. Detrás de "S" o "K" viene un número de dos dígitos que no tiene ningún significado excepto para ordenar los archivos entre sí, con el objetivo de que el usuario pueda controlar en qué orden se invocan los servicios. El resto del nombre del archivo es el nombre del servicio.

Tabla 4.8. Patrón de nombrado de archivos de guiones de inicio de SysV.

Formato	Significado	Ejemplo
KXXfoo	Detener el servicio llamado "foo", en XX lugar	/etc/rc.d/rc3.d/K87portmap: detiene el servicio portmap para el nivel de ejecución 3
SYYbar	Ejecutar el servicio llamado "bar", en YY lugar	/etc/rc.d/rc5.d/S55sshd: inicia el servicio sshd para el nivel de ejecución 5

El guión rc (/etc/rc.d/rc) sencillamente busca en el directorio adecuado para el nuevo nivel de ejecución, detiene cualquier servicio cuyo nombre de archivo comience con "K", e inicia cualquier servicio cuyo nombre de archivo comience con "S". Los nombres de archivo se invocan como guiones de shell y se les pasa como entrada "stop" o "start", dependiendo de si se detienen o inician, respectivamente. Por ello, para funcionar correctamente, cualquier guión de shell que esté dentro de un directorio rc debe de ser capaz de interpretar estas entradas. Realmente, hay dos entradas más: "status", que indica si el servicio está ejecutándose, y "restart", que es un atajo para un "stop" seguido de un "start". Si desea añadir un nuevo servicio al sistema, todo lo que necesita hacer es colocar un guión de shell que acepte estas cuatro entradas en el directorio /etc/rc.d/init.d y crear después enlaces simbólicos como sea apropiado en los demás directorios rc.

Nota: "Dejando caer" nuevas configuraciones

El concepto de configurar el sistema mediante la colocación de archivos en ciertos directorios merece que echemos un segundo vistazo. Es una de las técnicas más comunes de Red Hat porque facilita instalar software y conectarlo al sistema.

RPM es el formato de paquetes que utiliza Red Hat Linux, y este formato generalmente sólo puede instalar archivos en ubicaciones predefinidas. Sucede con bastante frecuencia que dos programas necesitan engancharse a la infraestructura de arranque para funcionar correctamente. Por ello, hay dos programas que necesitan modificar la configuración del sistema, y, mientras tanto, puede que el administrador del sistema ya haya modificado el mismo archivo durante la instalación de un paquete que no utiliza RPM. Resulta extremadamente inconveniente cuando dos programas separados machacan los cambios del otro programa mediante la modificación del mismo archivo. Debería ser capaz de resolver este problema con un uso intensivo del procesado de textos "inteligente", pero esa implementación supone una gran cantidad de trabajo, y ya que Red Hat no escribe la mayor parte del software que incluye, no pueden obligar a un uso universal de una única aproximación

centralizada. El modelo de guiones de inicio de SysV resuelve este problema al permitir a cada programa (en este caso, servicio del sistema) disponer de su propio archivo de configuración separado que ningún otro programa necesitaría modificar, los guiones de shell en /etc/rc.d/init.d. Los guiones fundamentales del marco siguen inalterados y se pueden gestionar por parte del sistema, y la configuración se realiza simplemente "dejando caer" (depositando) un nuevo archivo de configuración para un nuevo programa en el directorio. Los ingenieros de Red Hat se inspiraron en esta solución y han comenzado a utilizar la técnica en muchas otras partes. Puede considerarse que es una práctica recomendada muy exitosa utilizada con bastante rigor por parte de Red Hat.

La siguiente sección da un ejemplo acerca de cómo configurar manualmente los servicios para un sistema Red Hat Linux.

Configuración manual del sistema

Llegados aeste punto, sabe los suficiente como para particularizar la lista de servicios iniciados por su sistema Red Hat Linux. Sin embargo, para rematar el tema, esta sección muestra un ejemplo de cómo enganchar un nuevo servicio al sistema. El ejemplo se basa en las siguientes características de un software hipotético que se va a añadir al sistema:

- Representa un servicio (por ejemplo, un servidor o demonio) llamado "miservicio" e instalado en /usr/local/miservicio
- Incluye un guión que obedece a la semántica start/stop/restart/status
- Se ejecuta sólo en los niveles de ejecución multiusuario (del 2 al 5)
- Depende del servicio portmap, así que deberá iniciar tras ese servicio

El primer paso es copiar el guión de shell desde la instalación en el directorio /etc/rc.d/init.d. Esto no es realmente necesario; podría crear un enlace simbólico directamente en los diversos directorios rc que apuntara al directorio de instalación en /usr/local/miservicio. Sin embargo, es una buena práctica tener todos los guiones de shell en una única ubicación centralizada. Esto se vuelve especialmente importante si necesita que el servicio se ejecute en modo monousuario, ya que el modo monousuario puede no montar en absoluto la partición /usr/local, según como se haya instalado.

Una vez que el guión de shell se encuentra en /etc/rc.d/initd, debe activarse mediante la creación de enlaces simbólicos a él en los directorios rc multiusuario deseados. Como se va a activar el servicio para esos niveles de ejecución, el nombre del enlace simbólico debe comenzar con "S"; por ello, "S00miservicio" podría ser un buen primer intento. Sin embargo, ya que miservicio depende del servicio portmap,

debe iniciar con posterioridad. Como portmap es S13portmap en Red Hat Linux 7.3, el enlace para miservicio necesita ser algo como S15miservicio. (Cualquier número funcionará, mientras que sea mayor que el orden 13 de portmap.) Debe crearse un enlace con este nombre para cada nivel de ejecución multiusuario. El listado 4.1 contiene un simple guión de shell bash que se puede utilizar.

Listado 4.1. Creación de enlaces simbólicos para configurar un servicio

```
#!/bin/sh
cd /etc/rc.d
for i in 2 3 4 5; do
    cd rc${i}.d/
    ln -s ../init.d/miservicio ./S15miservicio
    cd ..
done
```

Como paso siguiente, se deben crear enlaces de terminación equivalentes en los directorios rc para los niveles de ejecución 0, 1 y 6. Como portmap se ejecuta como K87portmap, el "valor K" para mi servicio debe ser menor que 87 (por ejemplo, 85). Esto se puede lograr con sencillez modificando el guión del listado 4.1 para utilizar 0, 1, y 5 en lugar de 2-5 y cambiando "S15miservicio" por "K85miservicio". Ahora el servicio recién instalado arrancará y se detendrá adecuadamente en cada nivel de ejecución. Si desea arrancar o detener manualmente el servicio una vez que el sistema está en marcha, puede hacerlo del mismo modo en que lo haría para cualquier servicio (por ejemplo, `/etc/rc.d/init.d/miservicio restart`).

A veces necesita configurar algo que no es un servidor que se está ejecutando constantemente. En ese caso, podría resultar exagerado o incluso inapropiado tratarlo como un servicio y utilizar la aproximación anterior. Por ejemplo, podría necesitar simplemente ejecutar un programa en el arranque que guarde los datos y la fecha, que borre un archivo en /tmp, y otras tareas por el estilo. Existe un archivo en que se pueden ubicar estas modificaciones: /etc/rc.d/rc.local. Todos los niveles de ejecución invocan este guión en el arranque, que por defecto prepara el archivo /etc/issue (cuyo contenido se muestra como parte de los mensajes de solicitud de entrada).

Puede añadir pequeños "miniguiones" a este archivo si lo desea; no obstante, tenga en cuenta el hecho de que este archivo está compartido y que sufre de los problemas descritos al final de la sección anterior. Cualquier cambio que realice a rc.local podría quedar potencialmente sobrescrito por una actualización o por otro programa o usuario.

Todo este asunto de "enlaces simbólicos que apuntan a guiones de shell" ahorra una gran cantidad de trabajo, pero aún puede resultar bastante molesto en la práctica. Si se descubre gestionando mucho tiempo estos servicios, probablemente acabará por escribir algunos guiones de shell (similares al del listado 4.1) que automaticen gran parte del trabajo. Los ingenieros de Red Hat también reconocieron esto, e incluyeron algunos pequeños programas que hacen la vida un poco más fácil. La siguiente sección describe estas utilidades.

Uso de las herramientas de Red Hat

Red Hat incluye los tres comandos siguientes en su distribución:

- `/sbin/service`
- `/sbin/chkconfig`
- `/sbin/shutdown`

Estos comandos tienen como objetivo gestionar la configuración y uso de init y sus guiones. Estos comandos son sutilezas (el sistema puede administrarse sin ellos y no permiten a los usuarios hacer nada que no pudieran hacer anteriormente). Por este motivo, algunos de los usuarios expertos de Unix más partidarios del código puro se burlan de ellos. Sin embargo, la belleza de estos comandos (y de los sistemas Unix en general) está en que si no te gustan, no tienes que usarlos; el modo antiguo de hacer las cosas sigue funcionando. Esta sección describe qué son estos comandos y lo que hacen.

/sbin/service

El comando `/sbin/service` se incluye en el RPM initscripts, así que está disponible en cada instalación reciente de Red Hat Linux. Este programa es esencialmente un atajo para arrancar y detener manualmente los servicios que se encuentran en /etc/rc.d/init.d. En lugar de escribir `/etc/rc.d/init.d/miservicio start`, por ejemplo, podría escribir simplemente `service miservicio start`. En otras palabras, `service` le ahorra tener que escribir `/etc/rc.d/init.d`. El comando le permite realizar todas las acciones estándar (*start*, *stop*, *restart* y *status*).

/sbin/chkconfig

El comando `chkconfig` se incluye en su propio RPM (chkconfig-1.2.22-1) y se instala en /sbin/chkconfig. Este programa gestiona los enlaces simbólicos de los directorios rc; esto es, puede gestionar qué servicios se configuran para ejecutarse en qué niveles de ejecución. Es esencialmente una versión repleta de características del guión de shell del listado 4.1 (aunque con mucha más sofisticación), ya que crea y elimina los enlaces simbólicos según cual sea la entrada del usuario.

El programa chkconfig también proporciona un método para automatizar el proceso de adición de un nuevo guión de shell mediante la escritura de un comentario con el formato adecuado en su guión de shell. El formato es

```
# chkconfig <niveles_de_ejecución> <valor S> <valor K>
```

Cuando se ejecuta, chkconfig busca en el guión en busca de estos valores y los utiliza para construir los enlaces simbólicos. La línea para portmap, por ejemplo, es `# chkconfig 345 13 78`, que muestra de dónde provienen los valores "mágicos" de 13 y 87 para los niveles de ejecución 3 a 5. Para añadir un nuevo servicio con el apoyo de chkconfig, el usuario simplemente crearía una de estas líneas con los

valores apropiados para determinar en qué lugar de la secuencia de guiones necesita ejecutarse su servicio.

/sbin/shutdown

El comando `/sbin/shutdown` se incluye en el paquete SysVinit. Este programa se utiliza para indicar a init que conmute al nivel de ejecución 0, 1, o 6 (por ejemplo, detener el sistema, reiniciarlo o cambiar al modo monousuario). Una vez más, este comando es realmente un atajo para algo mayor (algo como `wall "El sistema se va a detener."; sync; sync; sync; init 0`) pero es una conveniencia que suele pasarse por alto por parte de los administradores que están acostumbrados a enfrentarse con otros sabores de Unix.

Si ha leído esta sección acerca del mecanismo de init para Red Hat Linux al completo, entonces ha dado un inmenso paso para poner el sistema bajo sus órdenes. Después de todo, una vez que sabe cómo se configura el sistema, puede utilizar esto como punto de partida para descubrir casi cualquier otra cosa. El resto de este capítulo trata otros aspectos de Red Hat Linux con un detalle igualmente exhaustivo. Lo próximo son las bibliotecas centrales del sistema.

Bibliotecas centrales del sistema

Tal y como se trató en el capítulo 1, Linux es sólo un núcleo de un sistema operativo; las bibliotecas de sistema son lo que proporciona al sistema operativo la mayoría de sus prestaciones. En esta sección, aprenderá un poco sobre las bibliotecas centrales que utiliza Red Hat Linux 7.3.

Generalmente, los usuarios no tienen que prestar demasiada atención a las bibliotecas de sistema; tienden ser bastante estáticas a lo largo de la vida de la versión de una distribución. Esto es, la mayoría de las correcciones de errores y otras actualizaciones similares tienen lugar en las aplicaciones de usuario en vez de en las bibliotecas de sistema. No obstante, sigue siendo importante saber qué bibliotecas de sistema están instaladas en su sistema, lo que hacen y cualquier cuestión relacionada.

La tabla 4.9 muestra una lista de los paquetes RPM más importantes para Red Hat Linux 7.3 y unos breves comentarios. Esta lista incluye no sólo las bibliotecas compartidas con las que se enlazan otros programas, sino también algunos programas clave, como la shell de comandos `bash` y el paquete de utilidades de archivos, fileutils. La tabla 4.9 está organizada según el nombre del paquete RPM (cada entrada de la tabla está empaquetada como su propio RPM). Mientras tanto, cada paquete RPM contiene múltiples bibliotecas (que son archivos con la extensión .so) y otros archivos de apoyo relacionados. Esta tabla debería verse como un punto de partida en lugar de cómo una referencia exhaustiva. La parte más dura del aprendizaje sobre las bibliotecas centrales del sistema es comprender qué bibliotecas son las más importantes; esa es la información que proporciona la tabla. Los lectores

deberían acudir a las páginas de los sitios web, de los manuales y otra documentación para aprender acerca de las bibliotecas que se utilizan.

Tabla 4.9. Paquetes de bibliotecas centrales de Red Hat Linux 7.3.

Paquete	Versión	Comentario
XFree86	4.2.0	Sistema X Window; la versión 4 contiene extensiones para la aceleración 3D y nuevos modos de representación.
glibc	2.25	Bibliotecas centrales del sistema de ejecución; contiene un enlazador/cargador, bibliotecas matemáticas, de seguridad, de red y de proceso paralelo.
bash	2.05a	*Bourne Again Shell*; shell principal del sistema (utilizada por el usuario root).
util-linux	2.10s	Herramientas específicas de Linux para el disco, la gestión de procesos y para usuarios.
PAM	0.75	Módulos de autenticación enlazables (*Pluggable Authentication Modules*); se utiliza para autenticar a los usuarios contra diversos repositorios de datos (LDAP, RADIUS, y otros).
gdbm, db2, db3	1.8.0, 2.4.14, 3.3.11	Bibliotecas de bases de datos de archivos; se usan por otros programas para almacenar e indexar información.
fileutils	4.1	Utilidades binarias; contiene programas básicos como `ls`, `cp`, `rm` y otros.

Merece la pena recordar los orígenes de algunos de estos paquetes. Como ya sabe a partir de la lectura del capítulo 1, el proyecto GNU de la Fundación para el Software Libre proporcionó gran parte del software que hizo que Linux fuera utilizable. A su vez, Linux proporcionó el núcleo que le faltaba a GNU. Bibliotecas y programas como estos son ejemplos concretos de la contribución de GNU. Los paquetes gdbm, fileutils, bash y glibc son todos ellos software GNU. Notablemente, XFree86 no lo es; al proyecto GNU aún le falta su propia implementación de X Window. (Se tratará XFree86 en más detalle dentro de la siguiente sección.)

El sistema X Window y el escritorio

Uno de los temas candentes en los medios técnicos en estos días es la denominada batalla por el escritorio, en la que "David" (la comunidad Linux) se enfrenta con

"Goliat" (empresas de software de escritorio como Apple y Microsoft). Dejando el drama a un lado, ofrecer una interfaz de usuario sólida resulta crucial para cualquier sistema operativo, y los sistemas Linux no son una excepción. Esta sección describe las soluciones de escritorio ofrecidas por Red Hat Linux 7.3.

El GUI que se suele escoger para un sistema similar a Unix está casi siempre basado en el sistema X Window. La mayoría de los sistemas comerciales, sin embargo, típicamente sólo "soportan" un entorno de escritorio que se ejecute bajo X. En el pasado, el Entorno de Escritorio Común (*Common Desktop Environment*, CDE) se propuso como el escritorio estándar para los sistemas similares a Unix y muchos fabricantes lo adoptaron. Sin embargo, CDE fue finalmente sustituido por otros productos, como Microsoft Windows, en cuanto a términos de interfaz de usuario y arquitectura de componentes. Esto, combinado con los costes típicamente altos de los sistemas Unix, provocó que Unix perdiera cuota de mercado a favor de Windows en el mercado de las estaciones de trabajo de escritorio. Hoy en día, hay varios entornos de escritorio que sobrepasan de lejos al CDE y que rivalizan con cualquier escritorio comercial. Red Hat Linux 7.3 soporta dos de estos entornos: el Entorno de Escritorio K (*K Desktop Environment*, KDE) y GNOME del proyecto GNU. (Estos dos escritorios se comentarán más adelante en esta misma sección.)

XFree86

Como la mayor parte de los sistemas similares a Unix libres, Red Hat Linux utiliza XFree86 como su implementación de X Window. La versión incluida con la versión 7.3, 4.0.x, introdujo varias mejoras importantes. XFree86 4.0 introdujo el soporte para tarjetas aceleradoras 3D y una API para lograr varios efectos gráficos mejorados, como la transparencia y el texto sin aliasing (sin efecto píxel). El soporte al hardware 3D hace a XFree86 mucho más viable para las aplicaciones de visualización 3D y para los juegos, haciéndolo mucho más competitivo contra lo que ofrecen los fabricantes comerciales de Unix y contra otros entornos de escritorio. Sin embargo, el número de tarjetas gráficas que actualmente soporta la funcionalidad de aceleración 3D es pequeño en el momento de la escritura de este libro. Sin duda, el número de tarjetas soportadas se incrementará con el tiempo.

Además del soporte al nuevo hardware, las prestaciones mejoradas de representación de XFree86 4.0 (conocidas como las extensiones de representación o *render extensions*) soportan efectos visuales avanzados muy atractivos, como ventanas transparentes y translúcidas, al igual que fuentes sin pixelación. Este soporte tenía como objetivo una de las carencias más frecuentemente citados y evidentes de XFree86.

Red Hat Linux incluye algunas herramientas útiles para llevara a cabo la configuración de XFree86 y todos los escritorios incluidos (como GNOME, KDE, fvwm, y otros). La tabla 4.10 muestra un resumen de estas herramientas.

Xconfigurator es una herramienta escrita por Red Hat para gestionar la configuración de XFree86. XFree86 típicamente se instala y configura para ejecutar el programa /etc/X11/X como el servidor X correcto para la tarjeta de video. Este archivo

es en realidad un enlace simbólico al servidor X adecuado, que suele estar instalado en /usr/X11R6/bin. Si cambia la tarjeta de video o si simplemente necesita modificar su configuración, en primer lugar necesita actualizar este enlace simbólico al binario del nuevo servidor X, y después modificar el archivo `XF86Config-4` con los nuevos valores. Xconfigurator automatiza estas tareas. Xconfigurator soporta tanto la versión 4.0 de XFree86 como las versiones 3.x anteriores, pero por defecto se dirige a las versiones 3.x, así que para poder utilizarlo para configurar XFree86 4.0 deberá utilizar el argumento `-preferxf4`.

Tabla 4.10. Herramientas relacionadas con XFree86.

Programa	Propósito
Xconfigurator	Configura XFree86 para un sistema en particular.
switchdesk	Automatiza el proceso de conmutar entre escritorios.
xev	Se utiliza para seguir la pista y mostrar los eventos de ratón y de teclado; es útil para la depuración y la prueba de aplicaciones.
xdpyinfo	Muestra información sobre la pantalla X actual.
xvidtune	Se utiliza para ajustar la configuración X para un monitor.

Los otros programas de la tabla 4.10 están destinados a ser utilizados para tareas de configuración menos amplias. EL paquete switchdesk lo utilizan los usuarios para cambiar su escritorio por defecto (por ejemplo, de GNOME a KDE). Los otros programas vienen incluidos en realidad con XFree86: xev se utiliza para vigilar los eventos de teclado y ratón y es útil para depurar correspondencias del teclado o configuraciones del ratón; xdpyinfo muestra información (como la resolución y la profundidad de color) acerca de la pantalla actual; y xvidtune se utiliza para ajustar la altura, anchura, centrado, y otros aspectos de la pantalla actual, y puede generar una línea que se debe añadir al archivo XF86Config para hacer el cambio permanente.

GNOME y KDE

No obstante, XFree86 es sólo una implementación del sistema X Window y por ello necesita un gestor de ventanas y un conjunto de componentes para poder ofrecer un entorno de escritorio.

Red Hat Linux 7.3 incluye KDE 3.0 y GNOME 1.4 como entornos de escritorio alternativos. Red Hat Software proporciona algún apoyo financiero al trabajo de desarrollo del proyecto GNOME, y por eso GNOME es el escritorio por defecto de Red Hat Linux. Sin embargo, Red Hat es en realidad "agnóstico" en cuanto al escritorio, y así proporciona diversas herramientas que permiten a los usuarios conmutar fácilmente entre GNOME y KDE como su escritorio. Red Hat cree en la inclusión

del mejor software de que se disponga, independientemente de sus orígenes, y por ello incluye tanto KDE como GNOME.

La información sobre estos entornos podría ocupar un libro completo con facilidad, pero merece la pena resaltar unos pocos puntos. Las últimas versiones de KDE y GNOME soportan (o pronto soportarán) fuentes sin *aliasing*, dándoles un aspecto más pulido que en versiones anteriores. Ambos escritorios también tienen modelos en evolución de componentes incrustados, que permiten a los desarrolladores de aplicaciones comenzar a utilizar algunas de las técnicas avanzadas de arquitecturas de componentes utilizadas en Windows. Además, ambos entornos ofrecen un amplio soporte para "temas" o "pieles" que permiten a los usuarios personalizar el aspecto y comportamiento de sus escritorios. En este camino y varios otros, tanto KDE como GNOME se encuentran por delante de sus competidores comerciales. El razonamiento de que los sistemas Linux no resultan viables como sistemas de escritorio está convirtiéndose poco a poco en un razonamiento sin fundamento.

El escritorio de usuario es una parte importante del sistema, pero al fin y al cabo es sólo una interfaz. La auténtica valía del sistema proviene de las aplicaciones reales que se instalen en él para que los usuarios trabajen con ellas. La siguiente sección tratará acerca de algunas de las aplicaciones que vienen junto con Red Hat Linux, así como algo de la aproximación de Red Hat a la distribución de estos programas.

Aplicaciones de usuario

Red Hat incluye una gran cantidad de software con su sistema. Generalmente, esto es cierto para la mayoría de las distribuciones Linux; tienden a ser paquetes de software muy completos. En un sistema comercial como Microsoft Windows o un sabor comercial de Unix, normalmente tendría que pagar para disponer de cosas como compiladores, software de productividad de oficina y servidores web, además de por el sistema operativo básico; pero este tipo de software es para los sistemas Linux tan gratuito como el sistema operativo en sí mismo. Esta sección describe el software que se incluye con un sistema Red Hat Linux.

Generalmente, Red Hat Linux contiene la "carta" habitual común a los sistemas Linux:

- El editor de texto emacs (así como su primo, xemacs)
- El compilador gcc y sus herramientas asociadas
- El servidor web Apache
- El software Samba basado en SMB para compartir archivos compatibles con Windows

Red Hat sigue a la vanguardia sin llegar a estar por delante de las demás distribuciones (al menos, la mayor parte del tiempo). Por ejemplo, en el pasado la suya

fue una de las primeras distribuciones en cambiar del paquete original libc5 típicamente empleado en Linux al paquete libc de GNU. Red hat también utiliza el último núcleo disponible; por ejemplo, Red Hat Linux 7.1 incluía la serie 2.4.x de núcleos al poco de que se hiciera pública. La versión 7.3 también incluye los últimos escritorios XFree86 y KDE, incluso utilizando su soporte de fuentes sin *aliasing*, a pesar de que esa característica sigue siendo técnicamente beta. Por otra parte, el proceso de garantizar la calidad de Red Hat a veces les impide liberar software "sobrecalentado". Ciertas pruebas con la serie 2.4 de núcleos, por ejemplo, mostró unos pocos problemas con el sistema de archivos *journaling* de ReiserFS, impidiendo su inclusión por defecto en Red Hat 7.1.

Red Hat Linux, como la mayoría de las distribuciones, contiene una gran cantidad de software. Parte de él se considera vital para la distribución, mientras que otra parte se considera principalmente de interés para los usuarios avanzados. De hecho, hay tanto software ¡que la distribución completa (en la que se incluyen paquetes adicionales) ahora consiste en tres CD, incluso más para el código fuente y la documentación! Para hacerse una idea de los que obtiene junto con Red Hat Linux, simplemente eche un vistazo a las listas de archivos RPM en los CD de instalación. El resto de esta sección trata acerca de cómo organiza Red Hat su distribución como sigue:

- Distribución central
- CD de código fuente y de documentación
- Ediciones de Red Hat Linux

Comprensión de la distribución central

La distribución central de Red Hat Linux contiene software que utilizan frecuentemente muchos o casi todos los usuarios. En versiones tan recientes como la 6.2, Red Hat Linux podía caber en un único CD. Sin embargo, debido al inexorable incremento del tamaño de los paquetes ya existentes y a la introducción de nuevos paquetes, la distribución central ahora consiste en tres CD, sin incluir el código fuente.

Por supuesto, cuantos más discos tenga que utilizar un usuario, más confusos se volverán los procesos de instalación y mantenimiento. Ahora, por ejemplo, hay tantos paquetes que realizar una instalación al gusto y seleccionar los paquetes individualmente puede tomar bastante tiempo, incluso para usuarios expertos, simplemente debido al número de paquetes disponibles. Para reducir estas dificultades, Red Hat identificó varios perfiles comunes de utilización del sistema, como estación de trabajo, servidor web, servidor de archivos, y otros más. Cada perfil se orienta hacia un patrón de uso diferente y está asociado a una lista de paquetes para ese perfil. El programa de instalación de Red Hat, anaconda, permite a los usuarios seleccionar uno o más de estos perfiles predefinidos para producir una configuración que se ajuste a la mayoría de sus necesidades. La cuestión, por supuesto, es

que estas configuraciones son inevitablemente la idea de alguna otra persona sobre lo que un sistema de ese tipo debería contener. Sin embargo, incluso aunque la mayoría de los usuarios terminen personalizando una configuración por defecto, al menos existen puntos de partida.

Para los usuarios (generalmente administradores de red y del sistema) que necesiten instalar múltiples instancias de la misma configuración, Red Hat proporciona un paquete conocido como kickstart que, básicamente, permite a estos administradores crear su propia configuración "enlatada". De este modo, los administradores pueden preparar una máquina servidor que contenga los paquetes de software deseados y utilizar un disquete para arrancar cada máquina objetivo. Las máquinas objetivo traerán entonces sus paquetes desde el servidor utilizado para la instalación. Adicionalmente, el software up2date puede personalizarse para trabajar con un servidor local de la red, así que los administradores pueden actualizar fácilmente sus instalaciones de acuerdo con sus propias necesidades. (Por supuesto, Red Hat no proporciona una versión personalizable del servidor up2date, así que deberá trabajar para conseguir o crear uno.)

CD de código fuente y documentación

Los CD de la distribución central de Red Hat Linux contienen el software típico, tanto para aplicaciones de escritorio como para aplicaciones de servidor. Sin embargo, algunos usuarios necesitan información adicional sobre la distribución y el software que incluye, mientras que otros usuarios pueden necesitar tener acceso al código fuente original del sistema (que está casi siempre disponible, ya que es software de código abierto). Para adaptarse a estas necesidades, Red Hat Linux 7.3 incluye no sólo los tres CD de instalación, sino que también dos CD con código fuente y un CD con documentación. Eso hace un total de seis CD, para aquellos que se encarguen de llevar la cuenta.

Ediciones de Red Hat Linux

Red Hat produce diferentes ediciones de su distribución. Cada edición tiene un enfoque diferente y algunas ediciones contienen algunos añadidos de software comercial que no es de código abierto. Las ediciones tienden a cambiar un poco de lanzamiento a lanzamiento, pero en el momento de la versión 7.3, son como sigue:

- Red Hat Linux Personal
- Red Hat Linux Professional
- Red Hat Linux Advanced Server

La edición *Personal* es adecuada para usuarios domésticos e individuales. La edición *Professional* está más orientada al entorno corporativo y otros usuarios de gran escala, y contiene algunas herramientas y software que resultan útiles en esos

contextos. La edición *Advanced Server*, mientras tanto, contiene diversas características y software muy bien pulidos, como la habilidad de agrupar servidores para sitios web, bases de datos, y otras tareas de servidor. Adicionalmente, esta edición incluye opciones premium de soporte. Red Hat también publica distribuciones para varias arquitecturas distintas de la i386, como ordenadores centrales IBM S/390 y el procesador Itanium de Intel.

Encontrar software adicional

En algún momento, casi todo usuario descubrirá que necesita software que no se incluye en la distribución de Red Hat. En estos casos, los usuarios suelen terminar acudiendo a la web para buscar software. Sin embargo, existen varias aplicaciones software que hacen este proceso más fácil. Las herramientas de gestión de paquetes de Debian integran la búsqueda de paquetes a través de la red y su descarga e instalación automática; RPM no soporta intrínsecamente esta característica, pero hay diversas herramientas disponibles que incorporan este comportamiento.

Una de estas herramientas es rpmfind. Esta herramienta es tanto un programa utilizado para buscar, descargar e instalar paquete RPM como un sitio web que se puede utilizar para búsquedas más cómodas de archivos RPM. Tanto la herramienta como el sitio web están mantenidos por Daniel Veillard del World Wide Web Consortium (W3C). El sitio web se encuentra en www.rpmfind.net. La herramienta rpmfind para los sistemas Red Hat Linux puede verse como una especie de último recurso; la mayoría de los usuarios tratarán (aunque no es necesario, por supuesto) a Red Hat como fuente principal de RPM vía up2date, simplemente por una cuestión de conveniencia. Sin embargo, para esos casos en los que Red Hat no mantiene oficialmente RPM para una determinada pieza de software, rpmfind puede resultar verdaderamente muy útil.

Idiosincrasias de Red Hat Linux

Hasta ahora, todo lo que se ha comentado en este capítulo es más o menos aplicable a otros sistemas Linux e incluso a algunos otros sistemas similares a Unix. Esto es, hasta ahora sólo he cubierto los pormenores de paquetes de software particulares y de herramientas que Red Hat Linux utiliza. En esta sección, cubriré algunas de las idiosincrasias específicas de Red Hat Linux.

Uso el cargador de arranque GRUB

Todos los sistemas operativos necesitan un cargador de arranque para iniciarse. Un cargador de arranque es un programa muy pequeño que el hardware del sistema ejecuta automáticamente y que es responsable de localizar, preparar y ejecutar el sistema operativo primario (en este caso, el núcleo Linux). La mayoría de las

distribuciones utiliza el cargador Linux (*Linux Loader*, LILO) como su cargador de arranque, pero, a partir de la versión 7.2, Red Hat Linux utiliza GRUB.

GRUB, el "*Grand Unified Bootloader*" (Espléndido Gargador de Arranque Unificado, sería su traducción al castellano), se desarrolló dentro del proyecto GNU. Es un cargador de arranque muy completo que contiene características que frecuentemente sólo se encuentran en sistemas y software comerciales de gama alta. Por ejemplo, GRUB permite a los usuarios editar la configuración de arranque manualmente al iniciar el sistema, mientras que LILO sólo puede crear configuraciones de arranque desde un sistema que ya esté funcionando. Esta característica de GRUB resulta extremadamente útil en el caso de que se equivoque en la configuración de arranque. Instalar GRUB es tan sencillo como instalar el RPM adecuado. (Por supuesto, probablemente debería desinstalar LILO, por si acaso.) GRUB es también el cargador de arranque por defecto, así que a no ser que lo cambie manualmente durante la instalación, ya lo tendrá instalado y funcionando en su sistema.

GRUB se configura mediante un archivo de texto. Como este archivo debe de ser accesible para GRUB durante el inicio del sistema, debe estar en la partición /boot. (El RPM de GRUB también crear un enlace simbólico en /etc/grub.conf que apunta al archivo real en /boot/grub/grub.conf.) Gestionar los sistemas operativos instalados en su ordenador se vuelven en algo tan sencillo como editar los contenidos de ese archivo. No obstante, GRUB está adecuadamente comentado en su página de manual y en su página web que se encuentra en www.gnu.org/software/grub/, así que no comentaré los detalles de la configuración de GRUB aquí.

Gestión de software similar

A veces podrá descubrir que necesita o quiere instalar dos programas independientes pero similares. Por ejemplo, tanto KDE como GNOME incluyen "gestores de escritorio" que permiten a los usuarios entrar en el sistema X Window. Sin embargo, sólo uno de estos programas se puede usar en un momento dado, así que debe asignar uno de ellos como programa a utilizar por defecto.

Red Hat Linux se enfrenta a esta situación mediante el uso del mecanismo de alternativas. Este es simplemente un directorio (específicamente, /etc/alternatives) que contiene enlaces simbólicos a la versión por defecto de un programa concreto en los casos en que más de un programa cubre una determinada necesidad. (Por ejemplo, el enlace /etc/alternatives/mta apunta al agente de transferencia de correo por defecto del sistema.)

El sistema de alternativas realmente se tomó prestado de la distribución Debian GNU/Linux. Como comentaremos Debian en el capítulo 6 (junto con el sistema de alternativas), debería consultar ese capítulo si quiere más información.

Actualización a un núcleo estándar

Red Hat realiza extensas pruebas para garantizar la calidad en todos los núcleos que incluyen oficialmente. Sin embargo, ya que el enfoque de Red Hat es el sistema

operativo al completo, generalmente no hacen públicos núcleos actualizados a no ser que deban añadir alguna funcionalidad importante o corregir un error. Por eso, no resulta extraña la necesidad de instalar un núcleo estándar en un sistema Red Hat Linux si la necesidad de una nueva característica o una corrección de errores es demasiado urgente como para esperar hasta el próximo lanzamiento de Red Hat. Esta sección describe cómo instalar un núcleo estándar en un sistema Red Hat Linux 7.3 sin dañar ningún otro software que pueda estar instalado.

Aviso

Antes de llevar a cabo estos pasos, es importante que comprenda los peligros que implican. El núcleo es la parte más crucial del sistema operativo y si se configura mal o se daña, el sistema no iniciará correctamente, o no iniciará en absoluto. Actualmente, los sistemas Linux están volviéndose más y más estables, así que los usuarios no deberían actualizar el núcleo sólo por hacerlo. No obstante, si se debe actualizar el núcleo por un núcleo estándar, se pueden utilizar estas instrucciones. Sea extremadamente cuidadoso al usar estas instrucciones, lea toda la documentación (tanto aquí como en cualquier otro sitio), y asegúrese de que tiene un disquete de arranque de rescate que funcione antes comenzar el proceso.

Para instalar un núcleo estándar en Red Hat Linux 7.3, siga estos pasos:

1. Instale las bibliotecas necesarias en caso de ser oportuno (como el paquete dev86 para Intel).
2. Elimine el enlace simbólico /usr/src/linux-2.4.
3. Extraiga el núcleo estándar en el directorio /usr/src.
4. Cree un enlace simbólico al nuevo núcleo como /usr/src/linux-2.4.
5. Copie la configuración del núcleo de Red Hat dentro del árbol de la nueva fuente (particularícelo si es necesario).
6. Construya el nuevo núcleo.
7. Copie los archivos bzImage y System.map en /boot, renombrados apropiadamente.
8. Actualice /etc/grub.conf para que se utilice el nuevo núcleo.
9. Reconstruya el sistema initrd mediante el comando `mkinitrd`, si el sistema lo utiliza.
10. Instale el nuevo núcleo para el arranque mediante la ejecución de `lilo`.

Tenga en cuenta que que probablemente esta lista no es lo suficientemente detallada como para seguirla de manera literal; sin embargo, Red Hat y otras fuentes

proporcionan documentación sobre la actualización del núcleo. Puede utilizar los pasos que se indican aquí como una lista de comprobación cuando lleve a cabo una de estas instalaciones. Tome nota también de que si desea volver más adelante a un núcleo estándar de Red Hat, tendrá que deshacer algunos de estos cambios (sobre todo el paso 7, ya que ese enlace simbólico provocará un conflicto con un paquete de código del núcleo de Red Hat).

El primer paso es asegurarse de que todo el software necesario está instalado. Lo sistemas Intel necesitan tener instalado el paquete dev86. Si el código fuente del núcleo ya está instalado, este paquete también lo estará. Si tiene instalado el código fuente del núcleo, debería de dejarlo instalado hasta después de que haya completado la actualización. Una vez que haya comprobado que su nuevo núcleo funciona correctamente, puede recuperar el espacio de disco, si realmente lo necesita, borrando el RPM del código del núcleo.

RPM kernel-headers

Puede haberse fijado en un segundo RPM instalado en su sistema que se llama kernel-headers (que contiene las cabeceras del núcleo). Este paquete, incluso aunque parezca estar relacionado con el núcleo, no necesita ser actualizado ni desinstalado. Este paquete contiene los archivos *include* del núcleo que utiliza un cierto software del espacio de usuario, como las bibliotecas C del sistema (glibc). La versión de glibc instaladas en el sistema depende de una versión particular de estos archivos de cabecera del núcleo, y generalmente no hay ningún motivo para actualizarlos a lo largo de un uso normal, aunque el propio núcleo se actualice. De hecho, hacer esto podría crear problemas. Ya que estos archivos no están relacionados con el proceso de construcción o ejecución de un nuevo núcleo, no hace falta actualizarlos, a no ser que haya una necesidad legítima muy fuerte. Para construir un nuevo núcleo, sólo se necesita actualizar el paquete kernel-source.

Enlace simbólico /usr/src/linux-2.4

Diversos componentes software y otras partes del sistema esperan encontrar el código fuente del núcleo para la serie 2.4 en el directorio /usr/src/linux-2.4. Este directorio debería ser un enlace simbólico al directorio real en el que está instalado el código fuente. Por eso, después de eliminar los RPM del código fuente, el siguiente paso es extraer el código fuente del núcleo estándar en el directorio /usr/src y posteriormente actualizar el enlace simbólico /usr/src/linux-2.4 para que apunte al directorio del nuevo núcleo estándar.

Una cuestión relacionada que merece la pena mencionar es que la aproximación de Red Hat consiste en instalar el código fuente del núcleo en el directorio /usr. Sin embargo, no necesita estrictamente estar ahí; un usuario normal puede (y quizás debería) generar un núcleo en su propio directorio local y después convertirse en superusuario para instalar realmente el núcleo. Por otro lado, el proceso completo de actualizar un núcleo es una tarea de administración de sistema, así que hay razones para colocar el código fuente en /usr/src. Para el propósito de este capítulo, intentémoslo y permanezcamos dentro de las convenciones propias de Red Hat.

Construcción del núcleo a partir del código fuente

En este punto, el código fuente del núcleo se ha instalado adecuadamente. El siguiente paso es construir realmente el núcleo a partir del código fuente. Para hacer eso, hay que configurar el núcleo. Puede utilizar el comando estándar `make menuconfig` u otros métodos para configurar el código fuente, pero Red Hat también proporciona sus archivos de configuración estándar del núcleo con el sistema. Mediante la reutilización de estos archivos, puede asegurarse de que su configuración del nuevo núcleo estándar es comparable a la de Red Hat, reduciendo así la posibilidad de que accidentalmente se omita una opción del núcleo que necesite el sistema. (Sin embargo, la configuración de Red Hat puede necesitar particularizaciones menores ya que los núcleos de Red Hat a veces incluyen opciones que los núcleos estándar no ofrecen. Esto variará en cada caso.)

Los archivos de configuración de Red Hat se encuentran en /lib/modules/<versión>/build/configs; copie el archivo apropiado para sus sistema en /usr/src/linux-2.4/.config y el proceso de construcción se encargará de encontrarlo. (El directorio <versión> de la ruta se refiere a la versión del núcleo antiguo que se va a actualizar: 2.4.2-2 en un sistema 7.3 estándar. Estos archivos no estarán presentes si ya se ha desinstalado el RPM del código del núcleo, que es por lo que esta operación debe realizarse al final.)

Una vez que el archivo de configuración de Red Hat se ha copiado (o se ha configurado de otro modo el código del núcleo), ya se puede construir el núcleo real. En este punto se aplican las instrucciones para construir un núcleo estándar; los usuarios deberían consultar la documentación en busca de una información más detallada. Esencialmente, este proceso implica ejecutar el proceso de construcción, que crea un nuevo núcleo en arch/i386/bzImage (en los sistemas Intel, o en otro lugar para otras arquitecturas) al mismo tiempo que un archivo System.map. Estos archivos deben situarse en el directorio /boot, hay que actualizar el archivo de configuración del cargador de arranque (/etc/grub.conf o /etc/lilo.conf para sistemas Intel), y en los sistemas que lo necesiten, hay que construir un nuevo archivo /boot/initrd mediante el comando `mkinitrd`. Durante la configuración del cargador de arranque, asegúrese de que añade un nuevo núcleo, en lugar de reemplazar el viejo; si por cualquier motivo el nuevo núcleo sale mal, puede recuperar el sistema al ejecutar la configuración antigua.

Truco

Estas instrucciones son algo esquemáticas. Sin embargo, puede encontrar una extensa documentación proveniente del *Linux Documentation Project* (LDP) en su sitio web www.linuxdoc.org/HOWTO/Kernel-HOWTO.html y en el sitio web de Red Hat. El LDP tiene documentación sobre muchos aspectos de los sistemas Linux y es un importante recurso que merece la pena tener apuntado en su lista de sitios web.

Una vez completados los pasos anteriores, puede reiniciar el sistema (mediante el comando `shutdown`, ejecutando `telinit 6`, o sencillamente con la combinación de teclas **Control-Alt-Supr**), y el nuevo núcleo debería de entrar en funcionamiento. Después de que se haya reiniciado el sistema, puede entrar en él y utilizar el comando `uname -a` para comprobar la actualización.

Entorno de usuario

Los usuarios de los sistemas similares a Unix adquieren un mal genio especial al hablar acerca de sus shells de comandos; quizás el único tema más sensible que éste sea el debate entre los incondicionales del editor de textos emacs y de vi que ha ocupado a los usuarios de Unix desde sus más remotos comienzos. Casi invariablemente, lo primero que un experto en Unix hace cuando se le concede una cuenta en un nuevo sistema es personalizar su entorno de shell. Esta sección está dedicada a aquellos expertos y describe la configuración del entorno de usuario de Red Hat Linux (no puede personalizar lo que no conoce).

Cada shell de comandos lee su configuración de una serie de archivos con nombres especiales. Generalmente, comienzan con un archivo del directorio /etc que contiene información de configuración global y específica del sistema y después van a los archivos personales de un determinado usuario. Lo que realmente sucede en estos archivos depende totalmente del usuario (y del administrador del sistema). Una de las cosas más difíciles de manejar, a pesar de todo, es el caso en que se instala un software nuevo que necesita que los usuarios definan variables de entorno. En este caso, se deben modificar los archivos de configuración global para que los cambios tengan efecto para todos los usuarios. Sin embargo, el administrador puede haber personalizado estos archivos, y por eso la instalación automática de nuevos archivos puede hacer polvo esas personalizaciones.

Esto debería comenzar a sonar familiar; es el mismo problema que motivó el enfoque de "depositar" archivos para la configuración de guiones de inicio de SysV. (Mencioné esta técnica en primer lugar en la sección "Guión de configuración del nivel de ejecución".) No debe resultar sorprendente, por tanto, que Red Hat se enfrentara a este problema de un modo similar: con el directorio /etc/profile.d. Las configuraciones globales de shell en los sistemas Red Hat Linux se construyen para leer cualquier archivo que se encuentre en /etc/profile.d como parte de su proceso de ejecución.

Los archivos de configuración global /etc/profile y /etc/bashrc (para bash), y /etc/cshrc y /etc/csh.login (para tcsh) son archivos muy simples. (En realidad, estos archivos son para sh y csh más que para bash y tcsh, pero la mayoría de las distribuciones Linux utilizan bash y tcsh como sus implementaciones de sh y csh.) Indican varios parámetros básicos como el indicador de comandos y algunos valores por defecto del sistema y después ejecutan cualquier archivo presente en /etc/profile.d con una determinada extensión. (La configuración de bash lee cualquier archivo que acabe en .sh, mientras que la de tcsh hace lo mismo con los archivos que cuyo nombre acaba en .csh.) Cada uno de estos archivos sirve para un propósi-

to específico. Por ejemplo, kde.sh configura KDE para los usuarios de bash, mientras que colorls.csh configura el programa ls en color para los usuarios de tcsh.

Siempre que se instala nuevo software en el sistema que necesita ser configurado en las shells de usuario, el administrador puede simplemente escribir dos guiones de shell (uno con la sintaxis de sh y otro con la sintaxis de csh, pero funcionalmente idénticos) que contengan la configuración necesaria, y después depositarlos en el directorio /etc/profile.d como archivos ejecutables. Las shells de usuario los leerán automáticamente en su siguiente entrada al sistema. Obviamente, este trabajo puede automatizarse en el interior de los paquetes RPM, haciendo muy sencillo el conectar nuevas instalaciones de software con el entorno shell de un usuario.

Red Hat Linux e Inetd

Existen muchos programas venerables en el mundo Unix, aclamados desde los más antiguos días de los primeros sistemas. Con la excepción de init y sh (la primera shell de comandos) quizás el más venerable de los programas de los sistemas Unix sea inetd, el superservidor de Internet. Esta sección describe xinetd, la implementación de inetd que utiliza Red Hat Linux 7.3.

Los lectores que ya se sientan familiarizados con los sistemas similares a Unix conocen con toda seguridad íntimamente el programa inetd. Este programa es el responsable de permanecer a la escucha en varios puertos TCP/IP y ejecutar procesos de servidor cuando se reciben conexiones de clientes. Los programas clientes que utilicen inetd pueden resultar un poco más simples de escribir que aquellos que se encargan de sus propias conexiones TCP/IP. El archivo de configuración para inetd ha sido tradicionalmente /etc/inetd.conf, y en esencia es una larga lista de puertos y comandos; inetd conecta a cada puerto el programa indicado y lo ejecuta cuando se recibe una conexión.

Versiones de Red Hat Linux anteriores a 7.3 utilizaban el programa inetd tradicional, con su típico archivo inetd.conf. Sin embargo, a partir de la versión 7.3, Red Hat Linux cambió a un programa conocido como xinetd, que, aunque sigue sirviendo para la misma función, es un poco distinto del inetd tradicional.

Nota

Otras distribuciones Linux, incluyendo a Debian y Slackware, aún utilizan el inetd tradicional. Si está interesado en esta implementación más antigua de inetd, lea el capítulo 5.

¿Por qué cambió Red Hat a xinetd? En una palabra, es más sencillo trabajar con xinetd. Las implementaciones tradicionales de inetd (incluyendo la que se suele utilizar en Linux) tienen varias restricciones (como un límite en el número de argumentos que se puede pasar a un programa) que hacen su uso algo inconveniente.

Además, el archivo /etc/inetd.conf tiene que modificarse para cada programa que desee utilizar inetd; esto lleva a los mismos problemas ya descritos para los guiones de inicio de SysV y para el entorno de shell del usuario.

La versión de inetd a la que se cambió Red Hat (xinetd) resuelve adecuadamente todos estos problemas. El directorio /etc/xinetd.d contiene archivos de configuración para los servicios inetd; este es otro ejemplo más de la aproximación de "dejar caer archivos de configuración" de Red Hat que mencioné con anterioridad en este capítulo al referirmeal guión de configuración del nivel de ejecución. Por otra parte, la sintaxis de estos archivos de configuración es mucho más rica que la soportada por el formato del antiguo inetd.conf, permitiendo más flexibilidad en la configuración de servicios.

xinetd también está soportado por el programa chkconfig mencionado con anterioridad. El programa chkconfig se utiliza para configurar qué servicios se habilitan para un determinado nivel de ejecución. Normalmente, chkconfig funciona sobre los guiones de inicio de SysV para habilitad o deshabilitar servicios para niveles de ejecución específicos. Sin embargo, chkconfig también soporta una funcionalidad similar para los servicios xinetd, con la misma sintaxis para habilitar y deshabilitar servicios. Esta integración hace de chkconfig una auténtica herramienta para todo de cara a la configuración de servicios, y fue más sencillo llevarla a cabo con el cambio a xinetd.

Dominio de los directorios de configuración de Red Hat

Hasta ahora ha visto el mecanismo de "depositar archivos de configuración" que utiliza Red Hat Linux en los guiones de inicio de SysV, el entorno de usuario y el superservidor de Internet xinetd. Hay, de hecho, otros ejemplos de esta técnica. Un simple vistazo al directorio /etc revela varios de estos ejemplos. La tabla 4.11 resume los ejemplos más importantes. Comprender el uso que hace Red Hat de esta técnica conducirá a una mejor comprensión de los sistemas Red Hat Linux.

Tabla 4.11. Directorios de configuración de Red Hat Linux.

Directorio	Paquete	Comportamiento
/etc/rc.d	initscripts	Todos los guiones en el directorio que se corresponde con un nivel de ejecución se ejecutan cuando se entra en ese nivel de ejecución.
/etc/profile.d	setup	Todos los guiones se ejecutan cuando un usuario abre una shell.
/etc/xinetd.d	xinetd	Cada archivo contiene información de configuración para un servicio en particular.

(Continúa)

Directorio	Paquete	Comportamiento
/etc/cron*	crontabs	Cada directorio contiene guiones de cron que se ejecutan con esa periodicidad (por ejemplo, /etc/cron.weekly contiene guiones que se ejecutan semanalmente [*week* significa en castellano "semana"]).
/etc/pam.d	pam	PAM son los módulos de autenticación enlazables (*Pluggable Authentication Modules*). Cada archivo configura los permisos de PAM para un programa o servicio en particular.
/etc/logrotate.d	logrotate	Cada archivo contiene parámetros de rotación de anotaciones para un determinado archivo de anotaciones o servicio.
/etc/makedev.d	forma parte de Red Hat	Cada archivo contiene información sobre la generación de una clase particular de archivos de dispositivo en /dev.
/etc/sysconfig	forma parte de Red Hat	Cada archivo contiene datos de configuración específicos del sistema para un aspecto en particular del sistema.

Cada uno de los directorios de la tabla 4.11 tiene el mismo contenido básico: cada uno contiene un conjunto de archivos que configuran individualmente alguna pequeña parte o subconjunto del paquete. Por ejemplo, los archivos individuales en el directorio /etc/pam.d configuran cada uno algún servicio para que funcione con el sistema PAM (Módulos de autenticación enlazables), mientras que el contenido de /etc/logrotate.d configura las políticas de rotado y limpieza de los archivos de anotaciones (*log files*) para un servicio particular. Incluso aunque cada directorio configure un servicio de alto nivel distinto, todos ellos utilizan la misma técnica de "depositar un archivo de configuración individual".

Trabajar con Kudzu

Kudzu es una enredadera importada por los Estados Unidos desde el otro lado del mar que rápidamente se ha vuelto omnipresente es la zona del Sudoeste del país. En busca de una "omnipresencia rápida" similar, Red Hat eligió "kudzu" como el nombre de su programa que se encarga de analizar el ordenador en busca de hardware nuevo y que configura el sistema si se detecta algún nuevo componente. Por supuesto, la planta kudzu también ataca a los árboles, bloqueando la luz solar que debería llegarles y causando su lenta muerte. Esta sección le ayudará a evitar que le suceda lo mismo a su sistema Red Hat Linux.

Generalmente, kudzu hace un buen trabajo de comprobación de hardware nuevo. Casi por definición, ningún analizador de hardware puede ser perfecto, pero

kudzu se ha ido volviendo gradualmente más fiable a lo largo de las últimas versiones de Red Hat Linux. Sin embargo, de vez en cuando, los usuarios se encuentran con una configuración particular de hardware que provoca que kudzu se comporte incorrectamente. Por fortuna, kudzu está configurado para ejecutarse como un servicio mediante los guiones de inicio de SysV. Si se encuentra en una situación en la que kudzu cause más mal que bien, puede deshabilitarlo mediante chkconfig: un rápido comando `chkconfig kudzu off` y ya no volverá a iniciar para el nivel de ejecución actual (por supuesto, si añade hardware nuevo, deberá configurarlo manualmente).

Si desea echarle un vistazo a los archivos que utiliza kudzu para realizar su trabajo, están instalados en el directorio /usr/share/kudzu. En general asigna las cadenas de caracteres que actúan como identificadores de los dispositivos hardware a módulos controladores de dispositivos del núcleo. Cuando se ejecuta, kudzu analiza el sistema en busca de nuevo hardware y configurar el sistema para que utilice cualquier módulo del núcleo que esté asignado al dispositivo. En teoría, podría enganchar el soporte para el nuevo hardware mediante la modificación de estos archivos, pero eso es definitivamente algo que sólo deberían intentar los desarrolladores.

Durante el arranque del sistema, la parte de kudzu que se encarga de la administración del sistema de archivos se ejecuta en /etc/rc.d/rc.sysinit, así que si eso es una causa de problemas, puede deshabilitarla al comentar la llamada a /usr/sbin/updfstab en ese archivo.

Instalar fuentes TrueType

Una de las características más esperadas de las versiones recientes de Red Hat Linux ha sido la inclusión del soporte en XFree86 para fuentes TrueType, que son omnipresentes en los sistemas Windows. Sin embargo, no es necesariamente obvio cómo se debe actuar para la instalación de estas fuentes en un sistema Red Hat Linux. Esta sección explica cómo se consigue.

Para instalar fuentes TrueType en Red Hat Linux 7.3, siga estos pasos:

1. Cree el directorio /usr/share/fonts/default/TrueType, si es que no existe previamente.
2. Coloque los archivos de las fuentes (.ttf) en /usr/share/fonts/default/TrueType.
3. Ejecute el comando `ttmkfdir > fonts.scale` en el directorio /usr/share/fonts/default/TrueType.
4. Ejecute el comando `mkfontdir` en el directorio /usr/share/fonts/default/TrueType.
5. (Este paso es opcional.) Modifique el orden del directorio de fuentes en /etc/X11/fs/config para que las fuentes TrueType aparezcan en primer lugar.

6. Reinicie el servidor de fuentes X (*X Font Server*, xfs), mediante el comando `service xfs restart`.

Las instrucciones precedentes describen como instalar las fuentes TrueType en la ubicación estándar. Realmente puede colocar estos archivos en cualquier directorio que le resulte conveniente; sin embargo, si los coloca en otro lugar que no sea /usr/share/fonts/default/TrueType, deberá modificar el archivo de configuración para el servidor de fuentes X (xfs) para que incluya el nuevo directorio. (El archivo de configuración para xfs es /etc/X11/fs/config.) También puede utilizar este archivo para modificar el orden en que se busca entre los directorios de fuentes cuando se trata de localizar el nombre de una fuente. Esto permite a los usuarios otorgar prioridad a las fuentes TrueType sobre otros tipos por si acaso tienen el mismo nombre. No obstante, debería de señalarse que no todos los programas utilizan los servicios de fuentes de X Window; algunos (como TeX, ghostscript, y la aplicación OpenOffice) utilizan sus propios mecanismos. Este tipo de programas aún no son capaces de utilizar las fuentes TrueType, incluso una vez que han sido instaladas.

Garantizar la seguridad del sistema

Una de las mayores preocupaciones sobre las versiones antiguas de Red Hat Linux fue el comparativamente débil énfasis hecho por la empresa en cuanto a la seguridad. En el pasado, los sistemas Linux han sido tradicionalmente el parque de juegos de los expertos técnicos. Las diversas distribuciones generalmente tomaban la postura de ofrecer la máxima funcionalidad y dejaban a los usuarios la tarea de particularizar y proteger los sistemas. Sin embargo, a medida que la penetración en el mercado de los sistemas Linux se incrementa, también lo hace el número de usuarios menos experimentados, que pueden no ser capaces de proteger sus sistemas, o incluso peor, ¡que podrían no saber que sus sistemas necesitan ser protegidos! Por eso, al igual que la mayoría de los fabricantes, Red Hat está adquiriendo gradualmente más conciencia de lo que supone la seguridad en sus distribuciones.

Uso de un cortafuego

Las versiones recientes de Red Hat Linux ya no dejan todos los servicios inetd estándar habilitados, y la versión 7.3 incluye un cortafuego por defecto. Un cortafuego incrementa la seguridad al impedir que los atacantes accedan a cualquier servidor potencialmente inseguro que esté ejecutándose, tanto intencionada como inadvertidamente. El proceso de instalación presenta un indicador al usuario para que seleccione un nivel de seguridad para el cortafuego y le permite excluir de esa protección puertos específicos para aplicaciones específicas (como el protocolo SSH para acceso remoto). Este cortafuego puede configurarse después de que el sistema se encuentre en funcionamiento mediante el comando `lokkit`. (Los cortafuegos de Red Hat Linux se basan actualmente en el sistema ipchains desarrollado en la antigua serie 2.2 del núcleo; Red Hat aún no ha adaptado la configuración al más

reciente sistema iptables creado para la serie 2.4, a pesar de que eso no supone una pérdida de prestaciones.)

> **Nota**
>
> Una configuración de servidor cortafuego para una red real es uno de los casos de estudio de este libro. Véase el capítulo 16 para obtener una información mucho más detallada.

Gestión de valores de seguridad con PAM

Mientras que los cortafuegos ayudan a proteger al sistema de atacantes externos, no hacen demasiado para impedir a los usuarios que ya tienen cuentas en el sistema ataquen las cuentas de otros usuarios o del superusuario. Este tipo de seguridad se lleva a cabo en los sistemas Red Hat Linux (como en la mayor parte de otras distribuciones) mediante el paquete de Módulos de Autenticación Enlazables (*Pluggable Authentication Modules*, PAM).

PAM ofrece un mecanismo extensible, enlazable y transparente para añadir métodos de autenticación contra diversas fuentes. Por ejemplo, existen módulos PAM para llevar a cabo la autenticación contra servidores LDAP y RADIUS tan fácilmente como contra el tradicional mecanismo Unix de /etc/passwd. PAM también permite a determinados servicios y programas que tengan requisitos específicos de cara a la autenticación. Finalmente, PAM también permite a los administradores configurar cosas como los periodos del día en que los usuarios pueden acceder a un sistema.

PAM se configura mediante los directorios /etc/pam.d y /etc/security. El primer directorio contiene parámetros de seguridad específicos para una aplicación. Tal y como se muestra en la tabla 4.11, este directorio utiliza el esquema típico de Red Hat de "depositar archivos": cada archivo en el directorio /etc/pam.d contiene información de configuración de seguridad para un programa en particular. El segundo directorio contiene opciones de configuración más globales, que controlan el modo en que a los usuarios se les concede permisos a diversos recursos y cuándo pueden entrar en el sistema.

Los archivos con los que es más probable que interactúen los usuarios finales son los archivos /etc/security/console.perms y los contenidos en /etc/pam.d. El archivo console.perms especifica a qué dispositivos tienen acceso los usuarios cuando entran en el sistema a través de una consola en particular. Por ejemplo, a los usuarios se les permite el acceso de lectura y escritura al dispositivo /dev/floppy (que representa a la disquetera) cuando entran en el sistema tanto desde la consola como a través de X, pero no cuando acceden al sistema remotamente (ya que, en ese caso, no estarían sentados ante la máquina y no necesitarían utilizar la disquetera). Los archivos que están en /etc/pam.d, mientras tanto, controlan el acceso a programas específicos, como el servidor HTTP Apache: los usuarios que instalan software

que tiene requisitos de seguridad específicos o no estándar podrían necesitar crear o modificar archivos en este directorio. La mayor parte del tiempo los usuarios no necesitarán modificar estos archivos. Sin embargo, es importante saber que existen de cara a una futura utilización. Cuando llegue el momento, los usuarios pueden consultar la documentación en busca de información más detallada.

Derivados de Red Hat Linux

Red Hat no es el único habitante del mundo Linux. Sin embargo, Red Hat Linux es una de las distribuciones más populares y la empresa ha supuesto un tremendo impacto sobre la comunidad de código abierto en general y sobre las distribuciones Linux en particular. Muchas distribuciones derivan de Red Hat Linux; algunas son "imitaciones" directas y otras simplemente utilizar el mismo formato de empaquetado. Esta sección proporciona una lista de algunos derivados populares o "primos" de Red Hat Linux.

Mandrake

Mandrake es una distribución Linux producida por MandrakeSoft. Mandrake se centra en el mercado de escritorio y ofrece algún apoyo financiero al proyecto KDE del mismo modo en que Red Hat apoya al proyecto GNOME. Mandrake se basó originalmente en una versión de Red Hat Linux pero desde entonces ha entrado en un camino divergente. Mandrake aún le sigue la pista a Red Hat Linux, pero muy ligeramente, para mantener una correspondencia característica a característica.

Mandrake generalmente se abstiene de usar la mayoría del software desarrollado por Red Hat (como kudzu y el instalador anaconda) a favor de sus propias versiones.

Mandrake también incluye herramientas adicionales orientadas al modo en que el usuario interactúa con el sistema; por ejemplo, Mandrake incluye un programa "supermount" que tiene como objetivo hacer transparente para el usuario el montaje de diversos dispositivos. Finalmente, Mandrake también incluye una amplia variedad de software, incluyendo una buena porción de software de calidad beta o alpha.

Caldera

La distribución OpenLinux de Caldera se dirige sin dudarlo al mercado corporativo. Caldera no es tan riguroso acerca de mantenerse estrictamente como código abierto como Red Hat, y por eso Caldera incluye una serie de paquetes de software con marca comercial (de código no abierto) con su distribución. Caldera también ofrece soporte y servicios profesionales para su distribución. Caldera tiene otras creaciones similares a Unix además de su distribución Linux; notablemente, Calde-

ra posee ahora UnixWare, que es el único descendiente directo del SysV Unix original de AT&T. OpenLinux de Caldera se basó originalmente en una versión antigua de Red Hat Linux, pero hace mucho tiempo que sus caminos se separaron.

TurboLinux

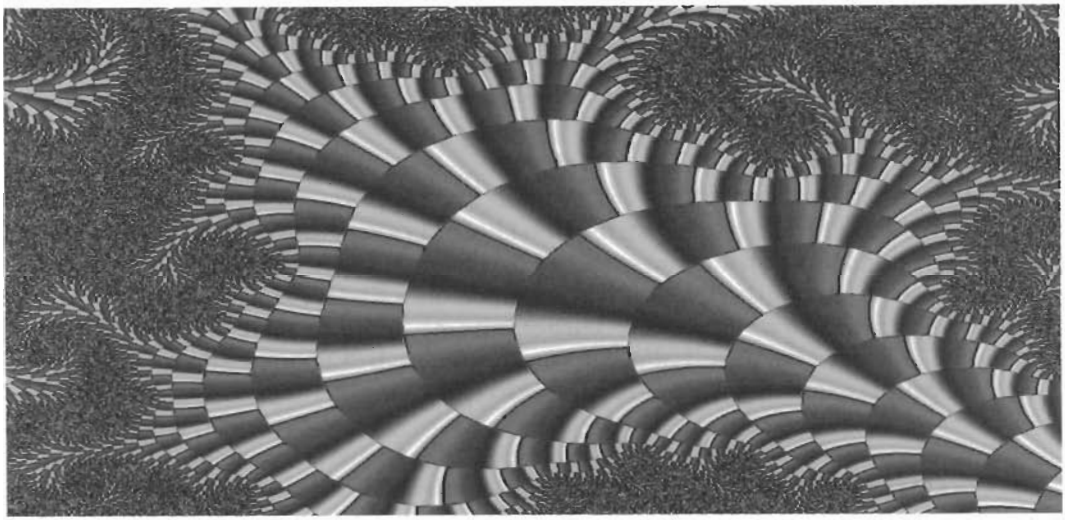
TurboLinux produce una distribución enfocada a servidores agrupados y de alta disponibilidad. Su distribución se utiliza típicamente en las grandes máquinas "mastica-números" paralelas Beowulf o en agrupaciones de servidores para sitios web o para servidores de aplicaciones. TurboLinux no sigue la pista del último y más potente software tan agresivamente como otras distribuciones, sino que se centra en sus propias soluciones de agrupamiento (*clustering*).

Resumen

Este capítulo trató acerca de los detalles de Red Hat Linux, centrándose en la versión 7.3. Red Hat es una de las tres importantes distribuciones Linux que los usuarios encontrarán. Hay otras distribuciones que se derivan de Red Hat Linux, pero en estos casos el conocimiento del propio Red Hat Linux resulta útil para trabajar con sus derivados.

Una vez leído este capítulo, se encuentra bien preparado para abordar cualquier sistema Red Hat Linux. Ahora es capaz de encontrar los archivos y programas que necesita utilizar, de instalar y administrar paquetes de software, y de personalizar el sistema para sus propias necesidades. Puede instalar un núcleo estándar, personalizar el inicio del sistema, ajustar el entorno de usuario o personalizar otros aspectos como la configuración del cortafuego. También tiene ahora una cierta familiaridad con RPM y con el conjunto de software incluido en la distribución. Probablemente lo más inmediatamente útil que ha aprendido es la técnica canónica de configuración de Red Hat "depositando" archivos de configuración.

Ya que ningún único libro puede incluir cada detalle particular sobre un sistema tan sofisticado como Red Hat Linux, quizás lo más importante que haya aprendido sea la habilidad de desplazarse en un sistema Red Hat Linux. Es muy probable que algún día quiera hacer algo con su sistema que no esté cubierto en este libro; por fortuna, ahora sabe cómo comenzar y dónde buscar la información necesaria. Los restantes dos capítulos de esta parte le proporcionarán un nivel de dominio similar sobre los sistemas Debian y Slackware.



5
Slackware Linux 8.0

Slackware Linux es una de las distribuciones más antiguas que aún se sigue desarrollando activamente. La creó en 1993 Patrick Volkerding, convirtiéndola en una antecesora de las otras dos distribuciones de muestra tratadas en este libro, Red Hat Linux y Debian. Debido a esta longevidad (al igual que a la fiable calidad de la distribución), Slackware suele considerarse una de las distribuciones Linux más robustas.

Después de la lectura de este capítulo habrá adquirido una comprensión práctica acerca de un sistema Slackware.

Tras la lectura adicional de los capítulos 4 y 6, habrá visto tres distribuciones Linux diferentes, y deberá haber adquirido una perspectiva bastante significativa sobre el funcionamiento de cualquier sistema Linux, al igual que sobre su potencia y flexibilidad.

Este capítulo tratará acerca de Slackware Linux 8.0 (a veces referido simplemente como Slackware) en detalle.

Para obtener una idea básica de lo que va asociado a una distribución, los lectores deberían revistar el capítulo 3. Además, gran parte de este capítulo se basa en mostrar características de Slackware en comparación con las características equivalentes en Red Hat Linux; esto es, este capítulo se apoya en parte en el capítulo 4 mediante la comparación entre las dos distribuciones, así que puede serle útil revisar también el capítulo 4.

Es importante que recuerde a lo largo de su lectura que estas comparaciones no tratan de dirimir si una distribución es superior a otra o no, simplemente pretenden mostrar técnicas diferentes.

Antecedentes y filosofía de Slackware

El sitio web de Slackware Linux (www.slackware.com) tiene una sección sobre "La filosofía Slackware" ("*Slackware Philosophy*"). No puede haber un resumen mejor de la filosofía de un proyecto que la que explícitamente hacen pública, y por eso reproducimos aquí sus propias palabras extraídas de su sitio web:

"Desde su primer lanzamiento en abril de1993, el proyecto Slackware Linux ha intentado producir la distribución Linux más "similar a Unix" que exista. Slackware cumple los estándares publicados de Linux, como el estándar de sistema de archivos de Linux. Siempre hemos considerado primordiales la simplicidad y la estabilidad, y como resultado, Slackware se ha convertido en una de las distribuciones más populares, estables y amigables disponibles."

Una frase importante es el término "similar a Unix". Muchas de las distribuciones actuales de Linux se centran agresivamente en la adición de prestaciones y características, esencialmente modernizando Unix, en su encarnación como sistema Linux. Sin embargo, tradicionalmente ha existido una especie de "misticismo hacker" alrededor de Unix, que se refleja en una filosofía estricta de la simplicidad y la autosuficiencia. (Lo crea o no, es sencillo utilizar las características avanzadas de Unix una vez que se comprende lo básico.) En la loca carrera hacia la modernización, podría argumentarse que gran parte de esta filosofía tradicional de Unix se está perdiendo. Debido a esto, el objetivo de Slackware de ser similar a Unix dice mucho sobre su filosofía. Los usuarios de otros sistemas Unix pueden encontrar en Slackware la distribución Linux a la que es más sencillo migrar, porque es como Unix. Esta filosofía se manifiesta típicamente en la práctica mediante la elección de métodos simples y estándar para realizar tareas, en lugar de reinventar la rueda. Por ejemplo, Slackware utiliza el tradicional formato comprimido de archivos *tarball* (nombre procedente de *Tape ARchive*), que es simplemente un archivo .tar.gz) como mecanismo de empaquetado en vez de utilizar un sistema de administración de paquetes mejorado como RPM o el sistema de Debian. La utilización de Slackware de archivos *tarball* estándar hace posible administrar todo el software a mano, pero sacrifica algunas características más sofisticadas, como el cumplimiento de las dependencias y la resolución de conflictos entre archivos.

Por un lado, Slackware carece de muchas de las características más nuevas que muchas otras distribuciones han presentado. Por otra parte, es posible administrar un sistema Slackware al completo con nada más que un editor de textos y un poco de conocimiento práctico. Este tipo de autosuficiencia atrae a muchos usuarios experimentados de Unix y Linux. También convierte a Slackware en un buen "entrenamiento" para los nuevos usuarios (si puede hacer algo en Slackware, puede hacerlo en cualquier otro lugar). Muchos usuarios experimentados de Unix y Linux (incluyéndome a mí mismo) aprendieron casi todo lo básico en un sistema Slackware. Por estos motivos, Slackware mantiene un cambiante séquito de usuarios (en ocasiones con bastante mal genio) que contemplan la distribución con ojos afectuosos.

Núcleo Linux

Uno de los objetivos establecidos por Slackware es la simplicidad y la estabilidad. Distribuir un núcleo personalizado introduciría un matiz diferenciador adicional en el sistema del que los administradores (y en ocasiones los usuarios) deberían ser conscientes y que tendrían que manejar. Por ello, Slackware no utiliza o distribuye núcleos particularizados, como sí hacen algunas otras distribuciones. No obstante, ocasionalmente Slackware incluirá un núcleo con un parche que corrige un error crucial que aún no se ha incorporado en el núcleo estable oficial.

Como Slackware solamente confía en núcleos estándar, es realmente muy sencillo actualizar el núcleo en un sistema Slackware. El código fuente para el núcleo se coloca en /usr/src; esta es la misma ubicación que en el caso de Red Hat Linux. Sin embargo, a diferencia de los núcleos de Red Hat, no hay que realizar ninguna acrobacia cuando se pretende actualizar el núcleo; únicamente se elimina el enlace simbólico con Linux, se extrae el nuevo núcleo, y se recrea el enlace simbólico de modo que apunte al directorio del nuevo núcleo. Después de eso, simplemente se construye un núcleo normalmente.

Sin embargo, hay un modo significativo en que Slackware se diferencia de Red Hat en la administración del núcleo. Como se mencionó brevemente en el capítulo 4, la biblioteca de sistema glibc de GNU confía en determinadas cabeceras del núcleo de Linux.

Estas cabeceras deberían corresponderse con la versión del núcleo contra la que se ha compilado glibc, en lugar de reflejar el núcleo que se ejecuta actualmente. (De hecho, Linus Torvalds y otros desarrolladores del núcleo preferirían que glibc eliminara esta dependencia de las cabeceras del núcleo Linux.) Red Hat solventa este problema ofreciendo un paquete RPM especial llamado *kernel-headers* (con las cabeceras del núcleo).

Sin embargo, Slackware trata esto mediante la creación de un enlace simbólico en /usr/include/linux que apunta a la instalación del código fuente del núcleo. En esencia esto significa que Slackware necesita que el código fuente del núcleo esté instalado en el sistema para poder utilizar los compiladores, y también significa que las cabeceras del núcleo cambiarán con la versión actual del núcleo. La solución de Red Hat es probablemente más "correcta", pero la de Slackware es verdaderamente más sencilla, mostrando una vez más las diferencias entre las filosofías de las dos distribuciones.

En todo caso, la distribución Slackware está enfocada hacia la estabilidad y la simplicidad. Por ello, a veces Slackware se rezaga tras otras distribuciones en la adopción de lanzamientos nuevos e importantes de componentes clave como el núcleo y las bibliotecas de sistema. Slackware 8.0, por ejemplo, incluye dos núcleos: el núcleo 2.4.5 actual y el antiguo (pero suficientemente probado) núcleo 2.2.19. Incluso aunque ambos núcleos sean considerados estables, Slackware 8.0 utiliza por defecto el núcleo 2.2, porque está muy más probado y libre de fallos (y por ello es más digno de confianza).

Formato de empaquetado

Slackware utiliza archivos *tarball* estándar (archivos con la extensión .tar.gz) como su formato de empaquetado. Como este tipo de archivos puede leerse en cualquier sistema Unix (y en cualquier distribución Linux), son independientes de la plataforma (aunque, por supuesto, sus contenidos típicamente no lo son). Esta sección tratará acerca de cómo maneja Slackware Linux 8.0 la administración de paquetes mediante estos archivos.

Resumen técnico

Los archivos TAR no están comprimidos, así que frecuentemente hay comprimirlos antes de distribuirlos. En los primeros días de Unix, el formato de compresión habitual era el común programa de Unix compress, que dotaba a los archivos de la extensión .Z. El proyecto GNU introdujo el formato de compresión Zip de GNU (`gzip`), con la extensión .gz. Más adelante, el programa de compresión bzip introdujo la extensión .bz2. Un archivo *tarball* es simplemente un archivo TAR comprimido siguiendo uno de estos esquemas; las extensiones terminan siendo .tar.Z, .tar.gz o .tar.bz2. El formato más común para Linux y otro software de código abierto es un archivo TAR comprimido con gzip (*gzipped*); a veces la extensión se reduce como .tgz. Todos estos tipos de archivo (es decir, los simples archivos .tar tanto como los .tar.gz, etc...) se refieren en ocasiones como *tarballs*; un *tarball* puede estar comprimido o no.

Administración de paquetes instalados

Al igual que Slackware apunta hacia la simplicidad con el uso de archivos `tarball` como el formato nativo para el gestor de paquetes, apunta hacia una simplicidad similar en las herramientas que se encargan de la administración de paquetes.

Esto es sobretodo evidente en el caso de la herramienta de administración de paquetes, conocida simplemente como pkgtool, que es un programa con un menú de consola (textual). En realidad no es mucho más que un envoltorio amigable con el usuario para la ejecución de otras herramientas que se describirán más adelante. El menú de pkgtool ofrece varias opciones de instalación, una opción para eliminar un paquete instalado y una opción para ver los contenidos de paquetes que estén instalados actualmente.

Puede instalar paquetes a partir de las tres fuentes siguientes:

- El directorio actual
- Otro directorio (arbitrario)
- Un disquete

Cada una de las opciones anteriores permite a los administradores instalar múltiples paquetes al mismo tiempo. Este es realmente el modo en que los paquetes se instalan cuando se instala el propio sistema en primer lugar. Estas son las acciones que pkgtool realiza cuando instala múltiples paquetes:

1. El programa de instalación de Slackware inicia la herramienta pkgtool para instalar los paquetes seleccionados por el usuario. (Este paso sólo se realiza durante la instalación del sistema, es un paso que se salta si ejecuta pkgtool manualmente.) Este paso prepara el directorio que pkgtool usará en el paso 2.
2. pkgtool analiza su directorio de entrada en busca de todos los archivos que pueda encontrar que acaben en .tgz o .tar.gz, es decir, busca cualquier archivo *tarball* estándar.
3. Después, pkgtool pregunta al usuario si se debería instalar cada archivo, y actúa de acuerdo con la respuesta del usuario.

Las otras opciones que soporta pkgtool permiten al usuario eliminar un paquete tras seleccionarlo en una lista, y también permiten al usuario ver una lista con todos los paquetes instalados, al igual que sus contenidos. Esta funcionalidad es sólo un bello envoltorio que enmascara las otras herramientas que se describirán en las siguientes secciones. Las herramientas siempre se pueden invocar desde la línea de comandos, pero el comando `pkgtool` ofrece una interfaz de usuario consolidada.

Instalación de paquetes

Los paquetes de Slackware se pueden instalar mediante la línea de comandos con el programa `installpkg`. Este programa toma un único parámetro (el nombre de un archivo *tarball*) para instalarlo y dispone de varias opciones que ajustan dónde y cómo se instalará el paquete. El comando es muy simple, así que no repetiremos todas las opciones. Sin embargo, la opción más útil es la opción `-root <ruta>`. Este parámetro cambia la raíz de la instalación por `<ruta>`; por ejemplo, un paquete que normalmente se instalaría en el directorio raíz puede colocarse en algún otro sitio, como /opt mediante el comando `installpkg -root /opt nombre_de_paquete.tgz`.

El comando `installpkg` tiene opciones adicionales que puede utilizar que proporcionan diversas capacidades de menú interactivo; estas características se utilizan típicamente cuando se arranca `installpkg` desde otro programa o guión, como el comando envoltorio `pkgtool`.

Actualización de paquetes

Los paquetes de Slackware se pueden actualizar mediante el comando `upgradepkg`. Este comando es más simple incluso que el comando `installpkg`. Si el nuevo paquete tiene el mismo nombre que un paquete ya instalado, entonces el

comando `upgradepkg <nombre_de_fichero>` actualizará el paquete. Si los nombres de los archivos son diferentes, entonces el comando `upgradepkg <antiguo_nombre_de_paquete>%<nuevo_nombre_de_archivo_de_paquete>` actualizará el paquete. Por ejemplo, si foo-2.0.tgz es una actualización de foo-1.0.tgz, entonces el comando `upgradepkg foo-1.0%foo-2.0.tgz` llevará a cabo la actualización. En cualquier caso, el efecto es el mismo: la herramienta consulta la lista de archivos que instaló el paquete antiguo, y después instala el paquete nuevo (actualizado). Si hay algún archivo en el paquete original que no reemplazó el paquete de actualización, entonces estos archivos viejos se eliminan. Si un paquete necesita actualizarse en un directorio raíz distinto, entonces la variable de entorno ROOT debe prepararse para conseguir ese efecto (en contraste con el comando `installpkg`, que acepta esta posibilidad como una opción en la línea de comandos). No olvide que el paquete tendrá que actualizarse en el mismo directorio raíz en que instaló, o sino no será realmente una actualización.

Desinstalación de paquetes

El comando utilizado para desinstalar un paquete es `removepkg`. Este comando toma un único argumento, que es el nombre del paquete que se quiere eliminar. Al igual que con `updgradepkg`, la raíz del paquete puede modificarse dando el valor adecuado a la variable de entorno ROOT. Por supuesto, el directorio raíz debe corresponder al directorio en que el paquete está realmente instalado. Para eliminar el paquete, la herramienta recupera la lista de archivos instalados por el paquete y luego simplemente los elimina.

Creación de nuevos paquetes

El sistema de administración de paquetes de Slackware también ofrece un medio muy sencillo de crear paquetes compatibles con Slackware. Un desarrollador administrador, u otro empaquetador, simplemente crea la estructura de directorios del software que se va a empaquetar en un directorio temporal que sirve de esquema. Por ejemplo, un administrador podría compilar un paquete de software y colocarlo en /tmp. Hay características avanzadas adicionales que se pueden aprovechar cuando se trata de la creación de paquetes para Slackware. El comando `makepkg` resulta útil para muchas o casi todas las situaciones, pero si se necesitan prestaciones más extensas (como un guión de instalación personalizada), se puede aprovechar. Los usuarios interesados pueden consultar la documentación de Slackware en www.slackware.com.

Mecanismo frente a interfaz de usuario

Las herramientas de administración de paquetes de Slackware son bastante básicas cuando se comparan con las herramientas RPM de Red Hat (tratadas en el

capítulo 4) o la herramienta apt de Debian (tratada en el capítulo 6). No obstante, incluso aunque las herramientas de Slackware podrían no tener el suficiente nivel de sofisticación entre las distribuciones Linux, aún son al menos tan sofisticadas como la mayoría de otras herramientas de sistemas Unix.

Es importante recordar la filosofía de Slackware sobre la simplicidad y la facilidad de mantenimiento. No hay literalmente nada que estas herramientas hagan que no pueda hacerse manualmente mediante un editor de textos. Cuando se instala un paquete, un archivo que contiene la descripción del paquete y una lista de todos sus archivos se crea en el directorio /var/log/packages. Estos archivos son legibles por humanos, que quiere decir que se pueden ver (o incluso acceder a ellos mediante varios guiones si es que un administrador se siente particularmente inteligente). Por ejemplo, se puede ejecutar el comando `grep`, en busca de la cadena de caracteres "etc" en el directorio /var/log/packages para encontrar la lista de todos los archivos dentro del directorio /etc/ que pertenezcan a paquetes; en Red Hat esto no se hace de un modo tan sencillo.

Las herramientas comentadas en esta sección utilizan estos mismos archivos para hacer su trabajo. De hecho, ¡las propias herramientas son simples guiones de shell! El mecanismo de administración de paquetes de Slackware es realmente un atajo para cosas que un administrador puede realizar manualmente. Que este hecho no sea inmediatamente obvio a los usuarios de las herramientas es un homenaje al éxito de la simplicidad elegante de Slackware, y una gran demostración de ese "misticismo hacker" tradicional de Unix de la autosuficiencia y la simplicidad.

Estructura del sistema de archivos

En este punto, la importancia de entender cómo desplazarse por la estructura de archivos de un sistema operativo debería estar clara (si no se puede encontrar algo, no se puede utilizar). En el capítulo 3 se comentó el Estándar de Jerarquía del Sistema de Archivos (FHS); en el capítulo 4 se comentó también el grado de adherencia de Red Hat al FHS. En general, no hay mucho que decir sobre Slackware, excepto, si acaso, que obedece aún más estrictamente que Red Hat el FHS. Esto es, puede no esperar sorpresas cuando busque un archivo en un sistema Slackware.

Aviso

Ya que las herramientas de empaquetado de Slackware soportan la instalación de simples archivos *tarball*, los administradores deberían asegurarse de que saben en qué lugar del sistema de archivos se instalan los paquetes que no son de Slackware. No todo el mundo sigue el FHS.

Lo único destacable sobre el uso que hace Slackware del sistema de archivos es cómo trata el directorio /opt. Recuerde del capítulo 3 que /opt está destinado a

utilizarse para componentes del sistema "opcionales". Red Hat trata a este directorio más o menos como a /usr/local, en cuanto que normalmente los paquetes RPM no colocan nada en /usr/local o /opt (y esta convención se respecta con bastante consistencia en general). Esos dos directorios están reservados para su uso por parte de un administrador de sistema para que coloque archivos locales.

Slackware también trata a /usr/local como un directorio estrictamente para administradores; sin embargo, Slackware realmente coloca algún software en /opt. Específicamente, Slackware coloca los paquetes opcionales en /opt (por lo que obviamente es difícil culpar a nadie). Por ejemplo, ni KDE ni GNOME se necesitan para el funcionamiento normal del sistema ni para que funcione XFree86; por ello Slackware instala estos paquetes en /opt. En cierto modo, puede decir realmente que al colocar los paquetes opcionales en /opt, Slackware realiza el trabajo de repartir lógicamente su instalación de software bastante mejor que Red Hat. Al final, es algo que da lo mismo, así que lo que los usuarios deberían extraer de estos comentarios es que si un paquete en particular no se encuentra dentro de /usr, es probable que esté dentro de /opt.

Guiones de arranque del sistema

En el capítulo 3 se mencionaron brevemente los dos modelos comunes de guiones de arranque del sistema (el modelo BSD y el SysV). En el capítulo 4 se comentó en detalle el modelo de Red Hat Linux, que se basa en el de SysV,. Ese capítulo contiene suficiente información como para adquirir un importante nivel de competencia en los mecanismos de arranque de cualquier sistema basado en SysV. Este capítulo comentará, en cambio, el modelo de guiones de arranque del sistema de Slackware. Ya que Slackware es bastante similar a BSD, este capítulo proporcionará información detallada sobre el modelo BSD de guiones de inicio. Finalmente, este capítulo asume que ya ha leído la sección " Guiones de arranque del sistema " del capítulo 4 y se basa en comparaciones con ese material.

Estructura de archivos y directorios

El primer paso hacia la comprensión del modelo de guiones de arranque de BSD utilizado por Slackware es mostrar una lista de los archivos relevantes. Es importante recordar que, fundamentalmente, tanto el modelo SysV utilizado por Red Hat como el modelo BSD que utiliza Slackware hacen las mismas cosas. Sólo la estructura de los guiones varía de uno a otro. Esta sección bosquejará los siguientes archivos utilizados por el modelo BSD y comentará cómo se comparan con sus contrapartidas de SysV:

- /etc/inittab
- /etc/rc.d/rc.S

- /etc/rc.d/rc.K
- /etc/rc.d/rc.M
- /etc/rc.d/rc.0 y /etc/rc.d/rc.6
- /etc/rc.d/rc.modules
- /etc/rc.d/rc.inet1 y /etc/rc.d/rc.inet2
- /etc/rc.d/rc.sysvinit

El directorio principal que contiene a todos estos archivos es /etc/rc.d, al igual que en el modelo SysV.

/etc/inittab

Recuerde que el propósito último de los guiones de inicio es configurar los diversos servicios que debe poner en marchael proceso padre global inicial. Como tanto Slackware como Red Hat utilizan la misma implementación del programa básico init, comparten el mismo punto de partida, que es el archivo /etc/inittab (la tabla de inicio, "*init table*").

Como se comentó en el capítulo 4, /etc/inittab contiene líneas que especifican programas (normalmente guiones de shell) que deberían ejecutarse cuando init entre en un determinado nivel de ejecución, o cuando sucedan otros eventos diversos (como un fallo de energía). El programa init permite que existan siete niveles de ejecución, numerados de 0 a 6.

Un sistema Unix determinado puede asignar los significados que desee a estos niveles de ejecución, pero tradicionalmente el nivel 0 se corresponde con la parada del sistema y el nivel 6 con el reinicio. Los demás niveles de ejecución dependen del sistema. La tabla 5.1 resume el significado que Slackware asigna a cada nivel de ejecución.

Tabla 5.1. Niveles de ejecución de Slackware Linux.

Nivel de ejecución	Significado
0	Parada del sistema.
1	Modo monousuario (sólo para el superusuario; típicamente utilizado para mantenimiento del sistema).
2	Sin uso, equivalente al nivel de ejecución 3.
3	Modo multiusuario estándar (sin X Window).
4	Modo multiusuario con escritorio X (entrada en el sistema mediante la interfaz gráfica).
5	Sin uso, equivalente al nivel de ejecución 3.
6	Reinicio.

Muchas de las líneas del archivo /etc/inittab estándar para Slackware son las mismas o muy parecidas a las líneas equivalentes en la versión de Red Hat. Como esas líneas ya se comentaron en el capítulo 4, no se volverán a comentar aquí. Sin embargo, hay algunas líneas que son distintas, y estas líneas son las que serán analizadas en detalle. Aunque, en primer lugar, es importante comentar algunas generalidades.

Un modo de definir el modelo de SysV de guiones de inicio es como un marco de guiones en el que se pueden enganchar servicios. En esencia, cada nivel de ejecución que aparece en /etc/inittab está configurado para invocar el mismo guión y es ese guión el que hace el trabajo de iniciar y detener los servicios apropiados para el nuevo nivel de ejecución. El modelo BSD, por su parte, no tiene la misma clase de extenso marco de guiones. En su lugar, el modelo BSD define un guión para cada nivel de ejecución por separado, y /etc/inittab conecta a cada guión con su correspondiente nivel de ejecución. Esto es lo que sucede en la mayor parte de las líneas del archivo /etc/inittab de Slackware que difieren de la versión de Red Hat. Aquí están las líneas relevantes:

```
si:S:sysinit:/etc/rc.d/rc.S
su:1S:wait:/etc/rc.d/rc.K
rc:2345:wait:/etc/rc.d/rc.M
```

La primera línea define el archivo /etc/rc.d/rc.S como el guión que se debe ejecutar cuando al iniciar el sistema, sin tener en cuenta el nivel de ejecución; la segunda línea define /etc/rc.d/rc.K como el guión que se ejecutará cuando se entre en el nivel de ejecución1 (que es el modo monousuario); la tercera línea define /etc/rc.d/rc.M como el guión que se ejecutará para todos los modos multiusuario. Estos son los tres guiones claves que utiliza Slackware para configurar el sistema.

Más adelante en el archivo, aparece esta serie de líneas:

```
c1:1235:respawn:/sbin/agetty 38400 tty1 linux
c2:1235:respawn:/sbin/agetty 38400 tty2 linux
c3:1235:respawn:/sbin/agetty 38400 tty3 linux
c4:1235:respawn:/sbin/agetty 38400 tty4 linux
c5:1235:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux
x1:4:wait:/etc/rc.d/rc.4
```

Estas líneas definen seis consolas virtuales de texto que se ejecutarán para todos los niveles de ejecución del modo multiusuario, excepto el nivel 4 (esto es, para los niveles 2, 3 y 5). El nivel de ejecución 4 es el que incluye las X; la última línea de código ejecuta X mediante el guión /etc/rc.d/rc.4. Recuerde que el nivel "multiusuario con X" de Red Hat es el 5. Por ello, el nivel de ejecución 5 en Red Hat Linux es más o menos equivalente al nivel 4 en Slackware Linux.

Hay una pequeña diferencia: bajo Slackware, sólo una de las consolas (tty6) está configurada para el nivel de ejecución 4, mientras que bajo Red Hat Linux están presentes las seis consolas y X simplemente se añade en la séptima consola. El motivo es que Slackware trata el nivel de ejecución 4 de un modo ligeramente diferente al que Red Hat trata su nivel de ejecución 5. Mientras que Red Hat deja las

seis consolas virtuales en ejecución y simplemente añade X en la séptima consola, Slackware inhabilita todas excepto una de las consolas de texto. Esto es, por supuesto, una simple cuestión "administrativa", pero hay una sutil diferencia: Slackware considera el nivel 4 como un "modo de uso" diferente, mientras que Red Hat ve su nivel 5 como la simple ejecución de X por conveniencia.

Nota

En cualquier distribución, siempre puede arrancar X automáticamente mediante el procedimiento de entrar en el sistema mediante una de las consolas de texto virtuales y utilizando después el comando xinit, así que al fin y al cabo iniciar X automáticamente es en realidad sólo algo conveniente.

/etc/rc.d/rc.S

El archivo /etc/rc.d/rc.S es el guión sysinit que realiza un trabajo muy básico, como activar el espacio de intercambio de memoria, comprobar las particiones de los discos locales en busca de errores y montar sistemas de archivos que no formen parte de una red. También configura hardware como el reloj del sistema y el generador de números aleatorios o los módulos del núcleo para los controladores de dispositivos (mediante la invocación del archivo rc.modules, que se describirá más adelante) y los dispositivos PCMCIA. También configura el hardware conectado a través del puerto serie mediante el guión rc.serial. Este archivo es de un nivel bastante bajo y generalmente muy sencillo; es probable que no tenga que modificarlo demasiado. Lo importante que debe comprender es que como realiza tareas muy básicas, este archivo está configurado para ser invocado sin importar el nivel de ejecución que se va a utilizar. Puede que merezca la pena analizar el archivo para captar la idea de los tipos de tareas que realiza.

/etc/rc.d/rc.K

El archivo /etc/rc.d/rc.K es el guión que se ejecuta cuando el sistema entra en el modo monousuario (que corresponde al nivel de ejecución 1). La "K" es una abreviatura de "*kill*" (matar), porque este guión mata procesos. En primer lugar finaliza todos los procesos que se estén ejecutando actualmente y después simplemente conmuta al nivel de ejecución 1. Por esa razón, los únicos procesos que iniciarán en el nivel 1 son las consolas tty definidas en /etc/inittab. Este estado obviamente no será demasiado útil (ya que no se ejecuta ningún demonio ni otros procesos) así que su uso principal es el modo administrativo.

/etc/rc.d/rc.M

El archivo /etc/rc.d/rc.M es el guión responsable de configurar el sistema para todos los modos multiusuario. Continúa el trabajo en el lugar en que lo abandona el

guión sysinit rc.S; una vez que rc.S ha terminado de configurar el hardware y preparar los aspectos básicos del sistema, como, por ejemplo, el montaje de los sistemas de archivos, el guión rc.M comienza a ejecutar procesos de servidor.

El guión /etc/rc.d/rc.M comienza por el arranque de los procesos básicos, como la facilidad syslog (el proceso de anotaciones del sistema), los demonios crond y atd encargados de la programación de tareas y también prepara algunos parámetros para las consolas de entrada al sistema y el lector de CD-ROM. También pone en funcionamiento varios demonios, como el de impresión (lpd), el servidor de correo saliente (sendmail) y el servidor web. Una de las tareas más importante que realiza rc.M es la activación de la red. Lleva a cabo esto mediante llamadas a otros dos guiones: rc.inet1 y rc.inet2. (Estos guiones se describen aparte, más adelante en esta misma sección.)

Nota

Muchos demonios (como lpd y sendmail) pueden suponer en ocasiones un riesgo para la seguridad; para obtener más información, vea la sección "Garantizar la seguridad del sistema" más adelante en este capítulo.

Finalmente, rc.M invoca otros guiones del sistema. Slackware, incluso aunque utiliza principalmente el modelo de guiones de inicio de BSD, también soporta guiones escritos para el modelo SysV. Soporta estos archivos a través del guión rc.sysvinit, que se describirá en esta misma sección, en la subsección "/etc/rc.d/rc.sysvinit". Mientras tanto, el último paso en el modelo BSD es invocar un guión llamado rc.local, que es el mismo guión rc.local comentado en el capítulo 4 como parte del modelo SysV que utiliza Red Hat Linux. Este guión simplemente contiene cualquier código adicional que el administrador necesite para un sistema particular. Su intención es permitir a los administradores particularizar la configuración de un sistema sin tener que realizar extensas modificaciones en los demás guiones de inicio. Por defecto, este archivo está vacío en Slackware (al contrario que en Red Hat Linux).

/etc/rc.d/rc.0 y /etc/rc.d/rc.6

Estos son en realidad el mismo archivo: rc.0 es un enlace simbólico a rc.6. El guión tiene un bloque en que detecta si ha sido llamado desde el nivel de ejecución 0 (parada del sistema) o desde el nivel 6 (reinicio del sistema). Estos niveles de ejecución son casi idénticos; sólo se diferencian en si el sistema se reinicia automáticamente al final. Por eso, los dos guiones son realmente un único archivo, para evitar la duplicación de código en dos archivos.

En cierto sentido, este archivo es justo el "inverso" de /etc/rc.d/rc.M y /erc/rc.d/rc.S. Es decir, donde rc.M tiene código que ejecuta una variedad de servicios, rc.6 los detiene todos. También lleva a cabo algo de trabajo relacionado con cosas que se inicializan en el guión rc.S, como por ejemplo la sincronización del tiempo

actual con el reloj hardware, guardar la semilla del generador de números aleatorios y desactivar el espacio de intercambio. El guión rc.6 también termina el servicio NFS (si es que está ejecutándose), desconecta las cuotas de disco y el cálculo de procesos y desactiva cualquier sistema RAID que pueda estar activo. Desmonta los discos locales y después reinicia el sistema (si se llamó como si fuera rc.6; si se llamó como si fuera rc.0, sencillamente detiene el sistema).

/etc/rc.d/rc.modules

El archivo /etc/rc.d/rc.modules es responsable de la configuración de cualquier módulo del núcleo que sea necesario para apoyar al hardware del sistema. Esencialmente, el archivo contiene líneas para cada módulo del núcleo que está disponible junto con el núcleo Linux. Tiene código que se encarga de comprobar la presencia de diversos módulos del núcleo, y si los encuentra, trata de cargarlos. También tiene muchas líneas para otros módulos del núcleo, que están comentadas. Generalmente, un administrador modificaría este archivo para activar el soporte para el hardware de un determinado sistema. La mayor parte del tiempo, sólo se necesita eliminar el comentario de las líneas apropiadas, pero en ocasiones se puede necesitar añadir una línea. Red Hat se encarga de esta funcionalidad de un modo más o menos automático mediante la utilidad de detección de hardware kudzu, y por eso no hay un archivo análogo de éste.

La única otra cosa notable que hace rc.modules es iniciar el guión rc.netdevice. Este guión es el responsable de configurar el controlador de dispositivo de la tarjeta de red; normalmente consiste en una única línea, que carga el módulo del núcleo para la tarjeta de red (si es que hay una). Este guión se invoca desde rc.modules, que a su vez se invoca desde rc.S, y así se garantiza que el dispositivo de red esté listo antes de que se invoquen los guiones de red (rc.inet1 y rc.inet2).

/etc/rc.d/rc.inet1 y /etc/rc.d/rc.inet2

Estos guiones son responsables de configurar la red para la máquina. Se invocan en orden: rc.inet1 en primer lugar, y rc.inet2 después. El guión rc.inet1 arranca las diversas interfaces de red o configura las direcciones IP y las tablas de rutas, o pide esos parámetros al servidor DHCP (*Dynamic Host Configuration Protocol*, protocolo de configuración dinámica de máquinas). (El administrador da valor a estos parámetros durante la instalación.) El guión rc.inet1 es a grandes rasgos análogo al archivo /etc/rc.d/init.d/network que utiliza Red Hat Linux (comentado en el capítulo 4). El guión rc.inet2 inicia los procesos relacionados con la red y realiza otros trabajos relacionados, una vez que la red se encuentra activa. Este guión, por ejemplo, arranca el "superservidor" inetd, el demonio portmap y el demonio de la shell segura (*Secure Shell*, SSH) para el acceso remoto, entre otros procesos. Este archivo también realiza otras tareas, como montar los sistemas de archivos de red. Algunas de las funciones menos comunes de rc.inet2 están comentadas en el archivo por defecto; por ejemplo, la mayoría de los sistemas no necesitan ejecutar un servidor de nombres de dominio (*Domain Nameserver*, DNS) o un software encaminador

(*router*), y por eso las líneas correspondientes, aunque estén presentes en el archivo como muestra, por defecto están inhabilitadas. El archivo rc.inet2 pone en funcionamiento la mayoría de los servidores relacionados con la red. Sin embargo, no maneja todo lo que tenga que ver con la red; realmente sólo se encarga de lo básico. Algunos servidores, como el servidor SMTP sendmail y el servidor web Apache realmente inician directamente desde el archivo rc.M que llama a rc.inet2.

/etc/rc.d/rc.sysvinit

Slackware Linux se basa en el modelo de guiones de inicio de BSD. No obstante, algunos paquetes de software incluyen guiones ya escritos para arrancar y detener (o en cualquier caso administrar) el software. Si estos guiones están escritos para utilizar el marco de inicio de SysV, podrían ser un poco pesado integrarlos en el marco BSD. Como una conveniencia para los administradores, Slackware Linux incluye soporte para el modelo de guiones de inicio de SysV. Este soporte se arranca mediante el archivo /etc/rc.d/rc.sysvinit, que a su vez se invoca desde otros guiones. (El guión rc.M invoca a rc.sysvint para iniciar los procesos SysV, y el guión rc.6 lo invoca para finalizarlos.) Todos los guiones de inicio SysV que inicien mediante este archivo deben estar ubicados en el directorio correspondiente para cada nivel de ejecución, usando la misma convención de nombrado que en los sistemas SysV estándar (como en Red Hat Linux). Por ejemplo, el directorio para el nivel de ejecución 3 sería /etc/rc.d/rc3.d. En general, la mayor parte de la funcionalidad comentada en el capítulo 4 también es soportada por los guiones de inicio SysV de Slackware. Cuando se entra en un nuevo nivel de ejecución, los guiones de terminación (que comienzan por la letra "K") se ejecutan, y después todos los guiones de arranque (que comienzan por "S") se ejecutan para el nuevo nivel.

El soporte que Slackware ofrece a los guiones de inicio SysV es bastante completo, pero no sofisticado; es una idea posterior o una conveniencia, así que realmente sólo es útil para enganchar guiones que ya se han escrito de un modo sencillo, en lugar de servir para el administrar el sistema. Para realizar la mayoría de las tareas de bajo nivel, tendrá que editar los guiones de inicio de BSD directamente, como se comentará en el resto de esta sección.

Otros archivos

Hay algunos otros archivos dispersos, principalmente responsables de actividades específicas. Por ejemplo, el archivo rc.httpd es el responsable de poner en funcionamiento el servidor HTTP Apache. El guión rc.sshd es responsable de ejecutar el servidor SSH (que en Slackware 8.0 es OpenSSH). Estos archivos son normalmente autoexplicativos: su simple lectura revelará lo que hacen ya que no son muy complicados.

Modificación de los guiones de arranque

EL motivo por el que Red Hat escogió el modelo SysV fue para permitir a los administradores añadir software al proceso de arranque sin tener que modificar

guiones de shell. El modelo de SysV recogerá automáticamente todos los guiones que estén dentro de su marco. La ventaja de este enfoque es que es más fácil automatizar la administración del sistema; el lado negativo es que los archivos necesitan tener más estructura, y eso puede hacer que la curva de aprendizaje sea más difícil. El modelo BSD, en contraste, es muy simple: hay un guión para una tarea, y es una estructura muy "poco profunda" (no hay que buscar en muchos niveles para encontrar el guión que se buscaba).

Modificar el proceso de arranque de Slackware supone comprender qué guiones son responsables de qué tareas. Si desea cambiar los parámetros de la red, tendrá que editar rc.inet1. Si quiere iniciar un nuevo proceso servidor o inhabilitar uno de los activados por defecto, editará rc.M o rc.inet2. Si sólo quiere añadir un pequeño guión para eliminar los archivos viejos durante el arranque, podría poder el código en rc.local. La simplicidad del enfoque BSD facilita decir dónde y qué modificar. Una vez que ha leído esta sección y ha echado un vistazo a los propios guiones, podrá administrar el proceso de arranque de Slackware sin problemas.

Seguir la pista a una ejecución

El mejor modo de comprender cómo funcionan los guiones de inicio BSD en conjunto es dar un rápido repaso a un proceso de arranque típico. Ya hemos cubierto la mayor parte del contenido real de los guiones en las anteriores secciones, así que esta sección será breve. En este ejemplo, asumiremos que el sistema está entrando en el nivel de ejecución 4 (modo normal multiusuario con X).

/etc/rc.d/rc.S

Cuando init inicia, procesa /etc/inittab. Lo primero que encuentra es esta línea:

```
si:S:sysinit:/etc/rc.d/rc.S
```

Esta línea indica que se ejecute el guión /etc/rc.d/rc.S, sin importar en qué nivel de ejecución se entre. Este guión, a su vez, realiza la configuración básica del sistema, e invoca al guión /etc/rc.d/rc.modules para que cargue los controladores de dispositivo apropiados. El guión rc.modules invoca a /etc/rc.d/rc.netdevice para configurar el hardware ethernet (u otro hardware de red) por separado.

/etc/rc.d/rc.M

La siguiente línea del archivo configura un guión que se ejecutará para el nivel de ejecución 1; ya que ese no es el nivel actual, el proceso se salta esa línea. Lo siguiente es el guión /etc/rc.d/rc.M, que se configura para ejecutarse para los niveles 2 a 5:

```
rc:2345:wait:/etc/rc.d/rc.M
```

Este guión realiza el grueso del trabajo de la configuración del sistema. Activa la red (invocando a rc.inet1 y rc.inet2), configura cosas como la fuente del sistema y el

ratón (mediante rc.font y rc.gpm)y ponen en funcionamiento servidores de red adicionales de alto nivel. Finalmente, activa cualquier guión SysV que pueda haberse configurado, mediante la ejecución del guión rc.sysvinit.

Otras acciones

Después de la ejecución del guión /etc/rc.d/rc.M, init continúa con la lectura de /etc/inittab. Las siguientes líneas definen guiones que se deben ejecutar en respuesta a diversos eventos, como fallos de energía o la combinación de teclas **Control-Alt-Supr**, que ordena el reinicio del sistema. También define los guiones que se ejecutarán durante la parada del sistema y su reinicio (/etc/rc.d/rc.0 y /etc/rc.d/rc.6).

El último bloque de líneas prepara las consolas de texto, tal y como se describe en al sección anterior "/etc/inittab". Sin embargo, ya que estamos entrando en el nivel de ejecución 4, se saltan las 5 primeras líneas. (Sólo tty6 se encuentra definida para el nivel 4.) La línea final activa X mediante el guión /etc/rc.d/rc.4, y se ejecuta en este caso porque estamos en el nivel de ejecución 4.

Eso es en esencia todo lo que sucede en el modelo de guiones de inicio de BSD que utiliza Slackware Linux. Si ha leído completamente esta sección, será capaz de doblegar Slackware a su voluntad, o al menos doblegar a sus guiones de inicio. Aún más importante, si también ha leído la sección equivalente del capítulo 4, ahora conocerá los dos modelos más importantes para guiones de inicio en sistemas similares a Unix en general. Podría encontrar algunas pequeñas variaciones de sistema a sistema, pero generalmente lo que ha aprendido será aplicable. Este único conocimiento probablemente se demostrará útil siempre que trabaje con sistemas Unix. ¡Podría incluso merecer el precio de este libro por sí solo!

Bibliotecas centrales del sistema

Esta sección comentará el software principal que se incluye con Slackware Linux. Como el objetivo de Slackware es la simplicidad y la estabilidad, Slackware Linux es un poco conservador a la hora de mantenerse a la última en cuanto al software. Sin embargo, esto no quiere decir que Slackware no esté al día. De hecho, Slackware sigue los últimos lanzamientos de software con regularidad, sólo que no agresivamente.

Slackware Linux incluye una selección de software similar a la de Red Hat (de hecho, similar a la mayoría de las distribuciones modernas). La tabla 5.2 muestra la lista de los paquetes importantes que vienen con Slackware Linux 8.0. Como comparación, véase también la tabla 4.9 del capítulo 4.

Hay algunos puntos que merecen una distinción. En primer lugar, Slackware 8.0 no incluye las bibliotecas PAM que Red Hat y otras distribuciones incluyen. (Este software es sólo un modo alternativo de manejar la autenticación del sistema, que supone en cierto sentido "reinventar la rueda".) Además, Slackware tiene nombres

y contenidos diferentes para otros paquetes. Por ejemplo, las herramientas de administración de usuarios se pueden encontrar en el paquete "shadow" en Slackware. Para obtener una lista completa de los paquetes que se instalan y de sus contenidos, simplemente eche un vistazo al directorio /var/log/packages en una instalación de Slackware Linux que esté activa.

Tabla 5.2. Bibliotecas principales de Slackware Linux 8.0.

Paquete	Versión	Comentario
XFree86	4.1.0	Sistema X Window. La versión 4 contiene extensiones de aceleración 3D y nuevos modos de representación.
glibc	2.2.3	Bibliotecas de sistema de ejecución principales. Contiene un enlazador/cargador, bibliotecas matemáticas, de seguridad, de red y de multiproceso.
bash	2.05	*Bourne Again Shell*. Shell principal del sistema (utilizada por el superusuario).
gdbm	1.8.0	Bibliotecas de base de datos de archivos. Utilizada por otros programas para indexar y almacenar datos.
fileutils	4.1	Utilidades de manipulación de archivos. Contiene programas básicos como ls, cp, rm, etc.

El sistema X Window y el escritorio

Slackware Linux, como casi todos los sistemas similares a Unix que se utilizan, incluye una versión del sistema X Window. Esta sección describirá la implementación que utiliza Slackware y cubrirá aspectos adicionales relacionados, como la instalación de entornos de escritorio.

XFree86

Slackware Linux 8.0 incluye XFree86 4.1.0 Esta era la última versión de XFree86 disponible en su momento. Red Hat Linux 7.3, en cambio, incluye XFree86 4.2.0 (que era la versión de XFree86 más reciente disponible en ese otro momento). XFree86 introdujo algunas mejoras (en forma de controladores adicionales de hardware y algún software adicional de soporte) y correcciones de errores, y es la base de ambas versiones 4.1.0 y 4.2.0.

La tabla 4.10 (en el capítulo 4) muestra una lista de varias herramientas presentes en Red Hat Linux que son útiles en compañía de XFree86. La mayoría de estas herramientas también está presente en Slackware, pero dos de ellas no lo están y merece la pena mencionarlo.

Nota

Véase el capítulo 4 para obtener información sobre XFree86 4.2.0; la mayor parte de este material también se aplica a XFree86 4.1.0.

La primera es el programa Xconfigurator, escrito por Red Hat. Este programa es una aplicación basada en texto que guía al usuario a través del proceso de configuración de XFree86. Sin embargo, Slackware no incluye este programa, ya que confía en su lugar en el programa xf86config que incluye el propio XFree86. El programa xf86config hace en esencia lo mismo que el programa Xconfigurator de Red Hat, pero la versión de Red Hat es un poco más amigable con el usuario.

A Slackware también le faltan las aplicaciones de conmutar los escritorios que sí que vienen junto con Red Hat. Estos programas permiten al usuario cambiar de entorno de escritorio. Sin embargo, Slackware confía en el método tradicional para realizar esto (el mecanismo xinit). Para obtener más información al respecto, lea la sección "Idiosincrasias de Slackware Linux" más adelante en este capítulo.

GNOME y KDE

Slackware Linux, de nuevo al igual que la mayoría de las distribuciones modernas, incluye los entornos de escritorio GNOME y KDE y permite al usuario elegir entre los dos. (El procedimiento para hacer esto se describe más adelante en la sección "Idiosincrasias de Slackware Linux" de este mismo capítulo.) Como con la versión de XFree86, Slackware Linux 8.0 incluye las versiones disponibles más recientes en el momento de KDE y GNOME; de hecho, ¡Slackware fue una de las primeras distribuciones en incluir GOME 1.4! Slackware Linux 8.0 también incluye KDE 2.1.2.

Aplicaciones de usuario

Esta sección comentará las aplicaciones de usuario que vienen incluidas en la distribución Slackware Linux. (Recuerde que una aplicación de usuario es un programa normal, en lugar de uno que se ejecuta junto con el núcleo.)

Aplicaciones incluidas

Slackware Linux incluye una selección de software bastante modesta si se compara con otras distribuciones. Esto es, donde Red Hat Linux incluye dos CD completos con un tercer CD con aplicaciones especiales (PowerTools), Slackware Linux viene en un único CD, con otro CD con extras. Debian, al igual que Red Hat Linux, viene en tres CD. A primera vista, podría parecer que esto supone que Slackware no

dispone de una paleta de software tan rica como Red Hat y Debian. Sin embargo, es importante recordar que Slackware trata acerca de la simplicidad, y por eso incluir montones de software que la amplia mayoría de los usuarios no necesitará es redundante. Las herramientas de empaquetado de Slackware trabajan con simples archivos *tarball*; esto hace que sea muy simple instalar software, sin importar de qué fuente proceda, y por eso Slackware evita la necesidad de incluir herramientas adicionales al hacer más sencillo para los usuarios instalar su propio software.

Dicho esto, Slackware aún incluye una gran cantidad de software. (Recuerde que, después de todo, incluso la más reducida distribución de Linux ofrece mucho más software que casi cualquier sistema operativo comercial.) La siguiente lista de paquetes que Slackware Linux incluye es bastante típica de los sistemas Linux:

- El compilador gcc y sus herramientas asociadas
- El navegador web Mozilla
- El editor de texto emacs

Cualquier usuario experimentado de un sistema Linux se sentirá como en casa con Slackware Linux. Generalmente, lo único que le falta a Slackware son algunos de los programas más oscuros y exóticos que se incluyen en otras distribuciones.

ZipSlack

Existe una variante de Slackware Linux conocida como ZipSlack. En el capítulo 4, se comentaron las diversas ediciones de Red Hat Linux. Slackware no tiene realmente un equivalente a esas ediciones; Red Hat las utiliza para trazar su línea de productos, para apoyar sus iniciativas comerciales. Como Slackware no es una entidad comercial, no hay motivo alguno para producir múltiples ediciones. Entonces, ZipSlack no es tanto una edición o variante de Slackware como una minidistribución. ZipSlack trata de ser lo suficientemente reducida como para caber completamente en un medio pequeño, como un disquete ZIP de Iomega. Al colocar una distribución Linux completa en un disco portátil, un usuario puede llevar una instalación Linux en su bolsillo. ZipSlack también incluye herramientas para iniciar directamente el sistema Linux desde una máquina Windows o MS-DOS. Para cumplir con ese tamaño tan reducido, ZipSlack simplemente se restringe a los componentes absolutamente críticos del sistema principal y a un conjunto mínimo de herramientas de usuario. Debido a su pequeño tamaño, ZipSlack también resulta útil en dispositivos con restricción de almacenamiento, como los asistentes personales digitales (los PDA) y dispositivos de red.

Idiosincrasias de Slackware Linux

Como cualquier distribución, Slackware Linux tiene su peculiar modo de hacer las cosas y sus singularidades. En el capítulo 4, conoció alguna de las idiosincrasias

equivalentes de Red Hat Linux; cuando haya leído esta sección, tendrá una buena idea de las diferencias entre las dos distribuciones, y habrá dado otro paso más hacia el dominio de los conceptos y no sólo las técnicas de uso y administración de un sistema Linux.

Configuración de inetd

El superservidor inetd (a veces denominado metaservidor) se comentó en el capítulo 4 en el contexto de Red Hat Linux. Red Hat utiliza el paquete xinetd como implementación de inetd para su distribución. Sin embargo, Slackware utiliza una implementación más tradicional y antigua. Esta implementación resultará muy familiar a quien haya trabajado con inetd en algún otro sistema similar a Unix tradicional. De hecho, los usuarios que lo hayan hecho pueden saltarse esta sección ya que no aportará ningún material nuevo; en cambio, los usuarios menos experimentados serán capaces de aplicar el contenido de esta sección a otros sistemas.

Recuerde que el propósito de inetd es hacer más sencillo escribir aplicaciones básicas de servidor. Es una especie de "molde de galletas" que se puede utilizar para obtener servidores similares. Los autores de software escriben pequeños programas que se encargan de las operaciones y después hace corresponder su aplicación con un puerto en la configuración de inetd. El servidor inetd, como contrapartida, enlaza todos los puertos indicados en su archivo y dirige las peticiones que le llegan al programa apropiado, según el puerto al que haya llegado la petición.

Generalmente, la implementación tradicional de inetd se configura de un modo bastante simple. La implementación de Slackware está en el paquete tcpip1 (junto con una serie de otras cosas) y esencialmente consiste en tres ficheros: el ejecutable del programa inetd, el archivo /etc/inetd.conf que configura el servidor, y el archivo /etc/services que establece nombres con significado para los números de los puertos.

El archivo /etc/services

El archivo /etc/services se usa para asignar nombre a los números de los puertos. Esto es una cuestión puramente de estética; permite a los usuarios humanos referirse a los números de los puertos mediante nombre, ya que seguir la pista de todos los puertos numéricamente puede resultar confuso. (Es similar al modo en que las direcciones IP de Internet se hacen corresponder con nombres de servidores mediante el sistema DNS; los nombres sólo sirven para el uso de los humanos.) Realmente, el archivo /etc/services lo utilizan otros programas además de inetd; puede ser utilizado por cualquier aplicación que necesite o quiera encontrar un nombre significativo para una persona que se corresponda con un número de puerto. (Se ofrece una API para las aplicaciones para permitirles buscar el nombre de un número de puerto o para encontrar un servicio a partir de un nombre legible por los humanos.)

Cada línea del archivo /etc/services tiene esta sintaxis:

```
<nombre simbólico>  <puerto>/<protocolo>   <alias>
```

El <nombre simbólico> es el término con significado para un humano, como por ejemplo "http" o "telnet". El puerto es el número de puerto TCP/IP de ese servicio, y el protocolo debe ser alguna de las cadenas de caracteres "tcp" o udp", según si el servicio utiliza un protocolo TCP (servicio orientado a flujos de datos) o UDP (servicio orientado a paquetes de datos). Los alias son opcionales, y simplemente indican nombres alternativos para el mismo número de puerto.

Nota: TCP frente a UDP

No se preocupe demasiado sobre la diferencia entre TCP y UDP; generalmente sólo lo utilizan aplicaciones de servidor y cliente para establecer conexiones del tipo apropiado. Un protocolo dado se basará en TCP o UDP dependiendo de sus necesidades. Si ha escrito un servidor usted mismo y lo va a añadir a inetd, sabrá si es TCP o UDP; si simplemente va a instalar uno, la documentación del software le indicará qué utiliza. De hecho, muchos protocolos de la lista de /etc/services tienen entradas tanto para TCP como para UDP, incluso aunque muchos sólo utilicen uno o el otro. De todos modos, esta correspondencia tiene como objetivo principalmente la referencia de los humanos, y no tiene impacto sobre el funcionamiento del software.

Aquí hay algunos ejemplos extraídos del archivo /etc/services que se incluye en Slackware 8. Esta es la línea para los servidores WWW:

```
www 80/tcp http  #WorldWideWeb HTTP
```

Esta línea indica que el protocolo "www" tiene un número de puerto 80, se basa en el protocolo de flujos de datos (TCP), y también se conoce como "http". Compare esto con la línea del protocolo trivial de transferencia de archivos (*Trivial File Transfer Protocol*, TFTP), que se basa en el protocolo UDP, se ejecuta en el puerto 69 y no tiene alias:

```
tftp  69/udp
```

Una búsqueda rápida en este archivo nos hará encontrar muchos números de puerto y protocolos familiares, como telnet (para el acceso remoto) en el puerto 23, HTTP (para servidores web) en el puerto 80, POP3 (para acceder al correo electrónico) en el puerto 110, y SMTP (para enviar correo electrónico) en el puerto 25.

El archivo /etc/inetd.conf

El archivo de configuración para el servidor inetd es /etc/inetd.conf. El contenido del archivo es también muy sencillo. Cada línea del archivo configura un pro-

grama de servicio diferente y lo enlaza a un número de puerto. Las líneas tienen esta sintaxis (que también está documentada en el propio archivo):

```
<nombre de servicio> <tipo de socket> <proto> <flags> <usuario> <ruta del
servidor> <args>
```

El parámetro <nombre de servicio> puede ser tanto un número de puerto como su nombre simbólico tal y como se especifica en /etc/services. (Si se especifica aquí un puerto no numérico, se busca en el archivo /etc/services para encontrar el número del puerto.) El <tipo de socket> puede ser "stream" (flujo) o "dgram" (que es una abreviatura de "datagrama", paquete de datos). El siguiente campo, <proto>, puede ser tanto "tcp" como "udp", y debe corresponderse con el campo <tipo de socket>. Es decir, si <tipo de socket> es "stream", entonces <proto> debe ser "tcp" y viceversa. Los campos "udp" y "dgram" están enlazados del mismo modo.

Nota

Si un sistema utiliza un protocolo de red subyacente distinto de TCP/IP, podría usar valores distintos de "udp" y "tcp" como protocolos para datagramas y flujos. Sin embargo, la inmensa mayoría de ordenadores actuales utiliza TCP/IP, así que esto es sólo un asunto teórico.

El campo <flags> debe tener el valor "wait" (esperar) o "nowait" (no esperar) para indicar si el servidor inetd debería esperar a que el programa secundario termine su labor antes de procesar más conexiones. El valor "wait" significa que esencialmente inetd cederá la posesión del puerto al programa designado cuando se reciba una conexión; entonces el programa gestionará (o no) las conexiones adicionales a ese puerto por sí mismo. Esto permite que aplicaciones críticas en rendimiento (como servidores web) puedan administrar las conexiones por sí mismas durante los momentos de uso intensivo. Hacer esto es más eficiente y mejor para la memoria del sistema que tener a inetd encargado de lanzar nuevos procesos cada vez que llega una petición, ya que el servicio puede autofinalizarse cuando ya no se necesita y ser despertado por inetd de nuevo cuando lleguen conexiones adicionales. El campo <usuario> es el nombre de usuario de la cuenta que debería ser dueño del programa secundario; normalmente, este es el usuario *root* (el superusuario), pero puede ser cualquier cuenta de usuario con los permisos apropiados. La <ruta del servidor> es la ruta hacia el programa secundario que inetd debe ejecutar. Todo lo que venga después del nombre del programa se consideran argumentos que se pasarán al programa cuando inetd lo invoque.

Los ejemplos, como siempre, son la mejor manera de mostrar cómo funciona esto. La siguiente línea define la entrada para el servicio de acceso remoto mediante telnet, y lo crea como un servidor TCP orientado a flujos conectado con el puerto telnet.

```
telnet  stream  tcp  nowait  root  /usr/sbin/tcpdin.telnetd
```

Esta línea también indica que inetd no debería esperar a que telnet finalizara antes de aceptar otra conexión telnet. (Si ese campo tuviera el valor "wait", entonces inetd esperaría a que el programa telnet finalizara antes de aceptar otra conexión; en ese caso, el programa in.telnetd debería administrar las peticiones adicionales que llegaran por sí mismo.) El proceso se ejecuta como el usuario *root* (pero el propio programa telnet conmutará al usuario apropiado una vez que acceda al sistema), y el programa que se debe ejecutar es /usr/bin/tcdp con el argumento in.telnetd.

Instalación de un nuevo servicio basado en inetd

Instalar un servicio nuevo para que sea administrado por el superservidor inetd se hace en sólo dos pasos:

1. Añadir una entrada en el archivo /etc/services que se defina un nombre para el nuevo servicio.
2. Editar el archivo /etc/inetd.conf para hacer referencia al nuevo servicio y asignar un programa que se ejecute cuando se reciba una conexión en el puerto deseado.

Nota

El capítulo 12 incluye una instalación de muestra del sistema de versiones concurrentes (*Concurrent Version System*, CVS) como servidor inetd.

Nota: Paso de argumentos

Una debilidad importante de los servicios inetd es el modo en que se pasan los argumentos desde inetd a los programas secundarios. Algunas implementaciones del programa inetd sólo permiten que se pase un número fijo de argumentos (típicamente cinco). ¡Y el primer argumento es el nombre del programa que se debe ejecutar! Si está utilizando una implementación de inetd con esa limitación (tanto si es un sistema Linux u otro sistema Unix), normalmente puede evitar ese problema al escribir un guión de shell que ejecuta el auténtico programa con todos los parámetros que necesite, y haciendo que la entrada en /etc/inetd.conf invoque ese guión de shell.

Garantizar la seguridad del sistema

Esta sección describirá cómo mejorar la seguridad de una instalación Slackware. El enfoque de Slackware se basa en la simplicidad y la estabilidad. Hasta un cierto

grado, además, los desarrolladores de Slackware esperan que los administradores asuman una cierta responsabilidad sobre el sistema. Esto significa que se espera de los administradores que comprendan y vigilen todos los aspectos de su sistema, incluida la seguridad. Por eso, Slackware deja la carga de proteger un sistema en manos de su administrador. En la práctica, esto quiere decir que las instalaciones por defecto de Slackware están optimizadas en busca de las máximas prestaciones en lugar de la seguridad. Son instalaciones que hay que cuidar y que se consideran como un punto de partida para preparar la funcionalidad necesaria, más que servidores "de arrancar y olvidar".

Una instalación por defecto de Slackware ejecuta muchos servicios; de hecho, en esencia ejecuta todos los servicios que puede ejecutar. La primera reacción de muchos usuarios nuevos ante este hecho es preguntar "¿Y qué? Si no lo necesito, sencillamente no lo usaré. No hacen ningún daño, ¿verdad?" Bueno, no, los servicios no causan ningún daño. Sin embargo, ningún programa software está libre al cien por cien de errores, y la amplia mayoría de las brechas de seguridad suceden cuando atacantes maliciosos explotan estos errores para conseguir acceso al sistema. Los servicios no causan ellos mismo ningún mal, pero sus errores pueden explotarse. Por eso, cualquier programa servidor que esté activo es un agujero de seguridad en potencia. Si se dejan activos, son un riesgo de seguridad; si se cortan, no se pierden prestaciones porque no se utilizan, pero se eliminan esos potenciales agujeros de seguridad. En pocas palabras, una buena política de seguridad es inhabilitar cualquier servicio y proceso que no sea absolutamente necesario.

Entonces, ¿cómo inhabilita un administrador estos procesos en un sistema Slackware? En el capítulo 4, se comentaron los mecanismos que utiliza Red Hat Linux (las herramientas `chkconfig` y `service`). Estas herramientas manipulan el marco de guiones de shell que el sistema de Red Hat utiliza para gestionar el sistema y los servicios. Sin embargo, Slackware Linux no dispone de herramientas de este tipo. La ausencia de estas herramientas significa que los guiones de shell tienen que modificarse manualmente. Con anterioridad se describieron en este capítulo los guiones de arranque del sistema Slackware. Esta sección tratará sobre el material que describe cómo inhabilitar servicios innecesarios.

Generalmente, una instalación de Slackware nueva tendrá típicamente activados estos servidores:

- Servicios inetd
- Demonio portmap
- Demonio de impresión lpd
- Servidor SMTP sendmail
- Servidor HTTP Apache

Esta lista puede variar, según el software que se haya seleccionado durante la instalación; si tiene una instalación de Slackware, su lista puede ser diferente. Muchos de los servicios comentados en esta sección se pueden reconfigurar para que

sean más seguros; por ejemplo, algunos pueden configurarse para responder sólo a peticiones que se originen en el sistema local (y no a través de la red). Los administradores y usuarios que necesiten estos servicios deberían leer la documentación para aprender qué opciones tiene para proteger el software. Aún así, en general, si no necesita que un servicio esté funcionando, es una buena idea inhabilitarlo, y por eso el resto de esta sección comentará como inhabilitar cada uno de estos servicios.

Inhabilitar los servicios inetd

Hay dos modos de inhabilitar los servicios inetd: o desactivándolos uno a uno en el archivo de configuración para inetd, o sencillamente desactivando totalmente el proceso inetd, de modo que no llegue a arrancarse. (Si inetd no está activo, obviamente tampoco lo estará ninguno de sus servicios.) Inhabilitar servicios inetd individuales, como se comentó con anterioridad den este capítulo, puede llevarse a cabo simplemente mediante la colocación de un carácter "#" al comienzo de la línea que se va a comentar. Sin embargo, si descubre que no necesita ninguno de los servicios inetd en absoluto, puede inhabilitar totalmente el servidor.

Recuerde que Slackware tiene dos archivos que controlan la mayoría de las configuraciones de red: rc.inet1 y rc.inet2. Ambos guiones se encuentran en el directorio /etc/rc.d. El guión rc.inet1 configura la propia red, mientras que rc.inet2 arranca la mayoría de los servidores de red, incluido inetd. Para inhabilitar inetd, entonces simplemente hay que comentar las líneas del guión que arrancan el programa. Estas son las líneas que debería buscar:

```
# Start the inetd server:
if [ -x /usr/sbin/inetd ]; then
  echo "Starting Internet super-server daemon:   /usr/sbin/inetd"
  /usr/sbin/inetd
else
  echo "WARNING:   /usr/sbin/inetd not found."
fi
# Done starting the inetd meta-server.
```

Simplemente comente las líneas que se encuentran entre las sentencias `if` y `fi` (incluyendo ambas) escribiendo un carácter "#" al comienzo de cada línea e inetd no se iniciará durante el proceso de arranque.

Inhabilitar el demonio portmap

El demonio portmap es el servidor "asignador de puertos" (*portmapper*) desarrollado originalmente por Sun Microsystems y utilizado por un cierto número de programas, incluido servidores y clientes NFS (el sistema de archivos de red). Si un sistema utiliza NFS (como servidor o como cliente) para compartir discos a través de la red, entonces necesitará que esté activo el demonio portmap. Sin embargo, si no se usa NFS, entonces este demonio se puede inhabilitar.

El nombre real del programa es rpc.portmap, y, como inetd, se ejecuta desde el archivo rc.inet2 durante el inicio del sistema. Para inhabilitar portmap, simplemente localice y comente estas líneas:

```
# This must be running in order to mount NFS volumes.
# Start the RPC portmapper:
if [ -x /sbin/rpc.portmap ]; then
  echo "Starting RPC portmapper:   /sbin/rpc.portmap"
  /sbin/rpc.portmap
fi
#Done starting the RPC portmapper.
```

Una vez más, las líneas pueden comentarse mediante el simple añadido del carácter "#" al comienzo de la línea.

El demonio portmap es realmente notorio como riesgo de seguridad. De hecho, supone un riesgo tan grande que nunca debería ejecutarse en un sistema expuesto mucho tiempo a Internet (como un ordenador doméstico o un servidor público). Si resulta absolutamente necesario tener activo portmap, entonces el sistema debería colocarse tras un cortafuego para que sólo los ordenadores locales puedan acceder a este demonio. Un caso de estudio basado en un cortafuego se tratará en el capítulo 16, así que no se preocupe.

Nota: Portmap o no

El asignador de puertos es un estándar para RPC (*Remote Procedure Calls*, o llamadas remotas a procedimientos). El servicio portmap es un medio para acceder a un programa que se ejecuta en un servidor a través de la red. Se utiliza para muchas aplicaciones, pero la más importante es el protocolo NFS que se utiliza en muchas redes. NFS se utiliza para ofrecer acceso a un único sistema de archivos (como el directorio local de un usuario) en múltiples máquinas diferentes, y NFS suele confiar en portmap. Si utiliza aplicaciones como NFS, entonces probablemente necesitará dejar el servicio portmap activo. Si no es así, debería considerar seriamente inhabilitar portmap, ya que supone un riesgo de seguridad bastante importante.

Inhabilitar el demonio de impresión lpd

El demonio lpd se utiliza para ofrecer acceso a las impresoras. Puede administrar tanto una impresora local conectada al PC, como ofrecer acceso a una impresora remota a través de la red, o también puede funcionar como servidor, recibiendo trabajos de impresión desde la red que se deben imprimir en una impresora local. A no ser que se necesiten estos servicios de impresión, el demonio lpd debería inhabilitarse. Al igual que inetd y portmap, lpd puede inhabilitarse mediante el proceso de comentar las líneas apropiadas en rc.inet2, que son estas:

```
# Start the BSD line printer spooler daemon:
if [ -x /usr/sbin/lpd ]; then
   echo "Starting BSD line printer spooler daemon:   /usr/sbin/lpd"
   /usr/sbin/lpd
fi
# Done starting the BSD line printer spooler daemon
```

Inhabilitar el servidor SMTP sendmail

Sendmail es un popular servidor SMTP. Los servidores SMTP son vulnerables a dos tipos de problemas de seguridad: además del "aprovechar externamente" sus vulnerabilidades que otros procesos también presentan, también es posible que sendmail esté mal configurado y permita que determinados atacantes abusen de él como un modo de enviar correo electrónico no solicitado (esto es, correo basura) a usuarios inocentes. Típicamente esto causa que el correo basura parezca (al menos tras una inspección superficial) haberse originado en el servidor utilizado como "retransmisor". Por estos motivos, tiene doble importancia asegurarse de que la instalación de sendmail está bien configurada o inhabilitada.

Es notoriamente difícil administrar sendmail, así que a no ser que sea realmente necesario, suele ser buena idea inhabilitarlo. Generalmente, el único motivo para dejar activo sendmail en cualquier sistema que no sea un servidor de correo es para beneficiar a algunos clientes de correo electrónico que no pueden utilizar un servidor STMP que se ejecute en una máquina a través de la red. Sin embargo, casi cualquier cliente gráfico de correo electrónico tiene esta capacidad, así que se suele poder inhabilitar sendmail. No obstante, al contrario de otros servicios, inhabilitar sendmail no se hace mediante el archivo rc.inet2. El archivo rc.inet2 es para los servicios de red vitales que tienen que estar funcionando para que el sistema funcione normalmente. Otros servidores menos cruciales (como los servidores SMTP y HTTP) se arrancan desde el archivo rc.M.

El archivo rc.M funciona comprobando primero si hay instalado un determinado programa servidor. Si se encuentra, se ejecuta; si no, se salta ese proceso. Hay, por ello, dos modos de inhabilitar un servidor que se arranque desde rc.M. Pude renombrar el programa de modo que rc.M no "lo vea" más y por ello se salte su activación, o puede simplemente comentar las líneas de rc.M que buscan el guión. Sin embargo, probablemente sea mejor comentar estas líneas que renombrar un programa ejecutable, ya que renombrar (o borrar) un programa podría causar problemas más adelante. Por ejemplo, otro administrador o usuario podría querer rehabilitar el servidor, pero no podría si se le ha cambiado el nombre por otro excesivamente críptico. ¡Si el programa se ha borrado, no puede recuperarse en absoluto, a no ser que se reinstale!

Para inhabilitar sendmail, comente estas líneas en el archivo rc.M:

```
# Start the sendmail daemon:
if [ -x /usr/sbin/sendmail ]; then
  echo "Starting sendmail daemon:   /usr/sbin/sendmail -bd -q15m"
  /usr/sbin/sendmail -bd -q15m
fi
```

Inhabilitar el servidor HHTP Apache

El servidor HTTP Apache es un servidor web, y por ello puede servir diversos tipos de contenido estático y dinámico a navegadores web y otro tipo de clientes. Al igual que sendmail, los guiones de arranque de Slackware inician Apache desde el archivo /etc/rc.d/rc.M. Estas líneas necesitan comentarse para inhabilitar Apache:

```
# Start Web server:
if [ -x /etc/rc.d/rc.httpd ]; then
  . /etc/rc.d/rc.httpd start
fi
```

Los lectores astutos se darán cuenta de que en este caso, Apache no inicia directamente. En su lugar, se invoca un guión de shell (etc/rc.d/rc.httpd), que a su vez inicia Apache. Sin embargo, ese guión de shell es muy simple y únicamente invoca el propio guión de control de Apache, llamado apachectl.

Otros servidores

Recuerde que según qué software se haya escogido, una determinada instalación de Slackware puede tener un conjunto de servidores distinto del tratado en esta sección. Sin embargo, sin importar qué servidor sea, se arrancará desde /etc/rc.d/rc.inet2 o desde /etc/rc.d/rc.M directamente. Como Slackware espera que los usuarios se ocupen de la administración de sus propios sistemas, es de vital importancia revisar estos archivos para cada nueva instalación de Slackware para asegurarse de que ningún servicio innecesario está ejecutándose.

Personalización del entorno de usuario

Esta sección comentará cómo adaptar los guiones de configuración de las shell de acceso al sistema en un sistema Slackware Linux. En el capítulo 4, se trató el marco de guiones de entrada al sistema de Red Hat. Los lectores deberían consultar ese capítulo para poder realizar una comparación, pero a modo de resumen, Red Hat Linux utiliza el modelo de depositar archivos de configuración en el directorio config.d. Realmente, esta sección va a ser bastante sencilla, ya que los guiones de shell de usuario de Slackware son substancialmente similares a los de Red Hat.

El modelo básico es el mismo que se usa en Red Hat Linux. Los guiones principales (para las shells csh y sh) preparan algunos parámetros básicos y después buscan en el directorio /etc/profile.d archivos adicionales para la shell. Las shells basadas en sh (usualmente bash, ash, ksh y zsh en Slackware) leen archivos con la extensión .sh; la shell csh (que es casi siempre tcsh en los sistemas Linux) busca archivos con la extensión .csh. Este es el mismo mecanismo utilizado por Red Hat Linux, así que los lectores deberían definitivamente consultar el capítulo 4.

Sin embargo, hay algunas diferencias. En primer lugar, y probablemente más importante, los guiones de Slackware son un poco más simples que los de Red Hat, de acuerdo con la filosofía de Slackware. Los de Red Hat, por otro lado, son más sofisticados y hacen algunas cosas más. Es el clásico compromiso de la simplicidad contra la funcionalidad, y volvemos a comprobar que Slackware se decide por la simplicidad.

La otra diferencia principal en la configuración de la shell entre Slackware Linux y Red Hat Linux radica en el uso de los archivos. Red Hat utilizar los archivos /etc/bashrc y /etc/profile para administrar bash y otras shells basadas en sh, respecti-

vamente. Mediante el uso de un archivo /etc/bashrc separado, Red Hat puede configurar los entornos de los usuarios de bash para aprovecharse de las características extendidas de bash. En contraste, Slackware no tiene un archivo /etc/bashrc y simplemente reutiliza /etc/profile para todas las shells derivadas de sh. Además, curiosamente, Slackware coloca todas las configuraciones de csh en /etc/login (/etc/cshrc está vacío). Red Hat utiliza /etc/csh.cshrc para realizar la configuración básica y /etc/lcsh.login para la configuración que sólo es apropiada para las shells de acceso (en oposición a las shells que se arrancan para ejecutar guiones).

Cambiar el entorno de escritorio de X Window de un usuario

En el capítulo 4 se mencionó el programa switchdesk de Red Hat Linux que se utiliza para cambiar el entorno de escritorio de X Window preferido del usuario; esta sección comentará cómo realizar la misma tarea en Slackware Linux. Slackware no tiene el programa switchdesk (ni ningún equivalente) y confía un modo más tradicional de realizar la tarea.

Por defecto, Slackware Linux utiliza el gestor de escritorio procedente del proyecto KDE, llamado KDM. KDM permite al usuario seleccionar uno de los entornos de escritorio instalados, y usará ese entorno durante la duración de la sesión del usuario.

Nota

Un gestor de escritorio es un programa gráfico en el que se introduce el nombre de usuario y la contraseña para acceder a X. La otra alternativa es ejecutar X desde la línea de comandos mediante el comando `xinit`.

Cuando el usuario accede al sistema, KDM ejecuta el guión /etc/X11/xdm/Xsession y le pasa el nombre del escritorio que el usuario seleccionó como su primer argumento. El guión Xsession inicia entonces el entorno de escritorio correspondiente a la elección del usuario. Sin embargo, si el usuario tiene un programa ejecutable en su directorio local que se llame .xsession, ese programa (que puede ser un guión) se ejecutará en su lugar, anulando la selección que el usuario haya realizado en KDM.

Cuando un usuario selecciona un escritorio desde KDM, la elección se guarda en un archivo llamado .wmrc en el directorio local del usuario. Este archivo contiene una única palabra que indica el escritorio más recientemente seleccionado por el usuario. Este archivo lo utiliza KDM para "recordar" la elección del usuario la siguiente vez que acceda a KDM.

Todo esto está bien, pero no contesta realmente a la pregunta de cómo un usuario cambia su entorno de escritorio. Bien, dado el mecanismo descrito, hay dos

caminos. Primero, el usuario puede editar simplemente el archivo ~/.wmrc y cambiar la palabra de ese archivo por otro entorno de escritorio. Sin embargo, esto no tiene mucho sentido ya que probablemente sea más sencillo seleccionar un nuevo escritorio desde el menú que aparece en el gestor de escritorios. El segundo método es crear un archivo ~/.xsession que contenga (o, más frecuentemente, apunte mediante un enlace simbólico) un programa que ejecute el entorno del usuario. Este parámetro siempre será prioritario frente a cualquier otra cosa que el usuario pueda seleccionar en el gestor de escritorios. Por ejemplo, el siguiente comando de enlace designará a GNOME como el entorno de escritorio del usuario.

```
ln -s /opt/gnome/bin/gnome-session ~/.xsession
```

Si el usuario seleccionó la opción por defecto y no hay un archivo ~/.xsession, el guión Xsession simplemente ejecutará el entorno por defecto mediante la invocación de /etc/X11/xinit/xinitrc (que es en realidad un enlace simbólico a uno de los otros archivos contenidos en /etc/X11/xinit/xinitrc, cada uno de los cuales maneja un entorno diferente). Probablemente esté quedando claro que los guiones que manejan todo esto son bastante complicados. Otras distribuciones, como Red Hat Linux son incluso más complicadas y elaboradas. Es por esta complejidad por lo que se escriben programas como switchdesk de Red Hat. Slackware mantiene las cosas en un estado relativamente simple, y los usuarios pueden cambiar de entorno manualmente mediante la creación de un archivo ~/.xsession o la selección del entorno deseado en el gestor de escritorios durante el proceso de acceso al sistema.

Añadir hardware nuevo

Una de las quejas más comunes sobre los sistemas Linux es que el soporte del hardware es en ocasiones irregular y a veces difícil de configurar. Esta sección comentará cómo instalar y configurar hardware nuevo en Slackware Linux. En el capítulo 4, se comentó la herramienta kudzu de Red Hat Linux encargada de la detección y configuración automática de nuevo hardware; lamentablemente, Slackware Linux no dispone de una herramienta equivalente y el procedimiento es mucho más manual y pesado.

De momento, hemos visto varios ejemplos de la filosofía de la simplicidad y la autosuficiencia de Slackware. Quizás el ejemplo más explícito de esto esté en su soporte al hardware. Recuerde que Slackware incluye núcleos Linux estándar. No hay un modo sencillo de rodear el problema: el proceso de añadir soporte para el nuevo hardware a un sistema Slackware es simplemente configurar el propio núcleo Linux para soportar el nuevo hardware (típicamente mediante la configuración del soporte para el controlador del dispositivo en forma de módulo del núcleo) y después cargar el módulo en Slackware.

En la mayoría de los casos, una vez que ha compilado el núcleo Linux con el controlador del dispositivo incluido en él o separado como un módulo, el núcleo detectará el dispositivo durante el arranque o si no lo cargará automáticamente cuando se le pida. Slackware confía en este comportamiento para preparar el

hardware, y generalmente es bastante sencillo conseguir que se cargue el controlador. La principal dificultad reside en configurar el resto del sistema para que utilice el nuevo dispositivo; por ejemplo, añadir un lector de tarjetas de memoria rápida compacta USB implica no sólo compilar y cargar los módulos el núcleo, sino también crear un punto de montaje y modificar el archivo /etc/fstab de manera apropiada.

La herramienta kudzu de Red Hat automatiza la mayoría de estas tareas. Lamentablemente, no hay un modo sencillo de hacer lo mismo con Slackware y por eso los administradores tienen que arremangarse y cavar su propio camino en la configuración del sistema. Si se necesita crear un punto de montaje, tiene que hacerse manualmente; si se necesita cargar un módulo en un cierto orden, tiene que hacerse desde /etc/rc.d/rc.local o rc.M; si un nuevo dispositivo de bloques (un disquete) se instala, necesita configurarse en el archivo /etc/fstab.

Todo esto está bien, por supuesto, pero no ayuda demasiado a los administradores. No hay una lista de comprobaciones que un administrador pueda seguir para instalar hardware nuevo; cuando se llega el momento de hacerlo, hay sencillamente demasiados casos que considerar como para crear una lista de comprobaciones realmente exhaustiva. Entonces, ¿qué puede hacer un administrador? Espero que este libro ofrezca las herramientas y el conocimiento necesarios para hacer esto.

Derivados de Slackware Linux

En esta sección se comentarán algunos de los derivados de Slackware Linux. Un derivado de Slackware Linux es una distribución Linux alternativa que se basa en Slackware pero no es idéntica. Algunos de estos derivados están más fuertemente emparentados con Slackware que otros.

SuSE Linux

SuSE Linux es una distribución especialmente popular en Europa, aunque también tiene un fuerte seguimiento en los Estados Unidos. SuSE es realmente una organización comercial, como Red Hat o Caldera. SuSE Linux se derivó originalmente de Slackware (y de hecho comenzó simplemente como una traducción de Slackware al alemán) pero con el crecimiento de la empresa, finalmente crearon su propia distribución a partir de la nada, basándose en la herramienta de administración de paquetes RPM de Red Hat. Hoy en día, SuSE Linux tiene poco parecido con Slackware Linux.

Peanut Linux

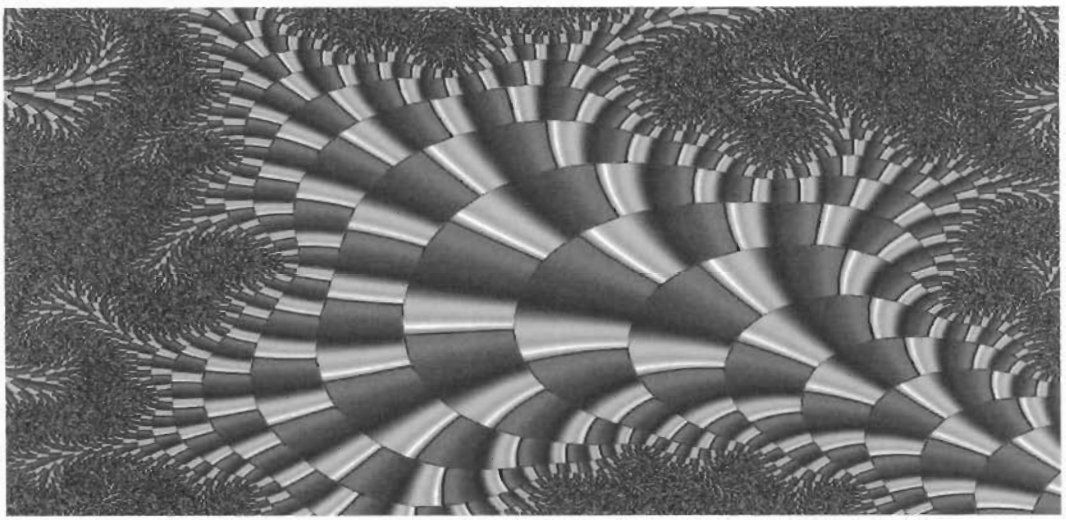
Peanut Linux es un derivado de Slackware Linux similar en ámbito a ZipSlack, mencionado con anterioridad en este capítulo. Esto es, Peanut Linux es una distri-

bución muy reducida que instala sólo unas pocas centenas de megabytes y por eso resulta útil en entornos con restricciones de espacio. Peanut Linux es un buen ejemplo de cómo la simplicidad de Slackware le permite aumentar o reducir su tamaño de acuerdo con las necesidades de un nicho de uso en particular.

Resumen

Slackware Linux en una distribución Linux completamente moderna que se centra en la simplicidad y la estabilidad. Slackware es la más similar a Unix de las distribuciones Linux (esto quiere decir que permanece más fiel a las técnicas tradicionales de los sistemas Unix que la mayoría de las demás distribuciones). Por este motivo, Slackware es probablemente el camino más fácil para que un usuario tradicional de Unix se introduzca en el mundo Linux. La simplicidad de Slackware lo hace muy fácilmente personalizable, y también lo convierte en un gran medio para que usuarios nuevos aprendan cosas sobre los sistemas Unix y Linux.

Si ha leído el capítulo 4 además de este, ya habrá visto dos distribuciones en profundidad: Red Hat Linux y Slackware Linux. Es importante recordar lo que se mencionó al comienzo de este capítulo; al comparar lass dos distribuciones no se pretende defender que una sea mejor que la otra, sino mostrar sus diferencias. Una vez que comprenda las diferencias entre los modos que tienen de funcionar estos dos sistemas, dispondrá de dos perspectivas sobre el mundo Unix y estará un paso más cerca de alcanzar ese remarcable estado de Zen en que será capaz de realizar lo que quiera con su sistema, sin importar cómo está preparado. O ya sabrá cómo hacerlo, o podrá aprenderlo. Una vez que haya leído también el capítulo 6, tendrá un conocimiento práctico sobre varias distribuciones Linux importantes, que se traducirá en un conocimiento práctico sobre cualquier sistema Linux (o incluso similar a Unix).



6
Debian GNU/Linux 3.0

El Proyecto Debian se fundó en 1993 para crear la distribución Debian GNU/Linux. Desde entonces, la distribución se ha vuelto bastante frecuente, y en muchas encuestas rivaliza o excede a Red Hat Linux en popularidad. Este capítulo comentará la distribución Debian GNU/Linux. Sin embargo, hay que hacer una advertencia importante: este capítulo sólo trata acerca de la versión beta de Debian 3.0. La versión final no se ha hecho aún pública en el momento de la publicación de este libro. Incluso así, probablemente descubrirá que las diferencias entre el material de ese capítulo y la realidad de la versión final de Debian GNU/Linux 3.0 son menores.

Como se mencionó en los capítulos 4 y 5, el objetivo de la segunda parte es ofrecerle diferentes perspectivas y soluciones al problema de desarrollar y utilizar una distribución. Las enseñanzas básicas y la exposición de las tres distribuciones de ejemplo le darán una base a la que echar mano cuando trabaje con cualquier distribución. Hasta ese punto, ha leído acerca de dos distribuciones: Red Hat Linux y Slackware Linux; tras la lectura de este capítulo, se habrá familiarizado con Debian GNU/Linux del mismo modo, y habrá dado un paso más hacia el dominio de Linux. Tras este capítulo, la tercera parte tratará acerca de cómo particularizar su sistema e instalar software en él.La idea básica es que cuantas más distribuciones vea, más capaz será de enfrentarse con cualquier distribución. Sin embargo, a partir de un punto suficiente será realmente suficiente. Llegado a ese punto comenzará a ver las mismas cosas una y otra vez, que no es realmente de mucha ayuda. Entre Slackware Linux y Red Hat Linux casi ha llegado ya a ese punto. Por ejemplo, Slackware utiliza el modelo BSD para los guiones de inicio, mientras que Red Hat

utiliza el enfoque de SysV. Detallar el enfoque de otra distribución más no merece la pena.

Como resultado, encontrará este capítulo bastante más corto que los capítulos 4 y 5. El motivo es que en los casos en que Debian GNU/Linux sea similar a Red Hat o Slackware Linux (o a ambos) simplemente citaré las diferencias que resten y proseguiré con la explicación. Sin embargo, entraré en detalle en aquellos casos en que Debian hace las cosas de otro modo.

El hecho de que este capítulo sea más corto que los otros es la evidencia de que mi estrategia está dando resultado. Es decir, ahora que ha visto dos distribuciones, puedo coger una tercera y comenzar a señalar los puntos en que su conocimiento sobre las otras distribuciones se aplica a esta nueva distribución. Esto le permitirá centrarse en las áreas en que Debian es único. Piense en este capítulo como en un simulacro de aprendizaje acerca de una nueva distribución.

Antecedentes y filosofía de Debian

Esta sección describirá el Proyecto Debian global y comentará las motivaciones y filosofía que lo respaldan; secciones posteriores comentarán la propia distribución. Quizás el mejor modo de entrar en contacto con Debian sea revisar dos documentos interesantes: el Contrato Social Debian (que delinea la filosofía del proyecto en su conjunto) y las Políticas Debian (que especifican completamente los detalles de cómo se implementa la distribución).

El Contrato Social Debian

El Proyecto Debian está muy ligado al Proyecto GNU, al menos espiritualmente. El nombre de la distribución hace esto explícito: "Debian GNU/Linux". Como leyó en el capítulo 1, puede argumentarse que "GNU/Linux" es un término más preciso que "Linux", porque refleja con mayor precisión la contribución del proyecto GNU al sistema. Ya que Debian está íntimamente relacionado con GNU, podría (correctamente) esperar que para comprender los objetivos y la filosofía del Proyecto Debian necesite comprender el software libre, el software de código abierto y cómo estos conceptos son similares y diferentes. En otras palabras, para entender Debian, lea en primer lugar el capítulo 1 para captar el concepto. (Sin embargo, no tome este libro como El Evangelio; vea también los sitios web relevantes y fórmese su propia opinión.)

Desde el comienzo, está claro que Debian es una organización consciente socialmente. Si piensa en Red Hat Software como en una organización de "software de código abierto" canónica, entonces podría pensar en el Proyecto Debian como en una organización canónica de "software libre". Déjeme señalar rápidamente, no obstante, que se trata sólo de una analogía, y las diferencias no son realmente tan nítidas: las dos organizaciones comparten muchos de los mismos ideales.

Para ilustrar su postura, el Proyecto Debian ha publicado un "Contrato Social Debian". Este documento establece la postura del Proyecto sobre el software de código abierto y libre, y establece políticas y garantías sobre el tipo de software que se incluirá en la distribución. Este documento es un compromiso con un cierto ideal social y un estándar de integridad en los contenidos de la distribución Debian GNU/Linux. Puede encontrar el Contrato Social en Internet, en la dirección web www.debian.org/social_contract.

El Contrato Social Debian es también notable en que parte de él se convirtió en la base para la definición de software de código abierto utilizada por muchas organizaciones. Esto ilustra la íntima relación entre los bandos del software libre y el software de código abierto y tal vez dé una perspectiva sobre todo el asunto. Después de todo, la mayoría de los objetivos son iguales; sólo unas pocas diferencias existen en los detalles.

Las Políticas Debian

El Contrato Social Debian delimita la filosofía básica de la distribución; las Políticas definen los detalles. El Proyecto Debian tiene un conjunto exhaustivo de políticas que gobiernan casi todos los aspectos del sistema. Hay políticas tan generales como la definición de la estructura del sistema de archivos (que está dictada por el estándar de jerarquía de sistema de archivos de Linux, el FHS) y tan específicas como la que gobierna el comportamiento preferido para los guiones en un archivo de paquete Debian (.deb). Todos los colaboradores a Debian GNU/Linux siguen las Políticas Debian, lo que hace la mayoría de los aspectos del sistema bastante consistentes y fiables en cuanto a su comportamiento. (Por ejemplo, es raro encontrar un paquete .deb que instale archivos en una ubicación inusual.) Si no es un colaborador, entonces probablemente esté menos interesado en estas políticas. Sin embargo, aún son útiles para cualquiera que intente adquirir dominio sobre Debian GNU/Linux; después de todo, explican en detalle el sistema entero.

Si ha leído los capítulos 3 a 5, entonces el contenido real de las Políticas Debian (especialmente la sección 10, "El Sistema Operativo") probablemente no le sorprenda; tratan acerca del formato de empaquetado, la instalación de archivos, el modelo de guiones de inicio utilizado y demás. Si utiliza un sistema Debian (e incluso si no lo hace), entonces es una buena idea hojear al menos las Políticas Debian. Puede encontrarlas en la dirección web www.debian.org/doc/debian-policy/.

El Contrato Social Debian contrae un compromiso con una determinada filosofía, y las Políticas Debian definen el modo en que esas filosofías se implementan en la distribución Debian GNU/Linux. Estos documentos tienen obviamente la primera y la última palabra sobre estos asuntos. Consecuentemente, en lugar de hacer un refrito de estos documentos, el resto de este capítulo se centra en mostrar cómo se puede comparar Debian GNU/Linux con respecto a Red Hat Linux y Slackware Linux. Para el resto de este capítulo, cualquier referencia a Red Hat Linux se referirá a material del capítulo 4, y las referencias a Slackware Linux tendrán que ver con el material del capítulo 5.

Una de las políticas de Debian es que la distribución debe seguir el FHS excepto cuando entre en conflicto con otra de las Políticas Debian. El FHS se comentó en el capítulo 3, y, como se mencionó en los capítulos 4 y 5, Red Hat y Slackware también siguen de cerca el FHS. Como resultado, descubrirá pocas sorpresas en la estructura del sistema de archivos de Debian. Puede encontrar alguna singularidad ocasional, pero no deberían tener ningún problema con ella.

Núcleo Linux

Como el núcleo es el corazón del sistema, un núcleo inestable implica una distribución inestable. Seleccionar una versión del núcleo para instalarla con la distribución supone siempre hacer malabarismos entre la estabilidad y las características; si desea las más novedosas características, puede tener que tolerar algo menos de estabilidad.

Para la versión 3.0, Debian GNU/Linux ha escogido la serie 2.2.x de núcleos. Slackware Linux 8.0 también incluye un núcleo 2.2.x, pero soporta núcleos 2.4.x para aquellos usuarios que los necesiten. Del mismo modo, Debian 3.0 también instala un núcleo 2.2.x por defecto, pero incluye un paquete de núcleos 2.4.x para los usuarios que los quieran. Contraste esto con Red Hat Linux 7.3, que incluye un núcleo 2.4.x por defecto. El enfoque de Debian es en este caso ofrecer la máxima estabilidad en la instalación por defecto, mientras que se proporciona la flexibilidad adecuada para aquellos usuarios que la necesiten.

Empaquetado Debian

El sistema de gestión de paquetes de Debian podría ser perfectamente el más sofisticado disponible para cualquier sistema similar a Unix. El mecanismo de Debian es incluso transparente ante la red (que significa que puede instalar paquetes remotamente a través de la red de forma transparente) y estuvo entre los primeros sistemas de empaquetados en serlo. La funcionalidad up2date de Red Hat es más reciente que su equivalente de Debian, y Slackware Linux no alcanza el nivel de una auténtica transparencia de red. Los usuarios familiarizados con otros sistemas operativos como Microsoft Windows pueden reconocer similitudes con Windows Update, aunque el mecanismo de Debian se puede defender con argumentos como que es más potente. En esta sección se comenta en detalle el sistema de administración de paquetes de Debian.

Resumen técnico

El sistema de empaquetado de Debian está construido en varias capas. En el nivel más bajo está la gestión real de los paquetes físicos, y en el nivel más alto hay

programas amigables con el usuario con varias interfaces preparadas para automatizar el proceso de localizar, obtener e instalar paquetes de software. La tabla 6.1 resume los principales programas involucrados.

Tabla 6.1. Resumen del sistema de empaquetado de Debian.

Programa	Basado en	Propósito
dpkg	No aplicable	Instala y administra los paquetes físicamente.
apt-get	dpkg	Administra la lista fuente de paquetes y obtiene los paquetes para su instalación.
dselect	apt-get, dpkg	Permite al usuario seleccionar paquetes de la lista fuente para su instalación.
tasksel	apt-get, dpkg	Permite al usuario seleccionar tareas (grupos lógicos de paquetes) para su instalación.

Las siguientes secciones comentan los programas de empaquetado de Debian en detalle. Con un propósito educativo, los programas de Debian se compararán con sus equivalentes en las distribuciones Red Hat Linux y Slackware Linux. Esto es puramente como comparación y no trata de ser un juicio acerca de qué sistema es mejor. El sistema de Debian es bastante sofisticado, rico en características y de una complejidad correspondiente, así que disponer de una terminología común para la discusión supondrá una ayuda importante.

Uso del programa dpkg

El programa dpkg es dónde la goma entra en contacto con el asfalto para el sistema de empaquetado de Debian. Este programa es responsable de desempaquetar e instalar físicamente los archivos de paquetes Debian (archivos con la extensión .deb), y de administrar las diversas bases de datos relacionadas con la administración de paquetes.

Puede utilizar dpkg para instalar manualmente paquetes individuales. Por ejemplo, puede descargar un archivo .deb de un sitio web e instalarlo con dpkg, o puede instalar un paquete a partir de un CD. Del mismo modo, puede usar dpkg para eliminar paquetes individuales específicos. También puede ejecutar comandos de preguntas comunes con dpkg, como el que proporciona la lista de los archivos instalados por un paquete en particular o el que encuentra el paquete al que pertenece un archivo en particular.

En este punto, puede haberse dado cuenta de que el programa dpkg es esencialmente equivalente al programa rpm, en el sentido de que puede instalar y eliminar paquetes y realizar diversas preguntas sobre los paquetes (tanto instalados como no instalados). Slackware Linux dispersa la misma funcionalidad entre varios programas (más específicamente, entre installpkg, upgradepkg, y removepkg).

En el capítulo 5, ha leído que Slackware Linux utiliza simples archivos de texto dentro del directorio /var/log/packages para almacenar la información acerca de qué paquetes son los propietarios de qué archivos. En el capítulo 4, por otra parte, aprendió que Red Hat Linux está en el otro extremo y almacena toda su información en una base de datos binaria a la que no puede acceder excepto a través del comando `rpm`. El programa dpkg de Debian reside en algún lugar entre ambos extremos: la lista de los archivos instalados por los paquetes está almacenada en una base de datos, pero la información sobre los paquetes se almacena en el archivo /var/lib/dpkg/available, y la información sobre los paquetes instalados se encuentra en el directorio /var/lib/dpkg/info. Hasta cierto punto, puede consultar estos archivos en busca de información sin tener que recurrir a la herramienta dpkg (aunque puede encontrar más conveniente el uso de esta herramienta).

Una de las diferencias principales entre dpkg y rpm es que dpkg mantiene una lista de todos los paquetes disponibles que aún no se han instalado, esto es, el archivo /var/lib/dpkg/available. (En realidad, Red Hat ofrece un paquete conocido como rpmdb que incluye algunas de estas prestaciones, pero tiene un uso bastante críptico y ni se acerca en cuanto a accesibilidad.) Esta lista se construye típicamente a partir de los contenidos de un CD u otro medio, y es consultada por diversas peticiones ordenadas por el usuario. (Por ejemplo, el usuario podría preguntar en busca de información de documentación sobre un paquete que aún no se ha instalado.) Esto resulta conveniente porque significa que no tiene que tener necesariamente el paquete Debian para poder extraer información sobre él; con rpm necesita disponer del paquete real para poder ejecutar comandos `rpm -qp` sobre él. Leerá muchas más cosas sobre esta lista de paquetes disponibles según avance en su lectura, así que manténgalo fresco en su mente.

Otra diferencia entre el sistema de Debian y RPM de Red Hat está en su modo de manejar las dependencias. Leyó en el capítulo 4 que RPM puede seguir la pista de la información de dependencia (es decir, de qué otros paquetes depende un paquete en particular). Por ejemplo, el paquete OpenSSH depende del paquete OpenSSL. Con RPM, una dependencia es una proposición de tipo "si" o "no". Sin embargo, la herramienta dpkg soporta diversos tipos de dependencias, como "necesario", "recomendado" y "sugerido". Esta flexibilidad añadida puede hacer más fácil transmitir la información de dependencia a los usuarios finales. Cuando instale un paquete Debian, asegúrese de comprobar sus dependencias con cuidado, ya que si no, puede haber características recomendadas o sugeridas que podría perderse.

Uso del programa apt-get

El programa apt-get se apoya en dpkg y gestiona fuentes de paquetes, ofreciendo la mayor parte de la transparencia de red mencionada anteriormente. Puede pensar en este programa como en una especie de buen alimentador de dpkg. Todos los paquetes que dpkg instala tienen que venir de alguna parte, después de todo, y apt-get es esa parte.

La otra responsabilidad principal de apt-get es seguir la pista de las dependencias entre paquetes (como qué paquetes son necesitados por otros).

Administración de los medios fuente

Típicamente, los usuarios instalan Debian desde los CD de instalación o desde un repositorio de instalación (como un sitio FTP) a través de una red. Estos medios de instalación realmente sólo consisten en una colección de paquetes Debian; apt-get vigila estos medios y sigue la pista de la lista de paquetes que se encuentra disponible desde cada fuente. Cuando el usuario desea instalar un nuevo paquete, simplemente indica a apt-get que localice el paquete y se lo ceda a dpkg para que lo instale. Localizar el paquete podría significar volver a ese sitio FTP o pedir al usuario que inserte uno de los CD de instalación, o incluso alguna otra cosa.

Esta habilidad para estar al tanto de los paquetes disponibles es una conveniencia muy importante. Los sistemas como RPM son realmente bastante miopes: rpm no puede ver ningún archivo que no esté justo frente a él. Por el contrario, el programa apt-get de Debian tiene una vista mucho mejor del "terreno" de los paquetes disponibles y puede recuperar un paquete sin que el usuario se tenga que molestar en darle caza. Red Hat tiene un sistema algo similar conocido como up2date, pero up2date típicamente sólo conoce el repositorio de paquetes que mantiene Red Hat, Inc., que vende el acceso hasta éste como un servicio con el que conseguir beneficios. apt-get de Debian tiene una lista de fuentes de paquetes mucho más flexible. Podría pensar en up2date como en un primo de apt-get que sólo puede hacerse cargo de una fuente de paquetes.

La lista de medios fuente que apt-get utiliza está a su vez gestionada por varios programas auxiliares. El archivo que almacena la lista de fuentes es /etc/apt/sources.list; este archivo puede contener una amplia variedad de fuentes, como CD, sitios FTP o HTTP (web), y muchos más. El contenido de este archivo puede ser administrado convenientemente por los programas auxiliares, como apt-cdrom, que analiza los CD de paquetes de Debian y crea las entradas apropiadas en /etc/apt/sources.list.

Nota: Vuelta a la transparencia de red

Debian GNU/Linux contiene la herramienta apt-get que, entre otras cosas, ofrece acceso a los paquetes de software de un modo transparente a la red. Otros sistemas de empaquetado incluyen prestaciones similares, y hay una gran cantidad de trabajo en curso para "hibridar" esas herramientas. Por ejemplo, hay al menos una implementación de código abierto del servidor de Red Hat para su programa up2date que permitiría usar up2date con proveedores distintos a Red Hat. Mientras tanto, hay esfuerzos orientados hacia la configuración de apt-get para su uso con el sistema RPM de modo que los usuarios de sistemas Red Hat Linux puedan utilizar apt-get para obtener y administrar paquetes RPM al igual que los usuarios de Debian administran paquetes .deb. También se trabaja en otras iniciativas interesantes, así que de ningún modo se ha cerrado el libro sobre el futuro de la administración de paquetes.

Trabajo con apt-get

Cuando se administran paquetes en un sistema Debian, muchos usuarios trabajan con apt-get la mayoría del tiempo. Está mejor preparado para añadir rápida y fácilmente aquellos paquetes que olvidó incluir cuando instaló el sistema, o para actualizar paquetes actuales con una versión más nueva cuando se hagan públicas correcciones o mejoras. No obstante, apt-get es tan sofisticada que puede encargarse de actualizar una distribución al completo a una nueva versión. Estas características de apt-get se comentarán con más detalle después.

Uso del programa dselect

El programa dselect es una interfaz basada en menús para seleccionar paquetes para su instalación. Este es realmente el mismo programa que se invoca durante el proceso de instalación de Debian que permite a los usuarios ajustar la lista de paquetes para adaptarla a sus necesidades. El programa también está disponible para ejecutarlo en cualquier momento tras la instalación para ajustar la lista seleccionada.

Este programa es realmente sólo una forma de ver la lista de paquetes disponibles mantenida por apt-get y dpkg. Es decir, apt-get mantiene una lista de fuentes de paquetes y dpkg mantiene una lista de paquetes disponibles (que está formada a partir de los contenidos de las fuentes). El programa dselect simplemente muestra una lista de todos los paquetes disponibles, indicado cuales están ya instalados y permitiendo al usuario que marque paquetes disponibles para su instalación, o paquetes instalados para su eliminación. El menú de paquetes disponibles también incluye información sobre las dependencias que tiene cada paquetes, y ayuda al usuario a seleccionar paquetes para resolver estas dependencias.

Cuando el usuario realiza su selección y abandona el programa dselect, apt-get (y luego dpkg) se ejecuta para instalar y eliminar físicamente los paquetes de acuerdo con las selecciones del usuario. Red Hat Linux y Slackware Linux no tienen realmente herramientas equivalentes a dselect.

Uso del programa tasksel

El programa tasksel es similar a dselect, pero en lugar de permitir al usuario la selección de paquetes para su instalación, le permite elegir tareas. Una tarea, según la nomenclatura de Debian, es en esencia un conjunto de paquetes. Por ejemplo, la implementación XFree86 del sistema X Window consiste realmente en un cierto número de paquetes diferentes, incluyendo software vital, controladores para diversos modelos de tarjetas de video y fuentes (entre otras cosas). Debian tiene el concepto de una tarea para permitir a los diseñadores de la distribución agrupar conjuntos de paquetes como XFree86 en forma de un único paquete "virtual". Cuando un usuario selecciona una tarea, en realidad está seleccionando un conjunto de varios paquetes Debian .deb.

El programa tasksel es bastante simple, y utiliza apt-get y después dpkg para manejar la instalación de los paquetes que forman las tareas. Al igual que sucede

con dselect, Red Hat Linux y Slackware Linux no ofrecen ninguna funcionalidad similar. (Red Hat Linux permite a los usuarios seleccionar grupos de paquetes de software parecidos a las tareas de Debian durante el proceso de instalación, pero estos grupos no están disponibles una vez que la instalación se ha completado.)

Entrar en detalles

Las secciones anteriores dieron una visión resumida de las prestaciones ofrecidas por los diversos programas que configuran el sistema de empaquetado de Debian. Las siguientes entrarán en más detalles y mostrarán algunos de los usos habituales del sistema de empaquetado. Aún así, cada programa es bastante complicado, así que si desea detalles completos debería comprobar las páginas del manual y otra documentación sobre estas herramientas proporcionadas por Debian GNU/Linux.

Las siguientes secciones se centran en mostrar las maneras más comunes en que usará el sistema de empaquetado de Debian en su conjunto, en lugar de en la mecánica de los programas por separado. Llegado este momento, probablemente tenga una idea bastante acertada de lo que hacen exactamente herramientas como dpkg y rpm (desempaquetar un montón de archivos, apuntar adónde fueron a parar en una base de datos para que puedan ser eliminados más tarde, y tal vez ejecutar un guión). En lugar de repetir el mismo material que podría conseguir al leer una página de manual, voy a centrarme en la distribución, no sólo en las herramientas.

Administración de paquetes con el sistema de Debian

Incluso aunque pueda leer las páginas del manual para entender cómo usarlas, las herramientas de empaquetado de Debian ofrecen mucha más potencia y flexibilidad de lo que puede resultar inmediatamente obvio a partir de las páginas del manual. Cada una de las siguientes secciones se centra en una tarea común específica y delinea el procedimiento general para esa tarea, para ayudarle a dar el salto desde saber cómo se ejecutan los programas a saber cuándo ejecutarlos. Es importante tener en mente que con el sistema de empaquetado de Debian, hay más de un modo de hacer casi todo, así que mucho de este material se reduce a una cuestión de preferencias.

En general es posible gestionar paquetes Debian en dos niveles: en el nivel de dpkg, y el de apt-get y dselect. Esta sección comentará cada caso, mostrando brevemente los mecanismos de la administración de paquetes e indicando cuándo y por qué querría utilizar cada nivel.

Trabajo con dpkg

La herramienta dpkg de Debian es la herramienta de menor nivel en el sistema de empaquetado. Opera al nivel del paquete individual y puede llevar a cabo las

típicas órdenes de empaquetado como instalar, eliminar, y diversas preguntas. La tabla 6.2 resume algunas de las operaciones más comunes que ofrece dpkg; si quiere más detalles, debería consultar la página de manual para dpkg, u otra documentación.

Tabla 6.2. Operaciones comunes de dpkg.

Comando	Comando RPM equivalente	Función
`dpkg -i <nombre_de_fichero>`	`rpm -i <nombre_de_fichero>`	Instala un archivo de paquete Debian.
`dpkg -r <paquete>`	`rpm -e <paquete>`	Elimina un paquete instalado.
`dpkg -l`	`rpm -qa`	Muestra la lista de todos los paquetes instalados.
`dpkg -l <paquete>`	`rpm -q <paquete>`	Comprueba si un paquete está instalado y proporciona su nombre completo.
`dpkg -L <paquete>`	`rpm -ql <paquete>`	Genera la lista de los archivos instalados por un paquete en particular.
`dpkg -S <archivo>`	`rpm -qf <archivo>`	Identifica el paquete que posee un archivo que se ha instalado.
`dpkg -p <paquete>`	Ninguno	Determina si el paquete indicado se encuentra disponible para ser instalado.

Como puede ver, usar dpkg para administrar paquetes individuales es bastante simple. Si ha leído los capítulos 4 y 5, probablemente le resulte familiar, puesto que es muy similar a los mecanismos equivalentes de Red Hat Linux y Slackware Linux. En los sistemas Debian, probablemente sólo utilice dpkg directamente para instalar paquetes individuales que consiga de algún sitio distinto de la instalación original o un medio de actualización, como una pieza específica de software que haya descargado de un sitio web. La mayor parte del tiempo, probablemente utilice dselect o apt-get, que se comentarán más adelante en esta sección.

Nota: Globalizar por diversión y productividad

El término "globalizar" hace referencia a la expansión de comodines como el asterisco (*) utilizados para corresponderse con patrones específicos. El programa dpkg de Debian, como la mayoría de las herramientas de empaquetado, soporta la globalización en los argumentos que se le pasan. Por ejemplo, para generar la lista de todos los paquetes instalados con "libc" en el nombre, puede usar este comando:

```
$ dpkg -l "*libc*"
```

Los asteriscos serán "globalizados" del mismo modo que lo serían nombres de archivos en una shell, y conseguirá una lista de paquetes con "libc" en sus nombres. Este es un truco bastante conveniente, ¡así que asegúrese de usarlo!

Gestionar fuentes de paquetes con apt-get

El único problema al usar dpkg (o rpm) para administrar paquetes en su sistema Debian es que es muy laborioso. Después de todo, si quiere instalar un paquete en particular, ¡primero tendrá que encontrar el paquete! Esto puede implicar introducir y montar un CD (quizás varias veces, si el paquete no está en el primer disco que compruebe) o descargar un fichero de un servidor web o FTP. Incluso si todo lo que hace es instalar un paquete que olvidó seleccionar durante la instalación, aún tendrá que buscar entre los CD o los URL para dar con él. Además, también tendrá que hacer frente a cualquier dependencia que el paquete deseado pueda tener.

Para estos casos, Debian GNU/Linux incluye apt-get, que gestiona los medios de instalación en su lugar. Ya que apt-get mantiene una lista de los medios fuente que contienen paquetes y una lista de los paquetes que hay realmente disponibles, todo lo que necesita hacer es solicitar que apt-get encuentre e instale el paquete deseado. La tabla 6.3 muestra un resumen de los usos más habituales de apt-get. Una vez más, debería consultar la página del manual si necesita información específica.

Tabla 6.3. Operaciones comunes de apt-get.

Comando	Función
`apt-get install <paquete>`	Localiza e instala el paquete indicado en la lista de paquetes disponibles.
`apt-get remove <paquete>`	Desinstala el paquete instalado y cualquier paquete que dependa de él.
`apt-get update`	Refresca la lista de paquetes disponibles (una vez que la lista de fuentes se ha actualizado).
`apt-get upgrade`	Actualiza todos los paquetes del sistema que tengan disponibles nuevas versiones.

Las dos primeras funciones de apt-get (instalar y eliminar paquetes) son muy sencillas. Sólo tendrá que introducir el comando `apt-get install foo` y apt-get comprobará su lista de fuentes y paquetes disponibles en busca de un archivo Debian llamado "foo". Si encuentra uno, lo instalará, resolviendo al mismo tiempo cualquier diferencia que haya encontrado. A la inversa, `apt-get remove foo` desinstalará el paquete. Sin embargo, el resto de las prestaciones no están tan claramente definidas.

Actualizar paquetes instalados

Una propiedad notable de apt-get mostrada en la tabla 6.3 es que la función `upgrade` no se usa sobre paquetes específicos. En su lugar, es un comando de ámbito global que actualiza todos los paquetes para los que haya actualizaciones disponibles.

A simple vista, podría parecer una restricción extraña, ya que convierte la actualización de paquetes individuales en algo difícil. Sin embargo, si lo piensa, una vez que dispone de un sistema que se actualiza automáticamente rara vez va a necesitar preocuparse por los paquetes individuales, así que esto pasa a no suponer ningún problema en absoluto.

Refrescar la lista de paquetes disponibles

Otra entrada de la tabla 6.3 que puede no resultar inmediatamente obvia es el comando `update`. Este comando se diferencia del comando `upgrade` que se acaba de describir. El comando `update` refresca la lista de paquetes disponibles, pero no actualiza realmente ningún paquete.

Típicamente, ejecutará `apt-get update` después de actualizar el archivo /etc/apt/sources.list, o cuando un sitio FTP o web actualiza su lista de paquetes disponibles. Una vez que apt-get es consciente de las actualizaciones que existen para los paquetes instalados, se puede usar el comando `upgrade` para que realice el trabajo.

Comparación entre apt-get y up2date

Si ha leído el capítulo 4, probablemente haya encontrado similitudes entre apt-get y el sistema up2date de Red Hat. De hecho, son bastante parecidos, pero apt-get es bastante más flexible, sencillamente porque up2date sólo es realmente práctico cuando se usa con el sitio propio de Red Hat, mientras que apt-get le permite utilizar cualquier número de fuentes. (El trabajo que se esté realizando siempre puede modificar esto en el futuro, por supuesto.)

Al fin y al cabo, la herramienta apt-get es sólo una cuestión de conveniencia. Sin embargo, es una gran conveniencia; después de que haya empleado algo de tiempo gestionando los paquetes de unos pocos sistemas Linux a mano, probablemente se sentirá agradecido por todas las ayudas que pueda encontrar. Una vez que se acostumbre a usar apt-get, probablemente no se preocupará nunca más por usar dpkg directamente. Enseguida se acostumbrará a ejecutar comandos como `apt-get install paquete` y a dejar que el sistema encuentre el paquete, lo descargue o le pida que introduzca el CD apropiado y gestione todas las dependencias en su lugar.

Administración de paquetes con dselect y tasksel

En ocasiones, podrá necesitar instalar un cierto número de paquetes de una sola vez, o no estará demasiado seguro del nombre específico que tiene un paquete. A veces ni siquiera sabrá si existe el software que cubre una necesidad en particular, prescindiendo del problema del nombre. Las herramientas dselect y tasksel están orientadas a resolver este asunto.

Uso de dselect

El programa dselect se encuentra pronto en la vida de un sistema Debian GNU/Linux, ya que es la herramienta que se utiliza para seleccionar paquetes individuales durante la instalación. El programa dselect es realmente sólo un envoltorio basado en menús que agrupa todas las prestaciones comentadas hasta ahora en este capítulo.

Los administradores pueden utilizar dselect para navegar por una lista de todos los paquetes disponibles, y ver instantáneamente cuáles están instalados y cuáles no. Al modificar la lista de selecciones, muchos paquetes pueden instalarse o eliminarse de un solo paso. Para estos casos, trabajar con dselect puede resultar más conveniente que usar apt-get para administrar cada paquete individualmente.

Uso de tasksel

El programa tasksel ofrece el nivel más grueso de granularidad. Una tarea es una lista de paquetes (u otras tareas). Por ejemplo, la tarea XFree86 incluye un número de paquetes individuales, como bibliotecas, controladores de tarjetas de video y fuentes. El programa tasksel puede usarse para instalar tareas adicionales una vez que se ha completado la instalación.

Por ejemplo, un sistema puede haberse configurado originalmente como algún tipo de servidor, y por eso no se habrá instalado XFree86 en absoluto. Si se desea modificar el uso de ese sistema para que sea ahora un sistema de escritorio, habrá que instalar los paquetes asociados con XFree86. Por supuesto, esto puede hacerse manualmente mediante apt-get, pero hay implicados una gran cantidad de paquetes. Debido a eso, podría ser más fácil utilizar dselect para seleccionar todos los paquetes de XFree86 para su instalación, pero incluso en ese caso tiene que darles caza a lo largo de la lista. Con tasksel, sin embargo, se puede instalar rápida y fácilmente el conjunto completo de paquetes XFree86 (es decir, la tarea XFree86).

Configuración de paquetes Debian

Los sistemas de empaquetado son muy buenos cuando se trata de instalar archivos y actualizarlos y desinstalarlos cuando resulta necesario. La mayoría también son capaces de gestionar los archivos de configuración, aunque en un grado variable. El sistema RPM de Red Hat, por ejemplo, permite a los creadores de paquetes designar algunos archivos como archivos de configuración. Cuando administra más tarde estos archivos, RPM tiene cuidado de preservarlos y nunca sobrescribir los archivos particularizados de un administrador. En cambio, el sistema de Debian incluye un soporte más amplio a la configuración de paquetes.

Debian incluye el paquete debconf. Este paquete es un conjunto de bibliotecas y utilidades que implementa un protocolo que los diseñadores de paquetes pueden utilizar para manejar la configuración de sus paquetes. Este protocolo supone un acuerdo estandarizado que controla la comunicación entre programas; el protocolo debconf esencialmente permite a los creadores de paquetes hacer preguntas a los usuarios de un modo estandarizado. Las utilidades debconf y su interfaz de usuario recogen la información necesaria en beneficio del paquete y pasan la información a

los guiones de instalación incluidos en el paquete. Los guiones pueden entonces hacer aquello que sea necesario, como generar archivos de configuración.

Un ejemplo de un paquete que utiliza debconf es el paquete del sistema X Window (que instala XFree86). XFree86 necesita una cantidad bastante grande de información para configurar el servidor X para su funcionamiento con el hardware del usuario. Red Hat escribió un programa llamado Xconfigurator para su distribución, pero Debian utiliza las herramientas debconf. Las similitudes entre las dos herramientas son bastante significativas, al menos ambas utilizan una interfaz basada en menús de texto; también preguntan cosas similares y generan archivos parecidos (lo que tiene sentido, ya que son distintas herramientas para configurar el mismo paquete).

Uso de debconf para reconfigurar paquetes

Los usuarios y administradores frecuentemente necesitan reconfigurar paquetes una vez que la instalación ha terminado. Por ejemplo, una estación de trabajo puede actualizarse con una nueva tarjeta de video, con lo que necesitaría la reconfiguración de XFree86. Para dar soporte a esto, debconf proporciona dos herramientas: dpkg-preconfigure y dpkg-reconfigure. Hay también una tercera herramienta, debconf-show, que muestra la información de configuración actual para un paquete.

Los dos comandos de configuración (`dpkg-preconfigure` y `dpkg-reconfigure`) son bastante parecidos. En esencia, dpkg-preconfigure se utiliza para recoger información sobre un paquete antes de su instalación, y es más útil durante el proceso de instalación; de hecho, la mayoría de los usuarios rara vez lo invocarán directamente. El programa dpkg-reconfigure, sin embargo, básicamente lleva a cabo las mismas tareas pero puede reconfigurar programas ya instalados. Por ejemplo, el comando `dpkg-reconfigure xserver-common` puede usarse para reconfigurar aquella estación de trabajo después de la actualización de la tarjeta de video. Del mismo modo, `debconf-show xserver-common` mostrará las variables de configuración que utiliza XFree86 y sus valores actuales.

Configuración de debconf

El propio programa debconf tiene algunos componentes de configuración. Los componentes principales son el directorio /etc/apt/apt.conf.d y el archivo /etc/debconf.conf. Cada uno está descrito en el texto que viene a continuación; sin embargo, la mayoría de los usuarios no tendrán que enfrentarse a ellos muy a menudo, si es que alguna vez llegan a hacerlo. Simplemente debería ser consciente de su existencia, por si acaso los necesita alguna vez.

El directorio /etc/apt/apt.conf.d lo utiliza el sistema de empaquetado apt (que incluye a apt-get, dpkg, y otros programas más) como un enganche para ejecutar programas cuando se invocan las herramientas apt. Este directorio contiene archivos cuyos contenidos son utilizados por las herramientas de empaquetado durante sus operaciones.

El archivo /etc/debconf.conf configura el propio sistema debconf. Este archivo realmente sólo especifica rutas hacia archivos de la base de datos que se utilizan

para almacenar información de configuración. El sistema debconf "recuerda" respuestas que el usuario introdujo con anterioridad para preguntas hechas por paquetes mediante debconf, almacenándolas en una base de datos independiente de los contenidos de los paquetes. Si un paquete se elimina y después se reinstala (o simplemente se actualiza), debconf echa mano de los datos de configuración de su base de datos y utiliza esos valores en la reejecución de los guiones de configuración. Para anular este comportamiento, debería utilizar el programa dpkg-reconfigure.

Actualización del sistema

Hasta ahora he descrito un cierto número de herramientas que ofrecen una amplia variedad de modos de trabajar con el sistema de gestión de paquetes de Debian. Como probablemente haya notado, en general es parecido a otros sistemas, aunque el de Debian pueda ser un sistema más rico y sofisticado. Sin embargo, hay un aspecto que aún no he tocado, y es la habilidad de Debian para ser actualizado fácilmente mediante el sistema de gestión de paquetes.

Para la mayoría de las distribuciones, una actualización es un proceso bastante intensivo. Típicamente tendrá que arrancar el sistema a partir de un disquete o un CD que ejecutará un programa de propósito especial que actualiza el sistema a partir del medio de instalación escogido. Los sistemas Debian, sin embargo, se encargan de esto de un modo bastante diferente.

La clave está en la lista de paquetes disponibles. La mayoría del tiempo, configurará su sistema para usar un conjunto de fuentes que es en gran medida estático, pero que en ocasiones contiene correcciones de errores, de seguridad y demás. En estas ocasiones, típicamente actualizará su instalación con los nuevos paquetes siguiendo un proceso similar a este:

1. Si es necesario, actualizará el archivo /etc/apt/sources.list para reflejar cualquier medio nuevo. Por ejemplo, podría haber conseguido un CD que contenga paquetes actualizados y necesitaría añadir el nuevo CD mediante el comando `apt-cdrom`.
2. Ejecutará `apt-get update` para provocar que apt-get sea consciente de cualquier paquete actualizado.
3. Ejecutará `apt-get upgrade` para provocar que apt-get obtenga e instale cualquier paquete actualizado disponible.

Este procedimiento funciona bien cuando todo lo que necesita hacer es actualizar unos pocos paquetes. Sin embargo, en algún momento probablemente quiera actualizar un sistema Debian a una versión principal nueva. Este caso, si lo piensa, no es en realidad tan diferente del caso anterior; sólo necesita reemplazar los paquetes actualmente instalados con las versiones más nuevas. Las diferencia está únicamente en el alcance de la actualización: en una actualización a una versión principal, una gran cantidad (¡quizás todos!) de paquetes cambian al mismo tiem-

po, casi seguramente con cambios significativos en los contenidos de cada paquete. Administrar una auténtica actualización como esa no es sencillo, así que la mayoría de las distribuciones se limitan a producir nuevos CD, y el usuario sólo tiene que reiniciar el sistema para realizar la actualización. No obstante, con Debian puede utilizar un procedimiento como este:

1. Actualice /etc/apt/sources.list, eliminando todas las referencias a los medios de distribución actuales y añadiendo las entradas apropiadas para la nueva versión principal. (Por ejemplo, actualice los URL para una instalación vía web o añada los nuevos CD para una instalación local.)
2. Ejecute `apt-get update` para refrescar la lista de los paquetes disponibles y sus versiones.
3. Ejecute `apt-get dist-upgrade` para actualizar realmente la distribución. Este comando provoca que apt-get entre en un modo alternativo de actualización válido para manejar los problemas de dependencia potencialmente complicados que surgirán durante la actualización del sistema completo.
4. Cuando se complete la actualización, reinicie el sistema para completar la instalación si es necesario.

Resulta inmediatamente obvio que este procedimiento para la actualización de una distribución completa es muy similar al procedimiento seguido para realizar actualizaciones menores de una versión particular de la distribución. Aún así hay que reiniciar el sistema (si se actualizó el núcleo, por ejemplo) por supuesto, pero de todas formas es un proceso bastante elegante.

Manipulación de archivos de paquete Debian

Todos los paquetes sobre los que ha estado leyendo tienen que proceder de alguna parte, así que es lógico preguntarse cómo se han creado. Además, algún día puede querer o necesitar ver los contenidos de un paquete Debian, o extraer un archivo específico de él, o en cualquier caso enfrentarse a él a un nivel muy básico. Para apoyar en todas estas tareas, Debian incluye un programa conocido como dpkg-deb que gestiona paquetes Debian individuales. La tabla 6.4 resume el uso de dpkg-deb. Como siempre, consulte la página del manual si quiere detalles más completos.

Como muestra la tabla 6.4, dpkg-deb es más o menos equivalente funcionalmente a RPM. Sin embargo, se diferencia de un modo bastante sustancial en lo que sucede cuando se crea realmente un archivo. Recuerde del capítulo 4 que los paquetes RPM se construyen esencialmente a partir de guiones especiales contenidos en archivos .spec (de especificación); estos archivos contienen toda la información que se necesita para recrear un RPM binario a partir del código fuente limpio.

En contraste, los paquetes Debian se construyen con el comando `dpkg-deb --build`, que crea el paquete reuniendo los contenidos de directorio actual y proce-

sando algunos archivos de control nombrados de un modo especial que se encuentran dentro del subdirectorio DEBIAN. De hecho, el enfoque de Debian de la creación de paquetes se parece más al de Slackware que al de Red Hat.

Tabla 6.4. Operaciones comunes de dpkg-deb.

Comando	Comando RPM equivalente	Función
`dpkg-deb --build`	`rpm -ba`	Crea un nuevo paquete.
`dpkg-deb --info <nombre_de_archivo>`	`rpm -qpi`	Extrae información general de un paquete.
`dpkg-deb --contents <nombre_de_archivo>`	`rpm -qpl`	Muestra los contenidos de un paquete.

Administración del código fuente

Algunos lectores pueden estar preguntándose si Debian ofrece un soporte para los paquetes fuente de un modo similar al que RPM de Red Hat utiliza para soportar los archivos .src.rpm. La respuesta es "algo parecido". Como los paquetes de Debian se crean sencillamente para archivar los contenidos de un directorio en particular, es trivial construir un archivo .deb que contenga código fuente. Sin embargo, no existe el concepto formal de un paquete fuente Debian; el código fuente de los paquetes Debian se distribuye como archivos *tarball* estándar (es decir, archivos empaquetados y comprimidos) que contienen las fuentes junto con archivos que contengan los parches necesarios para construir el software para Debian GNU/Linux.

Nota

Véanse los capítulos 4 y 7 para más información sobre los RPM fuente.

Instalación de paquetes no Debian

Como se mencionó en el capítulo 4, probablemente le sea más sencillo encontrar una aplicación en particular empaquetada como RPM que como paquete Debian. Por ello, los usuarios de Debian frecuentemente se encuentran en la engorrosa situación de querer instalar un paquete binario de software que está distribuido en un formato diferente (no Debian).

Para hacer frente a este asunto, Debian GNU/Linux incluye el programa alien. Esta es una herramienta que realiza conversiones entre formatos de empaquetado.

Por ejemplo, permite a los usuarios de Debian convertir un archivo RPM en un paquete Debian. Así los usuarios pueden instalar posteriormente ese paquete en sus sistema Debian (siempre que se cumplan las dependencias del software, por supuesto).

Sin embargo, ya que cada formato de paquete posee sus propias peculiaridades, están obligados a soportar que alien no pueda manejar correctamente algunos paquetes. No es una solución perfecta, pero definitivamente resulta de ayuda. Los usuarios que se encuentren en esta situación pueden desear comprobar el funcionamiento de alien.

Mecanismo frente a interfaz de usuario

Como probablemente ya haya notado, el sistema de administración de paquetes de Debian es sustancialmente diferente de los de Red Hat Linux y Slackware Linux. Mientras que los sistemas de Red Hat y Slackware automatizan más o menos el proceso de instalar físicamente (o gestionar de otro modo) los paquetes, el sistema de Debian lleva la administración de paquetes a otro nivel.

El mejor modo de ilustrar esta circunstancia es comparar los procesos de instalación de Debian GNU/Linux y Red Hat Linux. Durante la instalación de Red Hat Linux, el usuario selecciona uno de entre un conjunto de clases de instalación (por ejemplo, Servidor, Estación de Trabajo, o Equipo de Escritorio). Estas opciones son plantillas que instalan una lista de paquetes personalizada para la clase seleccionada.

Los usuarios pueden ajustar esta lista tras la instalación, pero en ese punto todo se reduce a encontrar el paquete necesario en un CD o en Internet y a instalarlo después manualmente.

Debian GNU/Linux emplea un enfoque similar, pero con un giro interesante. El instalador de Debian en realidad sólo instala un conjunto muy reducido de paquetes (lo justo para arrancar el sistema). Entonces se pide al usuario que utilice el programa dselect para construir una lista de paquetes que instalar, seleccionándolos de una lista separada de paquetes disponibles.

En un momento dado, sólo un subconjunto de los paquetes disponibles se instalará en el sistema. Este enfoque también facilita la actualización del sistema con correcciones de errores, e incluso actualizarlo a una versión principal; el proceso es de una naturaleza muy fluida.

Por supuesto, toda esta flexibilidad tiene un precio llamado complejidad. Hay gente que es entusiasta de apt-get y lo conveniente que resulta su uso, mientras que otros defienden con la misma vehemencia que RPM es un formato de empaquetado superior. Aún hay otros, por supuesto, que siguen la filosofía KISS ("*Keep It Simple, Stupid*", en castellano "Deja que sea simple, tonto") y prefieren el mecanismo mucho más lineal de Slackware.

Hay defensores de cada uno de estos sistemas de empaquetado, incluso aunque comparten una gran cantidad de prestaciones. Al fin y al cabo, los usuarios deberían usar lo que más les guste.

Nota: ¿Qué herramienta es mejor?

Hasta ahora ha leído sobre tres herramientas de administración de paquetes diferentes: RPM de Red Hat, el sistema .deb de Debian y el sistema básico .tar.gz de Slackware. También ha leído acerca de una variedad de herramientas que añaden una capa de funcionalidad a estos sistemas, ofreciendo características como la transparencia de red y diversas interfaces de usuario. Si hay algo que debiera recordar, sin embargo, es que hay más de un modo de hacer lago. Gran parte de la "cultura" de un determinado sistema operativo está ligada al modo en que maneja las actualizaciones; por ejemplo, la introducción del sistema Windows Update de Microsoft supuso un evento importante para esa comunidad, a pesar del hecho de que las distribuciones Linux han estado ofreciendo un mecanismo similar durante años. Como otro ejemplo, el sistema operativo similar a Unix FreeBSD dispone de un método propio muy potente que implica la actualización del código fuente del sistema operativo al completo (¡y de todas las aplicaciones!) mediante CVS, y la posterior recompilación del conjunto. (¡Puede tomar días realizar una actualización completa!)
Trabajar con una distribución Linux (o cualquier otro sistema operativo) significa que va a tener que aprender el modo en que las cosas se llevan a cabo en ese sistema. Así que pruebe y evite atarse demasiado a una distribución en particular, porque probablemente sea de un modo completamente distinto en el sistema que utilizará mañana.

Guiones de arranque del sistema

Debian GNU/Linux utiliza el modelo SysV para los guiones de inicio del sistema. El capítulo 3 comentó en primer lugar los modelos de guiones de inicio de SysV y BSD, y los capítulos 4 y 5 describieron SysV y BSD en detalle. Como Red Hat, Debian utiliza un modelo SysV extensamente, mientras que Slackware se basa principalmente en BSD con algunos enganches de conveniencia para los guiones escritos para utilizar el modelo SysV. En esencia, todo lo mencionado en el capítulo 4 dentro de la sección "Guiones de arranque del sistema" sobre los guiones de inicio también es cierto para Debian, con la salvedad de las pocas excepciones comentadas a continuación.

Encontrar los directorios SysV

Debian GNU/Linux, como la mayoría de los sistemas que utilizan el modelo SysV, tiene sus directorios relevantes inmediatamente dentro del directorio /etc. Es decir, los directorios SysV son /etc/init.d para los propios guiones y de /etc/rc1.d

a /etc/rc6.d para los directorios específicos para los niveles de ejecución. El guión que gestiona todo esto es /etc/rc.d/rc.

Red Hat Linux 7.3, sin embargo, coloca todos los directorios SysV dentro del directorio /etc/rc.d. Es decir, en lugar de /etc/init.d, Red Hat utiliza /etc/rc.d/init.d. Por cuestiones de compatibilidad, Red Hat Linux tiene enlaces simbólicos desde /etc a sus correspondientes directorios en /etc/rc.d. (En una versión futura, Red Hat Linux cambiará a la práctica habitual de poner los directorios inmediatamente dentro de /etc.) Por ello, toda la información del capítulo 4 sobre el modelo de guiones de inicio de SysV también se aplica a Debian GNU/Linux, excepto por los cambios en la ubicación de directorios.

Trabajar con las herramientas de Debian

Red Hat Linux incluye diversas herramientas para automatizar el proceso de administrar los guiones de inicio SysV (llamadas chkconfig y service). Debian GNU/Linux no incluye estas herramientas, pero proporciona alternativas similares: update-rc.d e invoke-rc.d.

Uso de update-rc.d

La herramienta update-rc.d de Debian es parecida a la herramienta chkconfig de Red Hat. Mientras que chkconfig es un programa binario, update-rc.d es un guión escrito en Perl. Estas herramientas tienen opciones en línea de comandos muy diferentes, pero, aún así, realizan funciones muy parecidas. La tabla 6.5 resume el uso de update-rc.d, aunque si quiere información completa debería consultar la página del material, como siempre.

Nota

Para seguir convenientemente esta sección, debe haber leído el contenido del capítulo 4 acerca del modelo de guiones de inicio SysV.

Tabla 6.5. Uso de update-rc.d.

Comando	Función
`update-rc.d -f <servicio> remove`	Elimina el servicio de los directorios de configuración de todos los niveles de ejecución.
`update-rc.d <servicio> start <orden> <niveles_de_ejecución>`	Configura el servicio para que se inicie en el orden especificado para una lista de niveles de ejecución.
`update-rc.d <servicio> stop <orden> <niveles_de_ejecución>`	Configura el servicio para que se termine en el orden especificado para una lista de niveles de ejecución.

La mayor diferencia entre update-rc.d y la herramienta chkconfig de Red Hat es que se pretende que update-rc.d pueda configurar los enlaces con los guiones de servicios del directorio /etc/init.d para todos los niveles de ejecución en masa, mientras que chkconfig es capaz de configurar los servicios para cada nivel de ejecución de manera individual. En esencia, esto sólo significa que cada vez que modifique un nivel de ejecución mediante update-rc.d tendrá que modificarlos todos. Por ejemplo, la siguiente secuencia de comandos es equivalente al comando `chkconfig --level 2345 inetd off`:

```
$ update-rc.d -f inetd remove
$update-rc.d inetd stop 20 0 1 2 3 4 5 6
```

El primer comando elimina los enlaces al guión de servicio /etc/init.d/inetd para todos los niveles de ejecución; el parámetro `-f` provoca que update-rc.d se ejecute incluso aunque el guión inetd exista. El segundo comando crea después guiones de parada para cada nivel de ejecución con una prioridad de servicio de 20; por ejemplo, esto creará el enlace simbólico /etc/rc3.d/K20inetd para inhabilitar inetd para el nivel de ejecución 3. (Dese cuenta de que tiene que añadir explícitamente los niveles de reinicio y parada [6 y 0] aunque casi todos los servicios deberían estar siempre configurados para detenerse.) Como otro ejemplo más, los comandos que vamos a mostrar a continuación habilitarían inetd para los niveles del 3 al 5 y lo inhabilitarían para todos los demás:

```
$ update-rc.d -f inetd remove
$update-rc.d inetd start 20 3 4 5 . stop 20 0 1 2 6
```

El programa update-rc.d de Debian puede resultar bastante confuso al principio, pero una vez que lo comprenda, verá lo bien que funciona.

Uso de invoke-rc.d

El equivalente para Debian del comando `service` de Red Hat es `invoke-rc.d`. Los dos comandos son idénticos en la mayor parte de los casos; por ejemplo, los comandos `service inetd start` e `invoke-rc.d inetd start` tienen el mismo efecto (que es ejecutar el servicio inetd; el comando equivalente para detenerlo sería `invoke-rc.d inetd stop`). El comando `invoke-rc.d` tiene algunos parámetros en línea de comandos extra que puede inspeccionar, pero en general los dos comandos son equivalentes.

Diferencias en /etc/inittab

Como se comentó en el capítulo 3, el archivo que configura principalmente el proceso de arranque del sistema es /etc/inittab. Este archivo gobierna los guiones que se ejecutan durante el arranque del sistema. El archivo /etc/inittab que se instala en los sistemas Debian GNU/Linux debería resultar bastante familiar, ya que se parece al de Red Hat, como se comentó en el capítulo 4.

Hay algunos aspectos importantes en que el archivo /etc/inittab de Debian se diferencia de los de Red Hat y Slackware.

- El nivel de ejecución por defecto de Debian es el 2.
- Debian no configura el gestor del sistema X Windows (es decir, el componente gráfico de entrada en el sistema) desde /etc/inittab. (Lea la sección "Idiosincrasias de Debian GNU/Linux" más adelante en este capítulo para obtener más información.)
- Los niveles de ejecución 2 y 3 son idénticos; los niveles 4 y 5 son idénticos.
- Los niveles de ejecución 4 y 5 sólo inician una consola virtual; los niveles 2 y 3 inician seis.

Llegado a este punto, debería poder verificar los puntos anteriores mediante una inspección de /etc/inittab; si necesita detalles sobre cómo hacer eso, vea los capítulos 4 y 5, en los que los archivos /etc/inittab de Red Hat y de Slackware se analizan en detalle.

Dejando a un lado estos puntos, Debian usa un sistema de arranque SysV bastante típico. Para conocer los detalles esenciales, vea el comentario sobre la implementación del sistema de arranque SysV de Red Hat Linux en el capítulo 4; la mayoría de los detalles se aplicarán a Debian GNU/Linux, aunque el contenido de los propios guiones pueda ser diferente.

Bibliotecas centrales del sistema

Debian GNU/Linux es una distribución Linux moderna, así que no descubrirá ninguna gran sorpresa en este apartado. Las bibliotecas centrales incluidas con la versión 3.0 se listan en la tabla 6.6. Si compara esta lista con la de las tablas 4.9 y 5.2, que listan las bibliotecas centrales para Red Hat Linux 7.3 y Slackware Linux 8.0, verá que son bastante similares.

Tabla 6.6. Bibliotecas centrales de Debian GNU/Linux 3.0.

Paquete	Versión	Comentario
XFree86	4.1.0	Sistema X Window.
glibc	2.2.5	Bibliotecas del sistema principales en tiempo de ejecución.
bash	2.05	*Bourne Again Shell*; shell principal del sistema (utilizada por el usuario *root*).
gdbm	1.7.3	Bibliotecas de bases de datos de archivos.
fileutils	4.1	Utilidades de manipulación de archivos.

Lo único que podría hacer que se detuviera a pensar un momento es que Debian tiene nombres alternativos para algunos de estos paquetes. Por ejemplo, el nombre del paquete RPM glibc de Red Hat es simplemente "glibc", mientras que el nombre del paquete equivalente de Debian es "libc6". Esto es, por supuesto, una simple cuestión de nombrado y no implica ninguna modificación en los contenidos de los paquetes.

Sólo necesita ser consciente de esto cuando vaya a utilizar las herramientas de Debian (o, del mismo, las herramientas de cualquier distribución): si no puede encontrar un paquete según su nombre, no suponga que no se encuentra allí. Si no puede encontrar el nombre del paquete, entonces es práctico utilizar el comando `dpkg -S` para encontrar el paquete a partir de un archivo que sepa que le pertenece. Por ejemplo, `dpkg -S /lib/libc.so.6` revelará que el nombre de paquete de Debian es "libc6". En otras palabras, hay un motivo para que programas como dpkg soporten modos de preguntas de ese estilo; aprenda a hacer un buen uso de ellos.

El sistema X Window y el escritorio

Debian GNU/Linux 3.0 incluye XFree86 4.1.0. Una vez más no habrá sorpresas. Slackware Linux 8.0 también soporta XFree86 4.1.0; Red Hat Linux 7.3, por su parte, soporta una versión ligeramente posterior de XFree86, específicamente la 4.2.0.

Debian GNU/Linux 3.0 también incluye los entornos de escritorio GNOME y KDE, al igual que hacen Red Hat Linux y Slackware Linux. Debian incluye la versión 1.4.0 de GNOME, y la versión 2.2 de KDE. Estas son versiones ligeramente más antiguas que las incluidas con Red Hat Linux 7.3, pero son coetáneas de las de Slackware Linux 8.0.

Probablemente el único modo significativo en que la instalación de X de Debian difiere de las de Red Hat y Slackware es en el modo en que inicia el gestor de escritorio. (Recuerde de los capítulos 3 al 5 que el gestor de escritorio X es la pantalla gráfica de entrada al sistema que permite a los usuarios entrar a través de X en lugar de por la consola.) Hay más que comentar sobre este asunto en la sección "Idiosincrasias de Debian GNU/Linux".

Aplicaciones de usuario

Hay disponible una variedad extremadamente amplia de aplicaciones para los sistemas Debian. El Proyecto Debian históricamente ha luchado para crear paquetes Debian de tantas aplicaciones software como sea posible. Siempre que ha sido posible, se han incluido estos paquetes en la distribución, ya sea en los CD o a través de la red. (Es decir, se ofrecen en la lista de paquetes disponibles, como se describió con anterioridad en este capítulo, en la sección "Administración de paquetes con el sistema de Debian".) De hecho, ¡hay disponibles tantos paquetes Debian

que incluso puede conseguir una instalación de Debian en formato DVD! Claramente, a Debian GNU/Linux no le faltan paquetes de instalación.

Sin embargo, debido a la manera en que el sistema de empaquetado de Debian gestiona las fuentes de instalación, no hay realmente un modo sencillo de distinguir entre los CD. Es decir, mientras que Red Hat ofrece una clara delimitación entre los CD principales y los diversos CD auxiliares (como PowerTools y otros), Debian GNU/Linux consiste en un confuso revoltijo de archivos diseminados entre diversos CD. (No es realmente tan caótico, pero a veces puede parecérselo a los nuevos usuarios.) En otras palabras, Debian no tiene ningún CD adicional del mismo modo en que los tienen otras distribuciones. Por eso, no existen ediciones Debian, ni discos extras, ni nada parecido. Sin embargo, esto no importa realmente, después de todo, la única razón para poner nombre a los CD de ese modo es facilitar a los usuarios hacerse a la idea de que CD contiene el paquete deseado. Ya que Debian descarga al usuario del trabajo de encontrar los paquetes en los CD, hay pocos motivos para no juntar simplemente todos los archivos en un gran conjunto de varios CD. (Aún así, una vez más, la organización no es aleatoria; los paquetes más vitales se encuentran en el primer par de CD, para que los usuarios que descarguen las imágenes de los CD y creen sus propias copias no tengan que descargarlos todos si no los necesitan.)

Idiosincrasias de Debian GNU/Linux

Esta sección describirá algunos aspectos o características que son únicos de Debian GNU/Linux. Hasta ahora, ha visto que Debian GNU/Linux es bastante similar a Red Hat Linux y a Slackware Linux, pero que existen unas pocas diferencias menores en determinadas áreas. Esta sección delinea algunas diferencias algo más significativas de las que necesitará ser consciente cuando utilice un sistema Debian GNU/Linux.

Visita al directorio /etc

Generalmente, encontrará en /etc todo lo que esperaría encontrar. Aún así, hay unos pocos puntos que merecen que les prestemos algo de atención. La tabla 6.7 lista algunos subdirectorios de /etc notables acerca de los que debería saber. La lista no incluye los subdirectorios comunes definidos por el FHS (como /etc/X11 y /etc/init.d), sólo los directorios específicos de Debian. Cada entrada de la tabla 6.7 se comenta en el texto que está a continuación, excepto en el caso de /etc/alternatives, al que se le dedica una sección completa más adelante.

El directorio /etc/default

Este directorio contiene archivos que fijan valores de variables utilizadas por otros guiones. Típicamente, cada vez que un programa o guión necesite tener asig-

nados unos valores que sean específicos de la instalación, las variables se situarán en un archivo en el directorio /etc/default y el guión o el programa leerá ese archivo para determinar sus parámetros. De este modo, los administradores no tienen que editar el propio guión para ajustar sus parámetros. (Realmente, esta es una solución diferente para el mismo problema que inspiró a Red el uso del enfoque del "directorio de depósito de configuraciones" comentado a lo largo del capítulo 4.)

Tabla 6.7. Subdirectorio significantes de /etc en Debian GNU/Linux.

Directorio	Contenidos
/etc/default	Archivos con información de configuración para diversos programas.
/etc/network	Archivos relacionados con la configuración de las interfaces de red.
/etc/dpkg	Información de configuración de la herramienta dpkg.
/etc/alternatives	Enlaces simbólicos que escogen programas por defecto en una lista de varios programas.

Por ejemplo, los núcleos 2.4 soportan la funcionalidad iptables, que permite a los administradores crear cortafuegos, entre otras cosas. (Vea el capítulo 16 para encontrar un ejemplo de un cortafuego basado en Linux que utilice iptables.) Utilizar iptables necesita que se fijen algunas variables. Por eso, el guión suministrado por Debian que configura las iptables lee estos valores desde /etc/default/iptables en lugar de pedir que el administrador modifique el guión /etc/init.d/iptables directamente.

Ya que los únicos archivos de este directorio serán dejados allí por paquetes individuales, los contenidos de este directorio variarán obviamente según los paquetes que se instalen. Cada vez que instale un paquete Debian nuevo, asegúrese de comprobar /etc/default para ver si contiene un archivo que modificar. Por supuesto, siempre debería leer toda la documentación que trate acerca del software.

El directorio /etc/network

Este directorio es parecido a /etc/default en que contiene información de configuración para el sistema de trabajo en red. Sin embargo, los archivos relacionados con la red se sitúan aquí para mantener todos en un único lugar y evitar que se confundan con otros archivos en una extraña mezcla. Este directorio es análogo a /etc/sysconfig/network en los sistemas Red Hat Linux, aunque sus contenidos son diferentes.

El directorio /etc/network lo instalan los paquetes ifupdown y netbase. La tabla 6.8 lista los contenidos típicos del directorio.

Tabla 6.8. Contenidos del directorio /etc/network.

Archivo o directorio	Propósito
if-down.d	Contiene guiones que se ejecutarán cuando una interfaz de red se corte.
if-post-down.d	Contiene guiones que se ejecutarán después de que una interfaz de red se corte.
if-up.d	Contiene guiones que se ejecutarán cuando una interfaz de red se active.
if-pre-up.d	Contiene guiones que se ejecutarán antes de que una interfaz de red se active.
ifstate	Contiene información sobre si cada dispositivo de red está activo o no.
interfaces	Contiene información de configuración de red para todas las interfaces.
options	Habilita o inhabilita diversas características de red opcionales.
spoof-protect	Habilita o inhabilita la protección frente a ataques *spoofing* (de suplantación de personalidad) de red.

Generalmente, los archivos con los que probablemente interactuará serán interfaces y options. El archivo interfaces contiene información de red detallada (como dirección IP; máscara de red, servidor pasarela, o si utilizar el protocolo de configuración dinámica de máquinas [*Dynamic Host Configuration Protocol*, DHCP] para cada interfaz que el sistema utiliza.) Mediante la edición de los contenidos de este archivo, puede controlar la mayoría de los aspectos de la configuración de red de su sistema. El único otro archivo con el que seguramente necesitará interactuar es options, que habilita o inhabilita opciones generales de red adicionales, como si se debería configurar el núcleo para funcionar como un encaminador de red mediante la retransmisión de paquetes IP. Se proporciona una página de manual excelente acerca del archivo /etc/network/interfaces; simplemente ejecute el comando `man interfaces` y dispondrá de toda la información que necesita para configurar las interfaces de red de su sistema.

Los otros archivos de /etc/network son para casos en los que los administradores necesitan capacidades más avanzadas. Por ejemplo, los directorios if-down.d, if-up.d, if-pre-up.d e if-post-down.d ofrecen a los administradores un modo de enganchar guiones al proceso de activar o desactivar interfaces de red para tomar notas acerca de esos eventos, por ejemplo. (Estos directorios implementan el enfoque habitual de configuración de config.d que se comentó en el capítulo 4.) Del mismo modo, el archivo /etc/network/spoof-protect permite a los administradores fijar parámetros que se utilizan para detectar ataques de red de suplantación.

(Este archivo puede no seguir siendo necesario por los núcleos recientes.) Finalmente, el archivo /etc/network/ifstate es interno a los guiones ifup e ifdown y se utiliza para indicar qué interfaces están activados y cuales no. Una vez más, en general no necesitará trabajar con estos archivos, pero es aconsejable saber que están ahí por si alguna vez los necesita.

El directorio /etc/dpkg

El directorio contiene archivos que configurar los programas dpkg y dselect. Los archivos dpkg.cfg y dselect.cfg pueden utilizarse para fijar opciones por defecto para dpkg y dselect, de modo que no tengan que introducirse en la línea de comandos cada vez que se ejecuten los programas. El directorio origins contiene archivos que describen dónde se originó la distribución en particular. (Típicamente lo utilizan varias organizaciones para ofrecer variantes de la distribución básica de Debian.) Los archivos shlibs.default y shlibs.override pueden usarse para ajustar las bibliotecas compartidas que utiliza dpkg (como sucedería con la actualización de bibliotecas centrales del sistema como glibc). Esta es un área bastante críptica y arcana, ¡y no debería jugar con ella a no ser que sepa lo que está haciendo!

Comprensión del directorio /etc/alternatives

El directorio /etc/alternatives, recogido en la tabla 6.7, establece programas por defecto para los casos en que pueda haber instalado más de un programa que cumpla un determinado propósito.

Por ejemplo, probablemente el editor de texto Unix más omnipresente sea el programa vi. Sin embargo, hay un gran número de implementaciones libres de vi, como VI iMproved (vim), nvi, y elvis. Típicamente, cada vez que uno de estos programas se instala, se crea un enlace simbólico para ejecutar el clon de vi cada vez que un usuario quiera ejecutar el comando `vi`. (Es decir, /bin/vi suele ser un enlace simbólico a uno de estos programas.) Sin embargo, una determinada instalación de Debian podría tener varios o todos estos instalados al mismo tiempo. Cuando sucede eso, ¿cómo escoge el clon de vi que se esconde como el "auténtico" vi? Obviamente, simplemente debe escoger uno de ellos y hacer que sea el programa por defecto.

Esta situación se da con una cierta frecuencia. Además del ejemplo de vi que se acaba de comentar, hay múltiples implementaciones del programa awk (incluida gawk), yacc (incluida bison y byacc), y muchos otros programas. De hecho, sucede tan a menudo que el Proyecto Debian creó /etc/alternatives y el sistema de actualización de alternativas para enfrentarse a este problema.

El propósito del directorio /etc/alternatives es simplemente indicar qué clon de un programa en particular debería ser el "auténtico" en el sistema. El programa real en el sistema de archivos es en realidad un enlace simbólico al directorio /etc/alternatives, y el archivo que se encuentra allí es a su vez un enlace simbólico al clon. Volviendo al ejemplo de vi, suponga que están instalados nvi y vim, y que se va a configurar a vim como el vi auténtico del sistema. Para implementar esta configuración, /etc/alternatives/vi se enlaza para que apunte a /usr/bin/vim. El

archivo /usr/bin/vi, mientras tanto, está enlazado simbólicamente para que apunte a /etc/alternatives/vi. El archivo /usr/bin/vi jamás cambia; para cambiar a un clon diferente, sólo se cambia el archivo /etc/alternatives/vi.

El directorio /etc/alternatives realmente se encarga de más que sólo del propio programa; también gestiona las páginas man. (Después de todo, ¡no hace ningún bien tener a vim como el clon de vi elegido pero obtener la página del manual de nvi cuando ejecute el comando `man vi`!) El mecanismo utilizado para las páginas del manual es muy parecido al de los programas; las "auténticas" paginas man en los directorios /man y /usr/man se enlazan simbólicamente para apuntar a /etc/alternatives, donde a su vez están enlazadas simbólicamente con la página man de un clon.

Como probablemente pensará, administrar todos estos enlaces simbólicos se volvería rápidamente muy engorroso. Para enfrentarse a este problema, Debian GNU/Linux también incluye el programa update-alternative, que gestiona los enlaces simbólicos de /etc/alternatives en su lugar de un modo muy parecido al que usa la herramienta update-rc.d (y chkconfig en Red Hat Linux) para gestionar los enlaces simbólicos de arranque de SysV. El uso del programa update-alternatives es muy sencillo, así que lea su página del manual (ejecute `man update-alternatives`) si quiere detalles completos.

Configuración del gestor de visualización X

El gestor de visualización X es el programa que permite a los usuarios entrar directamente en X mediante una interfaz gráfica de usuario. Diversos programas pueden actuar como gestores del escritorio X; más notablemente, KDE y GNOME tienen uno cada uno, kdm y gdm. Esta sección comentará cómo configurar Debian GNU/Linux para que utilice kdm o gdm como su gestor de visualización X.

Tanto Red Hat Linux como Slackware Linux inician el gestor de visualización X desde /etc/inittab. Específicamente, Red Hat tiene un programa conocido como prefdm que es invocado directamente desde /etc/inittab y es responsable de escoger el gestor de escritorio por defecto (normalmente gdm o kdm) que se utilizará; de un modo similar, Slackware ejecuta un guión de shell muy simple (/etc/rc.d/rc.4) desde /etc/inittab que escoge el gestor de visualización dinámicamente y lo ejecuta. Sin embargo, Debian hace las cosas de un modo algo distinto.

Nota

Véanse los capítulos 4 y 5 para obtener más detalles sobre las técnicas empleadas por Red Hat y Slackware.

Debian inicia el gestor de visualización como un servicio SysV normal. Cada paquete que contiene un gestor de visualización instala su propio guión SysV en /etc/init.d. Por ejemplo, gdm y kdm instalan /etc/init.d/gdm y /etc/init.d/kdm,

respectivamente. Entonces Debian es capaz de tratar estos guiones como servicios iniciales SysV estándar, y no necesita disponer de una entrada en /etc/inittab para ellos.

Un problema potencial de este enfoque puede surgir si se instalan tanto gdm como kdm; el sistema tiene que escoger entre ambos ya que no se pueden ejecutar los dos al mismo tiempo. Para evitar conflictos de este tipo, ambos guiones de inicio están escritos de modo que comprueben el archivo /etc/X11/default-display-manager, que contiene la ruta del gestor de visualización por defecto oficial del sistema. Los guiones de inicio de gdm y kdm comprueban este archivo para determinar si deberían ejecutarse; como resultado, sólo uno de ellos se ejecutará cada vez.

Por eso, para cambiar el gestor de visualización por defecto en Debian GNU/Linux, debe editar el archivo /etc/X11/default-display-manager. Este archivo contiene una única línea, que es la ruta hacia un programa gestor de escritorio (por ejemplo, `/usr/bin/gdm`). Sencillamente cambie el nombre de este programa por algo como `/usr/bin/kdm`.

Ahora que ya sabe como hacer esto manualmente, debería conocer la herramienta que automatiza el proceso. Cuando instala realmente dos gestores de visualización en el sistema, el administrador de paquetes de Debian le pedirá que seleccione cuál desea utilizar. Para poder cambiar de gestor de visualización simplemente tiene que hacer que dpkg reconfigure uno de los paquetes; el siguiente comando mostrará cómo hacer esto. Cuando lo ejecute, se le pedirá que seleccione de nuevo su gestor de visualización. Por supuesto, todo lo que hace este programa es modificar los contenidos de /etc/X11/default-display-manager, así que sólo es una cuestión de qué método es más rápido. El comando que hace que dpkg reconfigure el gestor de visualización es el siguiente:

```
$ dpkg-reconfigure gdm
```

Garantizar la seguridad del sistema

Al igual que con Slackware Linux, Debian GNU/Linux deja en ejecución muchos servicios en una instalación por defecto. Es muy importante eliminar los servicios innecesarios, o si no se arriesga a dejar a su sistema comprometido ante un atacante, especialmente si el sistema está conectado a Internet. Esta sección comentará cómo poner cerrojos a su sistema.

Cierre de servicios SysV innecesarios

Típicamente, si instala un paquete Debian que tiene que ser ejecutado por el sistema de inicio SysV, se configurará para ejecutarse por defecto. Como resultado, dependiendo de cómo de cuidadosamente seleccione sus paquetes cuando instale el sistema, puede tener un gran número de servicios ejecutándose que no necesitará. Algunos de estos, como el programa portmap y el statd de NFS, tienen todo un historial de vulnerabilidades, así que tiene sentido desactivarlos.

En Red Hat Linux, se puede usar de un modo sencillo el programa chkconfig para ver qué servicios SysV están configurados para ejecutarse y cuales no para cada nivel de ejecución. Lamentablemente, no resulta tan sencillo en Debian GNU/Linux. Para poder decir si se ha configurado un servicio para un determinado nivel de ejecución, hay que comprobar los enlaces simbólicos en el directorio del nivel de ejecución apropiado. Por ejemplo, para ver qué es lo que está configurado para ejecutarse en el nivel 2, tendría que ejecutar el comando `ls /etc/rc2.d`; cualquier archivo que comience por "K" está configurado para detenerse en ese nivel de ejecución, y cualquiera que comience por "S" está configurado para arrancarse.

Nota

Para obtener detalles completos sobre cómo funciona el sistema de inicio SysV, véase la sección "Guiones de arranque del sistema" en el capítulo 4.

Una vez que ha visto lo que está funcionando, tiene que decidir qué necesita ejecutarse y qué no. Generalmente, si no necesita un servicio, debería cerrarlo, para impedir exponerse a vulnerabilidades potenciales futuras en un programa que ni siquiera necesita. Para configurar realmente un servicio para que no se ejecute, puede tanto modificar los enlaces simbólicos a mano (como se comentó en el capítulo 4), como utilizar el programa update-rc.d descrito anteriormente en este capítulo.

Eliminación de servicios inetd

Debian utiliza una implementación tradicional de inetd. (La misma que Slackware Linux; en contraste con Red Hat Linux que utiliza el programa alternativo xinetd.) Como en Slackware, la configuración por defecto de inetd de Debian viene con un gran número de servicios habilitados. Como siempre, es una buena idea inhabilitar los servicios que no necesite.

Uso del programa update-inetd

Para inhabilitar servicios, puede usar las mismas técnicas comentadas en la sección "Configuración de inetd" del capítulo 5. Sin embargo, Debian también proporciona una herramienta adicional llamada update-inetd. Este programa es muy parecido a chkconfig de Red Hat y a update-rc.d de Debian, excepto en que funciona estrictamente sobre los servicios inetd configurados en el archivo /etc/inetd.conf. Este programa hace bastante más sencillo añadir entradas nuevas a /etc/inetd.conf (como cuando instale un nuevo servicio inetd en el sistema; véase el capítulo 12 para un ejemplo sobre esto) al igual que habilitar e inhabilitar los servicios existentes. (Por ejemplo, puede querer inhabilitar el servicio CVS mientras lleva a cabo labores de mantenimiento, pero no querrá eliminar completamente la entrada.) El programa update-inetd también le permite eliminar fácilmente entradas de /etc/inetd.conf.

Cuando se ejecuta update-inetd, simplemente realiza las acciones comentadas en el capítulo 5, de manera que no tenga que realizarlas manualmente. El programa update-inetd se comentará con más detalle en el capítulo 12, por eso no lo comentaré aquí de manera extensa. No obstante, para usarlo para inhabilitar servicios inetd por motivos de seguridad, siga este ejemplo, que inhabilita el servicio finger:

```
$ update-inetd --disable finger
```

Simplemente repita este comando para cada servicio que quiera inhabilitar. Si descubre que no necesita que se esté ejecutando ningún servicio inetd en absoluto, entonces debería inhabilitar complementa el servicio inetd en el modelo SysV mediante la técnica mencionada en la sección anterior.

Nota

Para ver qué servicios están activados actualmente, tendrá que inspeccionar los contenidos de /etc/inetd.conf y encontrar las líneas que definen servicios. Véase el capítulo 5 para obtener más detalles.

Derivados de Debian GNU/Linux

Hay un cierto número de distribuciones comerciales que se han basado en Debian. Sin embargo, muchas han desaparecido recientemente, dejando sólo a la propia Debian y algunas otras. El Proyecto Debian mantiene actualmente una lista de derivados en su sitio web en www.debian.org/misc/related_links.

Esta lista seguramente no esté completa, y puede incluir algunos derivados que desde entonces se hayan abandonado, pero muestra que hay toda una comunidad en ebullición rodeando a Debian, no sólo en la distribución central, sino también entorno a otros trabajos relacionados.

Resumen

Bien, lo ha conseguido, ha recorrido su tercera distribución. Este capítulo trató a cerca de Debian GNU/Linux, presentándole los diversos aspectos del sistema. Sin embargo, ya había visto otras dos antes que esta, así que una gran parte de esto comienza a resultar familiar. De hecho, probablemente no haya podido impedir darse cuenta de lo mismo que yo mientras escribía este capítulo: ¡me encontré diciendo tantas veces ".... justo como en Red Hat Linux y Slackware Linux"!

Sin embargo, Debian GNU/Linux es claramente su propia distribución, por eso también ha leído una gran cantidad de material acerca de características únicas de Debian, herramientas, y modos de hacer las cosas. Ahora debería sentirse bastante

cómodo con un sistema Debian; debería saber cómo realizar las tareas más comunes, y tener una buena idea de dónde mirar cuando necesite saber cómo hacer algo que no se comente en este libro.

De hecho, ya sabe esto mismo para tres distribuciones. Por extensión, debería tener una gran ventaja para comprender cualquier distribución con la que pueda encontrarse. ¡Mucho de este material probablemente incluso se aplique a sistemas no Linux! Por supuesto, ese era el objetivo de la segunda parte.

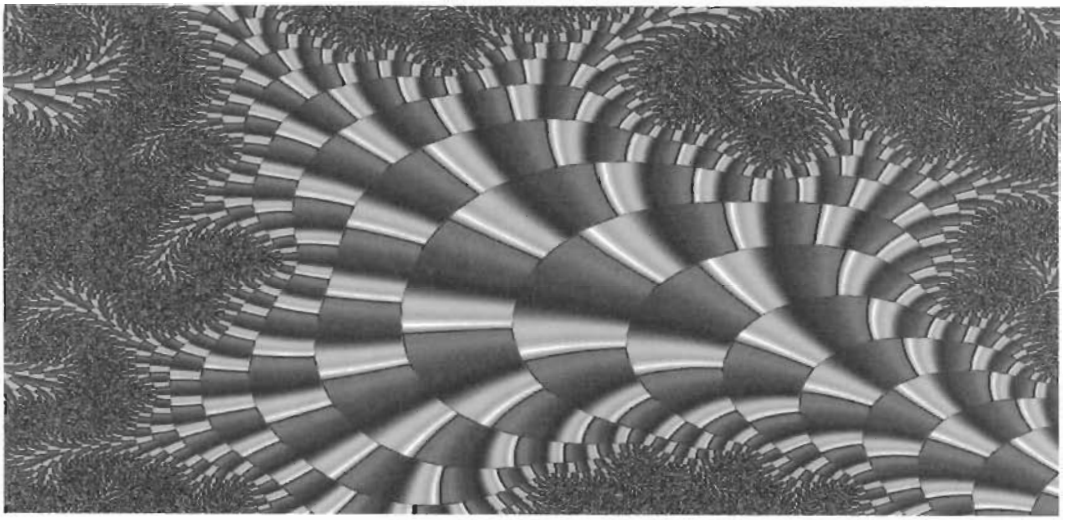
Nota: Aprendizaje de una nueva distribución Linux con diez sencillos pasos

Un ejercicio muy útil en este momento es pararse a pensar en lo que ha aprendido. Suponga que se le entrega un ordenador con una distribución Linux que jamás ha visto antes y tiene que utilizarla y administrarla. ¿Cómo haría para llevar a cabo esta tarea? La lista que viene a continuación es una breve lista de comprobación que podría utilizar, pero antes de que la lea, tómese un momento y piense sobre cómo lo haría usted mismo. Resultará un ejercicio productivo y gratificante, ya que pondrá en juego los conocimientos que ha adquirido. Aquí está la pequeña receta para el aprendizaje de una nueva distribución Linux, en diez sencillos pasos:

1. Descubrir qué gestor de paquetes utiliza, si utiliza alguno. (¿Debian? ¿RPM? ¿Algo parecido al de Slackware?)
2. Comprobar el archivo /etc/inittab y ver qué prestaciones están configuradas cuando inicia el sistema. (¿Hay consolas virtuales? ¿Funciona X?)
3. Ver si el sistema utiliza el modelo de guiones de inicio BSD o SysV.
4. Navegar por el sistema de archivos y buscar directorios cuyo propósito no se reconozca. (¿Con qué fidelidad sigue al FHS? Cuidado con las cosas únicas como /etc/profile.d de Red Hat y /var/log/packages de Slackware.)
5. Si es un sistema SysV, buscar las herramientas que pueden ayudar a gestionar los guiones.
6. Mientras tanto, buscar otras herramientas útiles, como Xconfigurator de Red Hat o update-alternatives de Debian.
7. Si el sistema permite ver la lista de paquetes instalados, comprobar qué versiones del software están instaladas, y mirar los contenidos de cualquier paquete que parezca interesante o no resulte familiar. (Mediante los comandos `rpm -qa` en Red Hat; `dpkg -l` en Debian; `cat /var/log/packages` en Slackware.)
8. Asegurarse de que el sistema no ejecuta ningún servicio innecesario, e inhabilitar cualquiera que esté ejecutándose.

9. Si hay que instalar software propio, hacerse a la idea de cuál es el mejor lugar, de manera que se acomode bien al sistema. (¿Utiliza el sistema / opt para algo o puede usarlo el propio usuario?)
10. Durante el uso del sistema, buscar patrones y generalizaciones que definan el "carácter" de la distribución; a largo plazo, es muy más aconsejable trabajar en el sentido de la distribución que en su contra.

¿Cuánto se asemeja su lista a esta? ¿Qué es lo que haría de otro modo? Mientras piensa en estas cuestiones, tómese un momento para disfrutar de esa sensación de logro; ¡ha aprendido muchas cosas! Disfrútelo. Cuando se encuentre preparado, vaya a la tercera parte, y comience a aprender cómo instalar software para personalizar su sistema.



7
Instalación y configuración de software

Una vez que tiene un sistema Linux listo y en funcionamiento y se ha familiarizado con su manejo y administración, puede comenzar a personalizar el sistema. Este capítulo trata el problema general de las instalaciones de software. Aprenderá acerca de algunas de las técnicas de instalación y configuración más comunes y generales utilizadas por muchos paquetes de software diferentes. Tras la lectura de este capítulo, habrá visto una gran variedad de técnicas que cubren la mayoría de las instalaciones. Los seis capítulos siguientes proporcionan más detalle en forma de ejemplos específicos de los conceptos comentados en este capítulo.

Parte de la información dada en este capítulo es bastante básica. Los usuarios que han instalado software por su cuenta en el pasado pueden desear saltarse mucha de la información ofrecida. Puede aprender gran parte del contenido con un poco de experiencia. Sin embargo, algunas secciones contienen información útil sin importar su nivel de experiencia, así que puede picotear un poco y escoger entre los diversos temas aquí cubiertos.

Comienzo

El proceso de instalación de software no puede comenzar realmente hasta que el usuario haya completado sus deberes. Una vez que se ha identificado un paquete de software, un futuro usuario debe solucionar algunos problemas logísticos básicos relacionados con la licencia del software y sus características y con él procedimiento de instalación.

Sólo porque el software ocupe un cierto nicho de utilidad, no significa que cubra las necesidades de un usuario. El servidor web Apache, por ejemplo, es extremadamente robusto y completo. Sin embargo, Apache no se diseñó para tener un alto rendimiento; en su lugar, se centra en la corrección y la gran cantidad de características.

Si el principal requisito de un usuario para un servidor web es la velocidad, entonces Apache podría no ser la elección correcta. El primer paso real para instalar software es sencillamente determinar si cumple con las necesidades del usuario.

Lectura de la documentación

Un usuario debería realizar sus deberes e investigar el software a través de su documentación. Los puntos de partida obvios son el sitio web del software (si tiene uno), o el casi omnipresente archivo *"README"* (o en ocasiones su versión en castellano, "LEAME"). Esta documentación también suele incluir instrucciones de instalación (normalmente en un archivo llamado *"INSTALL"*), así que es una buena idea revisarlos también. Sólo después de familiarizarse con el software debería un usuario comenzar la instalación con los pasos siguientes: es un auténtico incordio perder una hora instalando y configurando el software sólo para descubrir que no se adecua bien a las necesidades. Lamentablemente, a veces no hay modo de saberlo hasta una vez consumados los hechos. Algunos comienzos erróneos serán algo normal.

Aceptar la licencia

El capítulo 2 comentó el software de código abierto, que es fundamental para Linux y los sistemas basados en Linux. Además, el capítulo 2 presentó algunos ejemplos de licencias de código abierto y describió brevemente algunas de las diferencias entre ellas. La mayoría del software que los usuarios encuentran para Linux es de código abierto y está cubierto por alguna de estas licencias. Sin embargo, existen otras licencias, y por supuesto hay software comercial (de código no abierto) para Linux. El primer paso, que a veces se pasa por alto, es que el usuario descubra si es legal usar el software deseado y si la licencia es aceptable. Para el software de código abierto, hay típicamente un archivo llamado *"LICENSE"* (en castellano "LICENCIA") que contiene todo lo que el usuario necesita saber; los paquetes de software comercial frecuentemente tienen formularios en los que se acepta una licencia mediante el procedimiento de hacer clic en los que se solicita explícitamente que el usuario acepte. En cualquier caso, el usuario debe revisar la licencia antes de la instalación.

Después de que el usuario haya revisado la documentación del software, comprobado que cubre sus necesidades, y aceptado la licencia, es hora de instalar realmente software. La siguiente sección describe los pasos básicos de instalación, y proporciona algunos trucos y consejos.

Instalación de software en seis pasos sencillos

Instalar software en un sistema Linux es bastante fácil, tras algunos intentos y un poco de experiencia. Sin embargo, a largo plazo el mayor desafío es mantener todo ordenado y en buen estado. Es fácil que las cosas pierdan el control y se enreden como un gran ovillo de lana en lugar de formar un sistema funcional. Este capítulo describe por encima los siguientes pasos de la instalación de software:

1. Selección entre una instalación fuente o binaria.
2. Decidir una ubicación.
3. Construir el software.
4. Configuración de los parámetros del software.
5. Configuración del entorno de usuario.
6. Conexión con el sistema operativo.

Este capítulo describirá cada uno de estos pasos en detalle. Los pasos no ofrecen instrucciones detalladas sobre cómo instalar software; para detalles como esos, los usuarios deberían consultar la documentación del software u otro libro de referencia de carácter más introductorio. Como el resto de este libro, se hace énfasis sobre reglas simples, consejos, técnicas y patrones generales. Después de la lectura de estas secciones, los usuarios tendrán una comprensión más amplia sobre lo que sucede exactamente cuando instalan software.

Elección entre una instalación fuente o binaria

El primer paso para instalar un nuevo paquete de software es, por supuesto, conseguir el paquete. Sin embargo, ya que la amplia mayoría del software usado en los sistemas Linux es de código abierto, el código fuente suele estar disponible. Esto quiere decir que su primera decisión es si instalar archivos binarios precompilados o construir sus propios binarios a partir del código fuente. La opción apropiada va a variar de una situación a otra dependiendo del hardware y la distribución particular que se use y de las necesidades del usuario. Esta sección comentará cómo tomar esta decisión, basada en factores como las necesidades de la arquitectura de la plataforma y la configuración de software.

Comprensión de las plataformas hardware y las familias de procesadores

La mayor parte de los microprocesadores pertenecen a una "familia" de procesadores relacionados. A medida que los fabricantes desarrollan chips más nuevos y rápidos, generalmente aumentan las prestaciones existentes en chips más viejos, y por eso los últimos procesadores de un fabricante en particular están

relacionados (pero no son siempre compatibles al cien por cien) con otros chips de la familia. Dos ejemplos de familias de microprocesadores son la omnipresente familia x86 de Intel y la familia Alpha de procesadores RISC de Compaq. Los últimos chips de estas familias son simples extensiones de chips anteriores; por ejemplo, el procesador Pentium 4 de Intel es compatible hacia atrás con el procesador original 8086, mientras que el chip 21364 EV67 de Compaq lo es con chips como el 21269 EV56 (un chip anterior).

Sin embargo, cualquier chip subsiguiente en una familia generalmente introduce nuevas extensiones al procesador. Por ejemplo, Intel introdujo las instrucciones de extensión multimedia (*Multimedia Extension*, MMX) en su línea Pentium de procesadores. Uno de los trabajos del compilador es tomar el código fuente y optimizarlo para que use las características de un determinado procesador. Obviamente, un código que se haya compilado para utilizar las instrucciones u optimizaciones disponibles en un chip específico no va a funcionar en un procesador que no tenga esas características, incluso aunque los chips sean de la misma familia.

El impacto de estas optimizaciones varía. En un caso como el de las instrucciones MMX, el código simplemente podría no funcionar en todos los procesadores que no tienen las instrucciones MMX y fallar con un error. En un caso más sutil, el código optimizado para funcionar en un Alpha EV6 utiliza las características de temporización y programación de instrucciones del procesador EV6. Aunque estas instrucciones se ejecutarían físicamente en un procesador EV5 más antiguo, podrían no ejecutarse óptimamente ya que el EV5 podría tener distintos requisitos de tiempo.

¿Qué tiene que ver esto con instalar software? Bueno, si el único paquete binario disponible está compilado para una versión ligeramente distinta de su propio procesador, podría funcionar de manera no óptima, o no funcionar en absoluto. Esto varía según la arquitectura; en el mundo Intel generalmente no es muy importante, pero los usuarios de Alpha frecuentemente deciden construir su propio software para asegurarse de que tienen las optimizaciones apropiadas para su sistema específico. Por eso, según su plataforma puede desear construir el software por sí mismo a partir del código fuente en lugar de confiar en el paquete prefabricado por algún otro.

Opciones de configuración del software

A veces no son las optimizaciones del compilador de lo que se preocupa el usuario, sino de las opciones de configuración de software. Esto es, una pieza de software podría soportar diferentes características o comportamientos que el usuario puede querer activar o desactivar cuando el software se construye a partir del código fuente. Estos cambios son independientes de cualquier optimización que el compilador pueda realizar, y se denominan configuraciones en tiempo de compilación (o de construcción).

Lo que es importante comprender acerca de los parámetros en tiempo de compilación es que no se pueden cambiar después; si un usuario elige inhabilitar una

característica en tiempo de compilación, no se puede añadir después sin recompilar todo o parte del software. Por ejemplo, el servidor web Apache tiene la habilidad de cargar dinámicamente módulos que ofrecen diversas prestaciones; sin embargo, un usuario puede desear inhabilitar este comportamiento y construir Apache como un único programa, sin este soporte para módulos. Si un administrador decide hacer esto y posteriormente desea utilizar una biblioteca dinámica, habrá que recompilar Apache.

Un determinado paquete de software se habrá compilado con un conjunto específico de características. Si el usuario de este paquete desea usar un conjunto específico de características, entonces necesitará construir el software a partir del código fuente. Este asunto es independiente del formato de empaquetado; aunque el software binario se empaquete en formato RPM, Debian u otro formato, aún habrá sido compilado con un determinado conjunto de opciones en tiempo de compilación. Cuando se considere la instalación de un paquete de software binario, los usuarios deberían consultar la documentación de quien quiera que haya preparado el paquete para ver qué opciones se han habilitado. Si la configuración del paquete es inadecuada, el administrador tendrá que construir su propia copia a partir del código fuente.

Toma de la decisión

Lamentablemente, no hay realmente una regla general para tomar la decisión de instalar desde la fuente o desde los archivos binarios. La regla varía según la plataforma (como los usuarios de Alpha que prefieren construir a partir del código) y la necesidad. (Un usuario puede estar más preocupado por la seguridad en un servidor de producción que en una estación de trabajo, y por eso puede ser más previsor e inhabilitar características innecesarias del software.) Obviamente, por supuesto, es más fácil y simple instalar a partir de paquetes binarios si resulta aceptable hacerlo, y para la mayoría de las aplicaciones esa política funcionará perfectamente bien. Aún así, alguna vez podría necesitar instalar a partir del código fuente; en esas ocasiones, "lo sabrá cuando lo vea", como dice el dicho. La clave es solamente asegurarse de que sabe lo que necesita de su software antes de instalarlo (eso le dirá si instalar desde el código o las fuentes binarias).

Decidir la ubicación

Una vez determinado cómo se va a instalar el software, lo siguiente es dónde se va a instalar. Esta sección comentará las opciones y algunos consejos para elegir una ubicación para la instalación. La ubicación de un único paquete de software en sistema de archivos no es realmente tan importante (dado que puede acceder adecuadamente al sistema de archivos del ordenador cualquiera que lo necesite), pero es una buena idea estandarizar sobre la política de dónde se instala el software para prevenir los dolores de cabeza del mantenimiento y la actualización posteriores.

La mejor ubicación para instalar software va a variar en función del tipo de software y de la distribución. La mayoría de las distribuciones Linux siguen de

cerca el estándar de jerarquía de sistema de archivo de Linux (FHS) mencionado en primer lugar en el capítulo 3, así que esa documentación puede ser una buena guía para tomar una decisión. Aún así, para la mayoría de los sistemas hay una regla básica que puede aplicarse: si el software viene como un paquete binario preconstruido o un paquete con las fuentes preconfiguradas (como un archivo RPM fuente), deje al empaquetador instalar el software de acuerdo con los valores por defecto del paquete. Si el software se construye a partir del código en bruto (como un archivo ZIP o *tarball*), debería instalarse en un subdirectorio de /usr/local o /opt.

Instalación en /usr/local

Un lugar común para instalar software es /usr/local. Como se comentó en el capítulo 3, el FHS especifica que /usr es esencialmente un espejo del directorio raíz, en el sentido de que ambos tienen /etc, /lib, /bin, y otros subdirectorios. El directorio /usr está preparado para ser utilizado por archivos generales, mientras que el directorio raíz lo está para archivos que son absolutamente cruciales para el sistema durante el arranque. Mientras tanto, la intención del directorio /usr/local es contener software que es específico para un sistema en particular; /usr generalmente sólo contiene software general.

En los sistemas Linux, esto significa esencialmente que /usr y el directorio raíz contienen software administrado por el gestor de paquetes nativo (como RPM de Red Hat o el sistema Debian), mientras que /usr/local es para archivos que se instalan manualmente y esquivan el empaquetador nativo. El empaquetador nativo típicamente maneja conflictos entre paquetes; sin embargo, no puede hacer nada por el software que no conoce, y por eso es útil mantener el software instalado manualmente en /usr/local, que la mayoría de los gestores de paquetes usan rara vez.

Típicamente, el software instalado en /usr/local se coloca en los directorios estándar. Esto es, /usr/local/etc contiene archivos de configuración, /usr/local/bin contiene programas ejecutables, y así. El directorio /opt, comentado en la siguiente sección, suele ser un poco diferente.

Nota: Instalación de software en sistemas multiusuario

A veces podrá encontrar un sistema en que desee instalar algún software personal, pero donde no tenga los privilegios del usuario *root* (superusuario). Por ejemplo, muchos usuarios acostumbrados a los sistemas Linux y a las herramientas GNU que lo acompañan frecuentemente encuentran frustrante utilizar las herramientas menos capaces incluidas por defecto en los sistemas comerciales Unix. Estos usuarios pueden desear instalar las herramientas GNU para su propio uso.

La falta de acceso como *root* le impide instalar software en los lugares habituales, y está restringido a instalar en su directorio local. Una técnica

efectiva en estos casos es simplemente crear un subdirectorio en su directorio local y crear su propio /usr/local u /opt reducido. Por ejemplo, podría crear un directorio llamado "~/usr" e instalar todo su software en él siguiendo el método estándar, de modo que los programas estén en ~/usr/bin, las bibliotecas en ~/usr/lib, y así. Con esta técnica, puede usar todos los trucos tratados en este capítulo, incluso aunque no disponga de acceso como *root*. Si ss verdaderamente ambicioso, ¡podría incluso instalar su propia versión de un gestor de paquetes (como RPM) en su directorio local!

Instalación en /opt

Como /usr/local, el directorio /opt no suele ser utilizado por los gestores de paquetes. Por eso, es un buen lugar para instalar software manualmente, ya que los archivos colocados allí no entrarán en conflicto con archivos gestionados por el sistema. Sin embargo, /opt suele usarse de un modo algo diferente a /usr/local.

El directorio /usr/local típicamente contiene archivos de todos los paquetes mezclados. Esto lo hace más simple de configurar (ya que, por ejemplo, sólo tiene que añadir un único directorio, /usr/local/bin, a la ruta). Sin embargo, también hace algo complicado actualizar un paquete específico. Por ejemplo, si instaló la versión 1.0 de una biblioteca software llamada "mi-software" en /usr/local y después desea actualizarla a la versión 2.0, tendrá que sobrescribir lo que ya está allí. Los usuarios que no deseaban actualizar la biblioteca podrían sufrir el impacto de esto.

La mayoría de los sistemas que utilizan un directorio /opt lo usan para enfrentarse a estos problemas. Solaris de Sun Microsystems parece haber sido el primero en usar /opt extensivamente, y desde entonces muchos otros sistemas han copiado su estilo. Esencialmente, en lugar de mezclar todo el software en un conjunto de directorios compartidos, cada paquete se instala en su propio subdirectorio de /opt. Por ejemplo, la biblioteca software anterior podría haberse instalado en /opt/mi-software para mantenerla separada. Incluso podría haberse instalado en /opt/mi-software-1.0; eso permitiría que la nueva versión se pusiera en /opt/mi-software-2.0. Esto permite actualizaciones más fáciles y el uso de enlaces simbólicos (como un enlace en /opt/mi-software que apunte a la versión actual) para que los usuarios no tengan que hacer referencia a directorios específicos de las versiones.

Nota: Compromisos: /usr/local frente a /opt

Hay pros y contras para el uso ambos directorios. Por una parte, /usr/local hace más simple configurar el sistema, ya que sólo se necesita añadir un directorio a la ruta de las shells para que todos los programas sean accesibles. (Del mismo modo, sólo se necesita añadir un directorio "lib" a la ruta de bibliotecas, etc...) Sin embargo, una vez que se ha mezclado un gran número de archivos,

es difícil gestionarlo. El enfoque de múltiples directorios utilizado en /opt hace más sencillo seguir la pista de qué archivos pertenecen a qué paquetes (¡es una especie de gestor de paquetes para pobres!) Sin embargo, con /opt tiene que enganchar varios directorios (o múltiples rutas, rutas de bibliotecas, y demás) en la configuración del sistema. Esto también puede suponer demasiado esfuerzo de administración. ¿Cuál es mejor? En realidad depende del software; con un poco de experiencia, descubrirá sus propias reglas básicas por sí mismo.

Toma de la decisión

Elegir la ubicación del software es sencillo si se está instalando mediante el gestor de paquetes nativo: simplemente deje que se instale donde quiera y que el gestor se encargue de él. Es algo más complicado cuando tiene que instalarlo manualmente. Un buen criterio para escoger entre /usr/local y /opt es el tamaño o la complejidad del software. Si es un pequeño paquete de utilidades, como, digamos, un cliente SSH (shell segura), entonces probablemente está bien colocarlo en /usr/local. Sin embargo, si es un paquete más grande, como el servidor web Apache o el escritorio KDE, podría desear utilizar /opt para facilitar su actualización posterior. Además, si los usuarios necesitan tener instaladas múltiples versiones de un paquete, como el lenguaje de programación Java, donde los desarrolladores frecuentemente tienen que acceder a varias versiones, /opt es probablemente más fácil de usar.

Construcción del software

Después de que un administrador ha decidido dónde instalar el software y un conjunto de parámetros de configuración en tiempo de compilación, ya puede construir el software. Por supuesto, este paso sólo se aplica al software que se instala a partir del código fuente; en las instalaciones de paquetes binarios preconstruidos se omite este paso. Generalmente, hay dos fases para construir software: especificar opciones en tiempo de compilación y automatizar la construcción.

Especificación de opciones en tiempo de compilación

El código fuente se convierte en un programa real mediante un compilador (como gcc, el compilador de lenguaje C de GNU). Típicamente, el compilador debe invocarse para cada archivo de código fuente. La mayoría de los proyectos de software consisten en docenas o cientos de archivos (o más), y los compiladores son bestias complejas, con cientos de opciones de compilación. Está claro que esto no es algo que los administradores puedan hacer a mano (son simplemente demasiadas teclas). Casi cualquier proyecto de software conocido por la humanidad utiliza algún tipo de mecanismo de guiones para automatizar el proceso de construcción;

casi universalmente , éste es la herramienta *make*. Esta herramienta lee archivos (que se suelen llamar, de un modo suficientemente intuitivo, Makefile) que especifican opciones de compilación e identifican los archivos que se necesita compilar. En la mayor parte de los casos, la construcción del software es tan simple como ejecutar `make` desde el directorio de mayor nivel en el que está contenido el código fuente.

Aún así, antes de que el software se pueda construir, las opciones en tiempo de compilación tienen que haber sido especificadas de algún modo. Está bien que el archivo Makefile se encargue el proceso de compilación, pero ¿cómo sabe qué opciones de compilación desea el usuario? Hay, lamentablemente, un montón de maneras diferentes de lograr esto. Cierto software necesita que el usuario edite el archivo Makefile para fijar algunas opciones (como directorios de instalación o ubicación de las bibliotecas de las que depende el software). Sin embargo, la amplia mayoría del software de código abierto utiliza las herramientas autoconf del proyecto GNU, que se describen con detalle más adelante en este capítulo. Al fin y al cabo, el único modo de descubrir cómo configurar los parámetros de una aplicación es mediante la lectura de su documentación. Esto es por lo que el primer paso, como se mencionó con anterioridad, es leer el archivo *ReadMe*, *Install*, u otra documentación. Como se suele llamar en inglés, "RTFM", o "*Read The Fine Material*" ("Lee el buen material"), ¡o incluso algo más explícito, si no está trabajando con gente educada!

Realizar la compilación

El último paso tras la construcción del software es instalarlo. Una vez más, esto está típicamente automatizado por el archivo Makefile del proyecto. El comando `make` permite a Makefile especificar objetivos (*targets*). Un objetivo (el objetivo por defecto) normalmente construye el software, y un segundo objetivo (normalmente llamado "install") se encarga de la instalación. Por eso, en la mayoría de los casos instalar el software tras su construcción es tan sencillo como escribir `make install`. Sin embargo, otra vez, esto puede variar, y la propia documentación del software tendrá la última palabra.

Configuración de los parámetros del software

Aún no ha terminado la tarea del administrador, ¡incluso después de que el software esté construido e instalado! Frecuentemente, hay actividades adicionales de configuración que hay que realizar una vez instalado el software. Las opciones de configuración que modifican el comportamiento del software mientras se ejecuta se denominan opciones en tiempo de ejecución. Esta sección describirá las maneras más comunes de configurar un sistema en tiempo de ejecución.

Una opción en tiempo de compilación típicamente habilita o inhabilita alguna característica del software, o introduce en el código directamente un parámetro como la ubicación de un archivo necesario. Ya que estas opciones están literalmente compiladas en el software, no se pueden alterar. (Si inhabilita una característica

en tiempo de compilación, ¡simplemente no está ahí para poder ser activada más tarde!) Por el contrario, una opción en tiempo de ejecución fija un valor o modifica un comportamiento para una ejecución específica del programa. Hay una gran cantidad de maneras en que se puede realizar esto y el resto de esta sección describe algunas de las técnicas más habituales.

Parámetros en línea de comandos

El mecanismo más simple y directo para realizar la configuración en tiempo de ejecución son los parámetros en línea de comandos, a veces conocidos como *switches* o *flags* en línea de comandos. Esta no es realmente una auténtica opción de configuración como las otras mencionadas en esta sección, pero su resultado es similar. Incluso el usuario de Unix más verde está íntimamente familiarizado con esta técnica, y casi todos los comandos de ejemplo en este libro utilizan parámetros en línea de comandos.

En los sistemas similares a Unix, los parámetros en línea de comandos típicamente siguen al nombre del comando y están separados por guiones; por ejemplo, el comando de listado de archivos ls puede usar varias opciones en línea de comandos para modificar su salida. La mayoría son opciones de un único carácter y se unen con un único guión; por ejemplo, `ls -la` es equivalente a `ls -l -a`. El proyecto GNU introdujo una variante de esta técnica que utiliza dos guiones y una palabra completa en lugar de un único guión y una letra. Con esto se pretende conseguir una sintaxis más intuitiva y fácil de recordar. Como ejemplo, la versión de GNU de ls dispone de resalte por color que se habilita mediante la opción en línea de comandos `--color=always`. (Típicamente, los programas GNU también ofrecen la forma tradicional guión/letra además de este formato más largo.)

Hay también otros formatos de opciones en línea de comandos. Algunos programas, como el programa del sistema de versiones concurrentes (CVS), que es una herramienta de administración de modificación del software, emplea subcomandos, que son palabras simples. El comando `cvs checkout`, por ejemplo, trae un módulo desde un repositorio CVS. Hay igualmente otras variantes, y varían en complejidad. La página del manual (a la que puede accederse mediante el comando `man cvs`) y otra documentación son la mejor opción, pero resulta útil ser consciente de los patrones.

Herramientas y ayudantes específicos de la aplicación

Una vez que se han configurado las características del software y se ha compilado e instalado en el directorio apropiado, podría necesitar ser configurado para su uso real. Esto se diferencia del primer paso (configuración en tiempo de compilación) en que el software se configura antes de construirlo, para activar o desactivar características y comportamientos. En este paso, el software se configurará para ejecutarse realmente; esto se conoce como configuración en tiempo de ejecución. Por ejemplo, el servidor web Apache puede configurarse tanto para activar como para desactivar el soporte a objetos compartidos dinámicos antes de su compilación, pero en cualquier caso aún tiene que configurarse con los valores necesarios

para que funcione correctamente, como la ubicación de los archivos HTML que va a servir, o el número del puerto TCP/IP que usará.

Hay un gran número de maneras diferentes de configurar software, y paquetes distintos tomarán distintos enfoques. Aún así, generalmente, hay algunas técnicas tradicionales usadas en los sistemas similares a Unix que siguen la mayor parte de las aplicaciones. Las siguientes cinco subsecciones comentarán los modos más comunes de configurar software:

- Archivos de configuración global
- Directorios para depositar archivos de configuración
- Archivos o directorios planos
- Servidores arrancados por inetd
- Bibliotecas de usuario

Algunos son típicos de todos los sistemas similares a Unix, y algunos son más específicos de los sistemas Linux o de distribuciones particulares.

Archivos de configuración global

El modo canónico de configurar software para un sistema similar a Unix es hacer que el software lea uno o más archivos de configuración con un nombre especial que se encuentren dentro del directorio /etc. Por ejemplo, el archivo /etc/sendmail.cf se utiliza para configurar el servidor SMTP sendmail. A veces, las aplicaciones que necesitan más de un archivo de configuración usarán varios archivos dentro de un subdirectorio de /etc. Por ejemplo, por defecto el paquete OpenSSH lee los archivos de configuración de su cliente y servidor de /etc/ssh. (Estas ubicaciones frecuentemente están grabadas en el código en tiempo de compilación, y se pueden modificar si se reconstruye el paquete.) Esta es probablemente la manera más común de configurar software.

Directorios para depositar archivos de configuración

El directorio para poner archivos de configuración es una técnica utilizada de manera extensa por la distribución Linux de Red Hat, y está cobrando igual importancia en otros sistemas. Esta técnica se utiliza cuando un único paquete de software puede tener configuraciones separadas para diferentes escenarios. Entonces el software define un único directorio (típicamente como un subdirectorio de /etc) y lee una configuración de ese directorio por separado para cada escenario. Un escenario puede ser cualquier cosa, desde un usuario diferente a una aplicación de software distinta (cualquier cosa que necesite su propia configuración única). Por ejemplo, el paquete xinetd almacena archivos de configuración en /etc/xinetd.d y tiene un archivo de configuración separado para cada servicio inetd que va a gestionar. En cambio el paquete PAM (*Pluggable Authentication Modules*, Módulos de Autenticación Enlazables) lee archivos de configuración de /etc/pam.d y tiene un archivo aparte para cada servicio o programa para el que administra la autentica-

ción. La ventaja de este enfoque es que el software se puede particularizar para un determinado escenario mediante la modificación de un archivo específico para ese escenario en lugar de tener que modificar el contenido de un único archivo que contenga la configuración para todos los escenarios. Por ejemplo, un nuevo servicio inetd se puede configurar con xinetd al añadir un archivo a /etc/xinetd.d para ese servicio. La alternativa sería hacer que el servicio que se instala modifique un único archivo compartido de configuración, que tiende a provocar errores y podría acabar por destruir la configuración de otro servicio. Como se mencionó anteriormente, esta técnica se usa ampliamente en Red Hat Linux y se trata con más detalle en el capítulo 4.

Truco: Dejar caer un archivo

La técnica de dejar caer archivos de configuración funciona mejor con software que ofrece algún tipo de servicio a otro software. En estos casos, el software que ofrece el servicio no sabe con anterioridad cuantas aplicaciones necesita usar. Por eso, el software simplemente proporciona un "enganche" a otros programas para que añadan sus propios datos de configuración a la mezcla. Busca usos para esta técnica cada vez que un programa no conoce de antemano cuanto se usará. ¡Se está volviendo muy popular!

Archivos o directorios planos

Algunas aplicaciones sencillamente ubican sus archivos de configuración en el mismo directorio en que están instaladas. Normalmente, las aplicaciones de Unix colocan sus archivos de configuración en el directorio /etc, así que esta técnica se encontrará más habitualmente en aplicaciones que han sido adaptadas a Unix desde otra plataforma o que no tienen ningún tipo de requisitos de configuración global en absoluto (como la aplicación de productividad OpenOffice).

Además, los usuarios a veces instalarán software en un directorio específico que no es el mismo que el directorio por defecto. Por ejemplo, si el servidor web Apache se construye a partir de su código fuente y se coloca en /usr/local/apache-2.0, sus archivos de configuración se encontrarán en /usr/local/apache-2.0/etc. (Normalmente, estarían simplemente en /etc.)

Servidores arrancados por inetd

Algunas aplicaciones de servidor no son servidores autónomos, si no servicios administrados y ejecutados bajo demando por el superservidor inetd. Servicios como estos no tienen sus propios archivos de configuración, sino que en su lugar inetd les pasa la información que necesiten cuando se inician. El programa inetd, por su parte, ofrece una manera de especificar información de configuración para cada servicio. Hay varias implementaciones diferentes del servidor inetd, y cada una tiene su propio método de configurar los servicios.

Bibliotecas de usuario

Hay software que no se pretende que sea utilizado por sí mismo, si no que ofrece funcionalidad que otros programas necesitan (por ejemplo, hay software distribuido como bibliotecas usadas por otros programas). Este software puede tener o no necesidades de configuración. Por ejemplo, la biblioteca Dante, que ofrece la funcionalidad de SOCKS5, necesita un archivo de configuración que defina qué servidores y direcciones utiliza Dante, mientras que la biblioteca OpenSSL, que ofrece diversos algoritmos criptográficos, no necesita ninguna configuración en tiempo de ejecución. Las bibliotecas de usuario que necesitan información de configuración siguen alguno de los demás patrones (como los archivos planos de configuración y el directorio de archivos de configuración) comentados en subsecciones anteriores.

Configuración del entorno de usuario

En ocasiones el software necesita (o sencillamente permite) que los usuarios fijen valores de configuración adicionales. Típicamente esta es información específica del usuario como el directorio local, la ubicación de diversos archivos, etc., y suele manejarse al definir variables de shell. Para que el software recién instalado funcione correctamente para un determinado usuario, el usuario puede necesitar fijar algunas variables de entorno en su shell de entrada al sistema.

Los usuarios siempre podrán hacer esto ellos mismos, y este esquema funciona bien para el software que sólo utiliza un pequeño número de usuarios. Sin embargo, el software usado más habitualmente necesita que muchos usuarios fijen variables de entorno, y esto se convierte rápidamente en un engorro administrativo. En estos casos, es necesario que el administrador configure el software globalmente; sin embargo, el modo en que se hace esto varía de sistema a sistema.

Nota

Los entornos de usuario para Red Hat Linux, Slackware y Debian se describen en los capítulos 4, 5 y 6 respectivamente; la información de estos capítulos debería ser suficiente como para hacerse una idea de cómo personalizar cualquier entorno de usuario global de un sistema similar a Unix. En particular, vea el mecanismo de Red Hat en el capítulo 4. Parece estar tomando fuerza incluso entre las aplicaciones que no son de código abierto.

Enganche con el sistema operativo

Para las aplicaciones de usuario final, la instalación está completa una vez que se configura el entorno de usuario. Sin embargo, obviamente, las aplicaciones de

servidor sólo son útiles cuando se están ejecutando, y por eso los servidores necesitan engancharse con el proceso de arranque del ordenador. Cada sistema Unix hace esto según su propio estilo, pero hay dos métodos generales de uso común.

Un método es el conjunto de guiones de arranque utilizado por System V de AT&T (SysV) y el otro es el mecanismo de guiones de arranque equivalente utilizado por BSD. Estos dos sistemas sirven al mismo propósito, pero son muy distintos en la práctica. El enfoque de SysV utiliza la técnica del "directorio para depositar archivos de configuración" descrita anteriormente en este mismo capítulo, mientras que el enfoque de BSD es más simple pero confía en un único archivo. Red Hat Linux utiliza la técnica SysV, mientras que Debian y Slackware utilizan la técnica BSD. Si quiere más información sobre la técnica SysV, vea el capítulo 4; para más información sobre el enfoque de BSD, vea el capítulo 5.

Herramientas de instalación de software

Este capítulo ha descrito una variedad de herramientas y técnicas para la instalación de software. Esta sección describirá las siguientes herramientas y comentará las ofrecidas por la mayoría de las distribuciones:

- Herramientas autoconf de GNU
- RPM de Red Hat
- Archivos *tarball* de Slackware
- Formato dpkg de Debian

Tras la lectura de este material, los usuarios se sentirán familiarizados con el modo en que se construye e instala el software en varias distribuciones. Esta sección probablemente no sea una referencia completa, pero puede verse como una especie de "guía rápida" para utilizar estas herramientas. Los usos más comunes quedarán cubiertos, pero los lectores que necesiten una información más extensa deberían consultar la documentación de las herramientas específicas.

Uso de autoconf de GNU

Muchos paquetes de código abierto utilizan las herramientas autoconf de GNU para configurar las opciones en tiempo de compilación del software. Autoconf de GNU funciona generando archivos Makefile estándar a partir de plantillas producidas por los desarrolladores del software. Las herramientas autoconf analizan un sistema, en busca de la presencia o ausencia de características específicas mediante la compilación de pequeños programas de prueba. Los archivos Makefile que genera se utilizan después para automatizar la compilación real del software.

Puede ser algo esotérico y críptico escribir plantillas para autoconf; por fortuna, sólo los desarrolladores de software tienen que preocuparse por ello y las herra-

mientas son bastante sencillas para los usuarios que sólo están interesados en construir un paquete. Normalmente, habrá un programa llamado "configure" en el directorio de mayor nivel del árbol del código fuente; los usuarios simplemente ejecutarán `./configure` y el programa examinará el sistema y construirá archivos Makefile, de cabecera y otros similares utilizados por la herramienta `make` y por el código fuente.

El programa `./configure` generalmente usará una serie de argumentos. El más común de estos es la opción `--prefix`, que fija el directorio de destino en el que se instalará el software; por ejemplo, el comando `./configure --prefix=/usr/local/mi-software-1.0` configurará el software de modo que se instale en /usr/local/mi-software-1.0. Muchos paquetes soportan sus propias opciones mediante configure; los usuarios pueden comprobar qué opciones están disponibles mediante la ejecución del comando `./configure --help`. Después de que el software se haya configurado así, los usuarios simplemente ejecutan `make`, que construirá el software, y normalmente `make install` para completar la instalación. La documentación de un paquete dado explicará las opciones de configuración en tiempo de compilación específicas para ese paquete, pero la tabla 7.1 proporciona una lista con algunos de los parámetros comunes utilizados por autoconf.

Tabla 7.1. Opciones comunes para el programa configure de autoconf de GNU.

Opción	Significado
`--help`	Muestra la lista de los argumentos y opciones que soporta este paquete.
`--prefix=[ruta]`	Provoca que el software se instale en el directorio `[ruta]`.
`--host=[nombre]`	Especifica la arquitectura y plataforma del ordenador que construye el software; por ejemplo, `alphaev56-unknown-linux-gnu` o `i686-unknown-linux-gnu`; se usa para especificar que el software debería construirse para una arquitectura distinta de la estándar.
`--with-include-dir=[ruta]`, `--with-lib-dir=[ruta]`	Añade `[ruta]` a la lista de directorios que contienen archivos de cabecera o bibliotecas para el proceso de construcción.
`--enable-[opción]`	Activa la opción llamada `[opción]` en el software.
`--with-[paquete]=[ruta]`	Indica al proceso de construcción que busque el paquete llamado `[paquete]` en el directorio `[ruta]`; por ejemplo, `--with-openssl=/usr/local/openssl-0.9.5a`.

Nota: ¡Cuidado con prefix!

Normalmente, especificar un directorio de instalación a ./configure con la opción `--prefix`, provoca que todos los archivos se coloquen en subdirectorios del prefijo (*prefix*). Por ejemplo, `--prefix=/usr/local` causará que los archivos de configuración se ubiquen en /usr/local/etc, los binarios del programa en /usr/local/bin, y demás.
Sin embargo, algunos paquetes tratan al prefijo /usr como un caso especial. Estos paquetes colocarán sus binarios y bibliotecas en /usr/bin y /usr/lib, pero pondrán sus archivos de configuración en /etc (en lugar de en /usr/etc como podría esperarse). ¡Cuidado con estos programas! Para anular este comportamiento, probablemente pueda añadir una opción como `--config-prefix=/usr/etc`.

Las herramientas autoconf de GNU son extremadamente poderosas para escribir software *portable* (es decir, software adaptable, que pueda ejecutarse en muchos sistemas operativos) ya que puede particularizar un archivo Makefile para casi cualquier sistema similar a Unix. Como es tan potente, también se usa ampliamente. ¡Hoy en día, se encontraría en apuros si tuviera que encontrar un paquete de software de código abierto que no usara estas herramientas! Una comprensión práctica de las herramientas autoconf suponen un importante beneficio para cualquier usuario o administrador de un sistema basado en Linux.

Uso de las herramientas de Red Hat

El sistema de empaquetado utilizado por Red Hat Linux, RPM, fue descrito en el capítulo 4. Aquél capítulo mencionó brevemente el soporte que RPM ofrecía para paquetes de código fuente. En esta sección se comentará esa funcionalidad en detalle.

RPM y arquitecturas

Cada paquete RPM tiene asociada una "arquitectura". Normalmente, esta arquitectura indica la plataforma hardware para la que se compiló el paquete; por ejemplo, un paquete .i386.rpm es un RPM compilado para la plataforma Intel. Un RPM fuente tiene la extensión .src.rpm. Mientras que un RPM binario obviamente es sólo útil para las plataformas para las que se compiló, un RPM fuente puede convertirse en un RPM binario y por eso es útil para una gran variedad de tareas.

Construcción de RPM fuente

Típicamente, un RPM fuente se construye para corresponderse con la configuración de un RPM binario. Por ejemplo, junto con los CD de instalación, Red Hat también distribuye varios CD con los RPM fuente para cada paquete incluido con el

sistema. Cuando se construyen, estos RPM fuente producirán archivos RPM como los que se incluyen en los CD de instalación. Es decir, reconstruir todo los RPM fuente duplicará los RPM contenidos en los CD. Esto resulta útil, por ejemplo, si se construye una copia de Red Hat Linux en una plataforma para la que Red Hat no proporciona una distribución (aunque en la práctica rara vez es tan simple).

Cada archivo RPM fuente se construye con una configuración específica. Es decir, los contenidos del RPM fuente se preparan con un conjunto predeterminado de opciones en tiempo de compilación. El modo en que se especifican estas opciones varía según el software que se empaquete; el método más común, como se mencionó con anterioridad en este capítulo, es mediante la herramienta autoconf de GNU. Estas opciones en tiempo de compilación se especifican en archivos de especificación RPM con la extensión .spec. Los archivos .spec se pueden particularizar para modificar la configuración de un RPM.

Comandos RPM

El comando RPM puede usarse para construir archivos RPM binarios a partir de su fuente, y hay generalmente dos maneras para conseguir esto. El método más rápido y simple es reconstruir un RPM fuente en forma de paquete binario con su configuración por defecto. Esto puede lograrse mediante el comando `rpm --rebuild <nombre_de_archivo>`.

El comando `rpm --rebuild` recompila un paquete de código fuente con su configuración por defecto. En los casos en que los parámetros de un RPM fuente necesiten particularizarse, se puede usar un proceso algo más largo. Cada RPM fuente suele contener un archivo *tarball* con el software, el archivo de especificación para el RPM, y a veces archivos de apoyo (como parches que hay que aplicar al software). Instalar un RPM fuente (mediante el comando `rpm --install` descrito en el capítulo 4) provoca que estos archivos se extraigan e instalen y después puedan ser particularizados.

Contenidos del directorio /usr/src/redhat

Instalar un archivo RPM fuente en un sistema Red Hat hace que sus contenidos se coloquen en el directorio /usr/src/redhat. Este directorio contiene diversos subdirectorios; el uso de esos directorios se describe en la tabla 7.2.

Una vez que se ha instalado un RPM fuente, puede particularizarse mediante la edición de su archivo .spec, que estará en /usr/src/redhat/SPECS. Tras la edición del archivo .spec del RPM, se puede utilizar el comando `rpm -ba <nombre_de_archivo>` para indicar a RPM que construya el paquete, incluidas todas sus fases. (Hay diversas fases en la construcción de un RPM, desde la configuración a la instalación, y cada fase pude especificarse independientemente, aunque no suele ser necesario.) El archivo .spec se analizará y sus contenidos (que se parecen mucho a un guión) se ejecutarán para compilar el paquete. Durante este proceso, tratará de ubicar el archivo *tarball* del código fuente en /usr/src/redhat/SOURCES, y cuando se complete el proceso, colocará el nuevo RPM binario en /usr/src/redhat/RPMS.

Tabla 7.2. Contenidos del directorio /usr/src/redhat en Red Hat Linux.

Directorio	Contenidos
BUILD	Directorios separados y temporales de trabajo para los paquetes que se construyen a partir del código fuente.
RPMS	Paquetes RPM binarios completos (por ejemplo, compilados) procedentes de RPM fuente que se construyeron con éxito.
SOURCES	Archivos *tarball* de código fuente y parches utilizados para construir el software extraído de los RPM fuente que se instalan.
SPECS	Archivos de especificación para construir paquetes RPM binarios a partir del código fuente instalado en SOURCES.
SRPMS	Copias de archivos RPM fuente (*source RPMs*, SRPMS) que se han instalado, como referencia para más adelante.

En la mayoría de los casos, los administradores del sistema y los usuarios necesitan modificar unas pocas opciones de configuración, como la inhabilitación de una característica en tiempo de compilación o la modificación de una ruta. Estas necesidades se pueden cubrir mediante las técnicas precedentes. Aún así, para realizar tareas más complicadas, los usuarios deberían consultar la extensa documentación sobre RPM de Red Hat en www.redhat.com. (De manera alternativa, el libro de referencia "Maximum RPM" está incluido en formato digital en el CD de documentación de las distribuciones de Red Hat Linux más recientes.)

Uso de las herramientas de Slackware

La distribución de Slackware Linux se comentó en el capítulo 5. El mecanismo de empaquetado de Slackware se basa en simples archivos *tarball*, que son archivos tar (*Tape Archive*) estándar de Unix comprimidos. Típicamente, estos archivos tienen las extensiones .tar.gz o .tar.Z. Ya que utilizar un formato tan simple, Slackware tiene una limitada capacidad de añadir soporte extenso para paquetes de código fuente.

El método típico para instalar un paquete a partir de su código fuente en un sistema Slackware Linux es por ello muy sencillo. El administrador simplemente construye el paquete manualmente, según el mecanismo que resulte necesario, y después construye un paquete de Slackware a partir de él y lo instala mediante las herramientas estándar. Típicamente, esto implica la ejecución del guión ./configure que se suministra con la mayoría de los paquetes.

Por ejemplo, podría instalar el hipotético paquete mi-software en el directorio /opt/mi-software-1.0 de acuerdo con estas instrucciones:

1. Descargue el archivo *tarball* con el código fuente, y consulte la documentación sobre las opciones en tiempo de compilación.

2. Extraiga el código fuente del software dentro de un directorio apropiado, como /usr/src/mi-software. Escoja un directorio temporal para "instalar" el software en él, como /tmp/mi-software-tmp-/opt/mi-software-1.0. Esta no será la ubicación permanente, si no que se pasará al siguiente comando.
3. Ejecute el programa ./configure con los argumentos apropiados, en particular la opción `--prefix` (por ejemplo, `./configure --prefix=/tmp/mi-software-tmp/opt/mi-software-1.0`).
4. Construya el software mediante el comando `make`; cuando se haya completado la construcción, "instálelo" en el directorio temporal mediante el comando `make install`.
5. Entre en el directorio temporal (es decir, ejecute `cd /tmp/mi-software-1.0`).
6. Cree un paquete para Slackware a partir del directorio temporal, mediante un comando como `makepkg /tmp/mi-software-1.0.tgz`. Este comando creará un paquete para Slackware (que es simplemente un archivo *tarball* estándar con alguna información extra).
7. Instale el paquete recién creado mediante el comando `installpkg /tmp/mi-software-1.0`.

Algunos lectores pueden preguntarse por qué el software se construyó e instaló en un directorio temporal como /tmp/mi-software-1.0 en los pasos 3 y 4. La otra alternativa habría sido instalar directamente el software en su ubicación final (como /opt, /usr/local o el directorio raíz). Sin embargo, si se usa este enfoque, el software se instalará en su ubicación permanente, sin el conocimiento de las herramientas de empaquetado. Esto significa que las herramientas no serán capaces de desinstalar, actualizar, o administrar de ningún modo el paquete. En su lugar, el software se instala en un directorio de trabajo temporal, para que la herramienta makepkg pueda ejecutarse para localizar y consolidar todos lo archivos. Cuando el paquete se instale, las herramientas sabrán acerca de él y por ello serán capaces de administrarlo. (En realidad, siempre podría editar el contenido de /var/log/packages directamente, ¡pero eso es un montón de trabajo y va contra el propósito de las herramientas en primer lugar!)

Lo último que se debe hacer notar con respecto al proceso de instalación es el directorio temporal escogido. En este ejemplo fue /tmp/mi-software-tmp/opt/mi-software-1.0. El motivo para esto es asegurarse de que el software se instala en su directorio temporal de tal modo que la herramienta makepkg puede crear adecuadamente el paquete. La herramienta incluirá todos los archivos y subdirectorios contenidos en el directorio actual dentro del paquete resultante; por eso, al instalar el software en /tmp/mi-software-tmp/opt/mi-software-1.0 y ejecutar el comando desde /tmp/mi-software-tmp, el paquete se creará de modo que todos los archivos sean almacenados en opt/mi-software-1.0. Ya que installpkg instalará los archivos con respecto al directorio raíz, esto provocará que los contenidos del paquete se

coloquen en opt/mi-software-1.0 (que es el objetivo original). De manera similar, si en cambio el objetivo fuera instalar el paquete en /usr/local, el directorio de trabajo tendría que haber sido /tmp/mi-software-tmp/usr/local.

Las herramientas de Slackware son muy simples, de acuerdo con la filosofía propia de Slackware. Esto hace que sean fáciles de usar pero aún deja a los administradores un poco más de trabajo por hacer. (Compare este ejemplo para Slackware con el mucho más corto comando `rpm --rebuild` que se usa en los sistemas Red Hat Linux.) Por otro lado, incluso aunque las herramientas de Slackware puedan necesitar un poco más de trabajo a la hora de construir software a partir del código fuente, son mucho más fáciles de comprender y no requieren tanto estudio como otras herramientas más sofisticadas.

Uso de las herramientas de Debian

Construir paquetes Debian se consigue mediante la utilización del programa dpkg-deb incluido con Debian GNU/Linux. Esta herramienta tiene similitudes con los enfoques de ambos sistemas RPM y Slackware. Específicamente, el sistema de Debian tiene un formato propio para los archivos, pero generalmente construye ese tipo de paquetes mediante el simple proceso y almacenamiento de los contenidos de un directorio en particular.

El formato propio convierte al de Debian un sistema similar a RPM, en el sentido de que puede realizar operaciones como instalar, desinstalar y preguntar sobre los paquetes, y en que el propio sistema sigue la pista de las dependencias en lugar del usuario. Sin embargo, crear este tipo de paquetes implica sólo archivar los contenidos del directorio actual (preparado de manera especial), que resulta similar al enfoque de Slackware.

Nota

Crear paquetes Debian también está comentado en la sección "Manipulación de archivos de paquetes Debian" en el capítulo 6.

Por ejemplo, suponga que quiere crear un paquete Debian para el paquete hipotético "mi-software" considerado en la sección anterior sobre las herramientas de Slackware. A un nivel muy alto, el paso para construir un paquete como este para Debian es muy similar a los siete pasos delineados en la sección de Slackware. Las diferencias principales son que en lugar de utilizar el comando `makepkg` en el paso 6, usará el comando `dpkg-deb --build /tmp/mi-software-1.0.deb`. De manera similar, una vez que haya creado el paquete, usaría el comando `dpkg -i /tmp/mi-software-1.0.deb` para instalarlo, en lugar del comando `installpkg` del paso 7.

Como puede ver, crear paquetes Debian a partir del código fuente es conceptualmente muy similar al proceso equivalente en Slackware Linux. Por otra

parte, una vez que ha conseguido el archivo, puede realizar diversas preguntas sobre él (como obtener la lista de sus contenidos o ver su descripción) al igual que lo haría con un RPM. Sin embargo, esta historia no está completa del todo y hay otro modo en que el sistema de Debian es similar al de Red Hat.

Específicamente, el proceso que se acaba de esquematizar omite un paso muy importante, la creación de los archivos y directorios necesarios para los archivos Debian.

Para que se pueda ejecutar correctamente, el comando `dpkg-deb` espera encontrar un directorio llamado "DEBIAN" como subdirectorio del árbol que desea convertir en un paquete Debian. Este directorio contiene diversos archivos adicionales, que se resumen en la tabla 7.3.

Tabla 7.3. Archivos de un paquete Debian.

Archivo	Propósito
control	Contiene información básica sobre el software que se va a empaquetar, como su versión, descripción, y dependencias.
rules	Un guión utilizado para preparar los contenidos del directorio para su empaquetado.
changelog	Lista de la historia de cambios realizada en el paquete.
copyright	Identifica al propietario del copyright y otros asuntos relacionados.

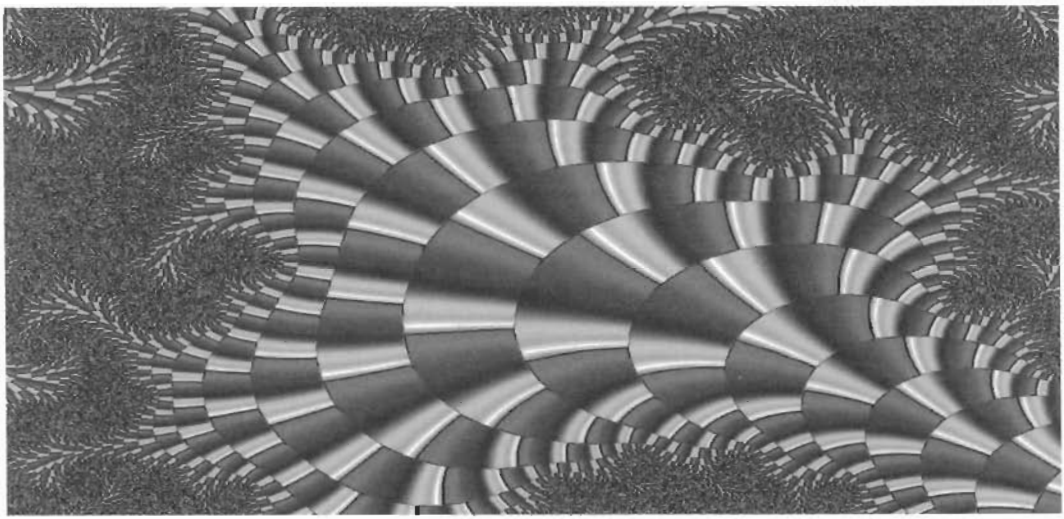
Probablemente ya se haya dado cuenta de que cuando se consideran como un conjunto, los archivos comentados en la tabla 7.3 equivalen a los archivos .spec de Red Hat para los paquetes RPM.

Estos archivos los utiliza el comando `dpkg-deb` para preparar el directorio y construir el paquete Debian a partir de sus contenidos. Por ello, mientras que el proceso presenta mecánicamente similitudes con Slackware (en el sentido de que construye un archivo a partir de los contenidos del directorio actual), también tiene grandes similitudes con Red Hat. Claramente, hay más de un modo de despellejar al mismo gato. (La mayoría de los sabores comerciales de Unix normalmente tienen cada uno un modo particular de enfrentarse a este problema.)

A no ser que sea un desarrollador de software, puede que no necesite crear paquetes Debian en absoluto. Como usuario, todo lo que probablemente necesitará saber sobre este proceso es que no es tan sencillo reconstruir un paquete Debian a partir de su código fuente como lo es en el caso de un RPM, ya que no hay preparado ningún paquete "source.deb" similar a un "RPM fuente". Dado que este no es un libro sobre el desarrollo de software o sobre cómo construir paquetes Debian, si necesita información más detallada, debería consultar la documentación del Proyecto Debian, que se puede encontrar en www.debian.org/doc/manuals/programmer/index.html.

Resumen

Un sistema se personaliza mediante el software y las configuraciones de software instaladas en él. Este capítulo dibujó algunas de las decisiones básicas que se deben tomar y los pasos a seguir para instalar un nuevo paquete de software en un sistema Linux. El capítulo 8 describe en detalle los procedimientos de instalación para varios paquetes de software populares. Ahora que ha leído este capítulo, dispone de las herramientas básicas necesarias para realizar casi cualquier instalación de software, desde la fase de toma de decisiones hasta la instalación.



8
La shell segura OpenSSH

El capítulo 7 cubrió los conceptos básicos de la instalación de software. También ofreció algunas reglas básicas y técnicas generales que son útiles para la instalación de muchos tipos de software. Este capítulo es el primero de seis que entrarán con detalle en la instalación de varias aplicaciones comunes reales. Estos capítulos se pueden utilizar como referencia. Tras su lectura prácticamente podrá instalar en cualquier sistema cualquier software que encuentre en su camino.

OpenSSH es un subproyecto del proyecto OpenBSD. El proyecto OpenBSD es uno de los descendientes de código abierto y libres del sistema BSD original similar a Unix. (Lea el capítulo 1 para ver comentado el árbol genealógico de los sistemas Unix.) OpenBSD se centra en la seguridad, y hace un uso extenso de la criptografía fuerte, la inspección rigurosa del código, y otras medidas de seguridad. OpenSSH es la implementación del protocolo estándar de shell segura (*Secure Shell*, SSH) de la *Internet Engineering Task Force* (IETF), que se desarrolló por parte del grupo de trabajo secsh de la IETF. OpenSSH se ha adaptado a una gran variedad de plataformas, incluidos los sistemas basados en Linux.

OpenSSH se compone tanto de un servidor como de un conjunto de clientes. El programa servidor, llamado sshd, proporciona acceso remoto seguro y cifrado (protegido mediante criptografía) a un servidor o estación de trabajo.

Los clientes acceden remotamente a los servidores que ejecutan sshd (u otro programa compatible) y proporcionan prestaciones que van desde la simple entrada remota al sistema (mediante el programa cliente ssh) a transferencias seguras de archivos, mediante los programas scp ("*secure copy*", copia segura) y sftp ("*secure FTP*", FTP seguro). Ya que SSH es un protocolo estándar, los clientes y el servidor

OpenSSH pueden interactuar con otro software que implemente el mismo protocolo SSH. El paquete OpenSSH que se instala en los sistemas Linux no es realmente la misma versión que se incluye con los sistemas OpenBSD. OpenSSH tiene dos sabores: la versión principal de OpenBSD y una ligera variante suya, que es la versión "adaptable". Esta versión adaptable se mantiene en paralelo a la versión principal de OpenBSD y es la que los usuarios descargan e instalan realmente en los sistemas que no son BSD.

Instalar y configurar OpenSSH es bastante sencillo. OpenSSH se basa en la biblioteca genérica de criptografía OpenSSL para algunas de sus funcionalidades. (OpenSSL se proporciona con la mayoría de las distribuciones, así que ya está presente en la mayor parte de los sistemas recientes. Si no se encuentra en su sistema, puede instalarla de un modo similar al que se describirá más adelante en el capítulo correspondiente a la biblioteca Dante.) Una vez instalado, busca los archivos de configuración (tanto para los programas clientes como para el servidor) en un directorio específico, que normalmente es /etc/ssh. Los archivos de configuración que se encuentran en ese directorio controlan el comportamiento en tiempo de ejecución de OpenSSH.

El propio OpenSSH se proporciona en la mayor parte de las distribuciones. Por eso, utilizarlo como ejemplo podría parecer algo redundante. Sin embargo, como este libro pretende ser útil más allá de los meros sistemas Linux, aún así se ha seleccionado OpenSSH como uno de los ejemplos. El material de esta sección puede aplicarse para instalar OpenSSH no sólo en los sistemas Linux que aún no lo tengan (como en el caso de distribuciones antiguas), sino también en otros sistemas similares a Unix, como los sistemas comerciales. Una implementación de SSH es algo que hay que tener, y OpenSSH es una de las mejores que se pueden encontrar. Además, su utilidad como un ejemplo clásico de una aplicación configurada desde /etc contribuyó a la inclusión de OpenSSH en esta tercera parte.

Nota: Sólo los hechos: OpenSSH

Propósito: Implementación libre del protocolo SSH para el acceso remoto.
Autores: El Proyecto OpenBSD
Sitio web: www.openssh.org
Descripción: OpenSSH es una implementación libre (de código abierto) del protocolo estándar SSH de la IETF. Este protocolo proporciona acceso seguro y con cifrado fuerte a sistemas remotos mediante una interfaz parecida a telnet.

Instalación de OpenSSH

Las siguientes secciones describen el proceso de instalación de OpenSSH. En ellas se comentará no sólo la mecánica de la instalación, sino también los detalles y

las decisiones que hay que tomar (por ejemplo, cómo escoger un directorio de instalación). Tras la lectura de este material, será capaz de instalar OpenSSH en cualquier sistema.

Opciones en tiempo de compilación

Como todos los ejemplos en este libro, compilará OpenSSH a partir de su código fuente. El primer paso en el proceso de la construcción de software es, como se comentó en el capítulo 7, decidir las opciones en tiempo de compilación. Esta sección comentará las opciones en tiempo de compilación que soporta OpenSSH y ofrecerá una configuración detallada real que resulte apropiada para un sistema de uso real.

Selección de una ruta de instalación

La primera opción que es relevante para casi cualquier paquete de software es el directorio de instalación, es decir, el directorio en el que se va a instalar el programa compilado. Los capítulos 3 y 7 comentaron algunas de las opciones típicas para los directorios de instalación, como los directorios /opt y /usr/local. El capítulo 7 ofrece detalles completos, pero en general el directorio /opt es apropiado para software que podría necesitar ser actualizado de forma regular o que necesite que se instale más de una versión, mientras que /usr/local es más apropiado para software que rara vez se actualizará. Como OpenSSH es un programa que se actualizará con poca frecuencia, en este ejemplo se instalará en /usr/local.

Selección de las opciones de configuración

OpenSSH soporta una amplia variedad de opciones de configuración. En general, estas opciones de configuración están relacionadas con la habilitación de diversos mecanismos de autenticación (como el soporte a Kerberos o a "tarjetas inteligentes") y unas pocas cosas más, como si se soporta el nuevo estándar de protocolo de Internet IPv6. Estas opciones, mientras tanto, generalmente se reducen a si el sistema operativo las soporta o no; por ejemplo, si el sistema operativo no tiene ni utiliza las bibliotecas Kerberos, entonces no tiene sentido que para OpenSSH se habilite el soporte de Kerberos (y de hecho el proceso fallará si lo intenta). Como el programa ./configure detectará automáticamente si cada una de estas opciones está disponible en el sistema y habilitará o inhabilitará la opción de un modo apropiado, no hay necesidad de que se preocupe por ellas.

Sin embargo, esa no es una regla general: muchos programas vienen con opciones que habilitan o deshabilitan características del programa. En ese tipo de casos, no hay modo de que el programa realice ninguna autodetección; después de todo, ¿cómo detecta automáticamente un programa si quiere que una determinada característica esté inhabilitada? En esos casos, las características pueden habilitarse e inhabilitarse explícitamente. (Vea el capítulo acerca del servidor web Apache en esta tercera parte para tener un ejemplo de este tipo de instalación.)

Además, incluso aunque la mayoría de las características de OpenSSH se puedan autodetectar, bajo ciertas circunstancias podría no desear realmente este comportamiento. Por ejemplo, puede que no quiera que OpenSSH emplee la autenticación Kerberos aunque esté instalado en el sistema. En esos casos, puede consultar la documentación y el comando `./configure --help` para descubrir cómo inhabilitar las características innecesarias.

En pocas palabras, su instalación de OpenSSH probablemente no necesitará opciones particulares, excepto para fijar la ruta de instalación. Utilice el siguiente comando para configurarla. (Este comando será distinto si escoge habilitar o inhabilitar explícitamente una o más de las características detectadas automáticamente.)

```
$ ./configure --prefix=/usr/local
```

Fíjese en que la opción `--prefix` indica que /usr/local debe ser el directorio de instalación. Esto significa que los archivos binarios del programa serán instalados en /usr/local/bin, los archivos de configuración en /usr/local/etc, y así. Si el comando se completa sin ningún error, el software debería de estar preparado para el proceso de compilación.

Instalación a partir del código fuente

Una vez que se ha ejecutado el programa ./configure tal y como se comentó en la sección anterior, se puede construir realmente el software. Es muy simple; consiste en ejecutar el comando `make` seguido del comando `make install`, como se muestra aquí:

```
$ make
```

(Aparecerá información sobre el proceso de construcción.)

```
$ make install
```

(Aparecerá información sobre el proceso de instalación.)

Además de copiar archivos, el comando `make install` también ejecutará algunos guiones que fijan claves criptográficas del servidor y otros detalles básicos. Si el comando `make install` (y el comando `make` antes de él) se completan con éxito, entonces ¡se ha instalado OpenSSH! Si alguno de los dos falla, probablemente signifique que algo ha ido mal durante el proceso de configuración; compruebe que el comando `./configure` se ha completado con éxito y que las diversas opciones que pueda haberle pasado son correctas. (Por ejemplo, tenga cuidado con posibles errores tipográficos en los nombres de las rutas.)

Configuración de OpenSSH

Con la ruta de instalación /usr/local, los archivos de configuración de OpenSSH deberían encontrarse en /usr/local/etc. (Esto puede modificarse con la opción

`--sysconfdir=` que se utiliza para indicar una ruta de configuración distinta, como /etc en lugar de /usr/local/etc. Sin embargo, generalmente es una buena idea dejar los archivos de configuración en la ubicación por defecto, a menos que haya una razón para no hacerlo así.)

Habrá una serie de archivos relacionados con OpenSSH en /usr/local/etc. La mayoría de ellos contienen las claves públicas y privadas para el servidor, tanto para el sistema RSA como para el DSA. (RSA y DSA con algoritmos criptográficos de clave pública. Si desea más información sobre estos sistemas, consulte un libro sobre criptografía.) Los archivos de clave pública y privada son ssh_host_dsa_key, ssh_host_rsa_key, ssh_host_dsa_key.pub, ssh_host_rsa_key.pub, ssh_host_key y ssh_host_key.pub. Cada uno de estos archivos está en el directorio /usr/local/etc. Estos archivos se crean automáticamente cuando se instala el software, así que no debería tener que hacer nada con ellos, a no ser lo que lo necesite (como por ejemplo si va a copiar archivos de claves desde otro servidor). Otro archivo relacionado es /usr/local/etc/moduli, que se utiliza como parte del intercambio de claves criptográficas. Como los archivos de claves, no debería necesitar hacer nada con este archivo.

Los archivos restantes son /usr/local/etc/ssh_config y /usr/local/etc/sshd_config. Estos son los auténticos archivos de configuración para OpenSSH. Configuran el programa cliente ssh y el programa servidor sshd respectivamente.

Nota: Actualización de OpenSSH

En algún momento aparecerá una versión actualizada del paquete OpenSSH, y puede desear actualizar la versión que tiene instalada. En este caso, necesitará actualizar los archivos binarios sin modificar los archivos de configuración. Por ejemplo, no querrá sobrescribir los archivos de claves privadas, o la identidad SSH de su sistema cambiará, lo que puede resultar muy inconveniente para los usuarios. Por eso, necesita tener cuidado cuando actualice OpenSSH para evitar perder los archivos de configuración.

La mayoría de las distribuciones incluyen OpenSSH junto con la instalación básica y proporcionan actualizaciones periódicas cuando son necesarias. (Por ejemplo, Red Hat Linux incluye un paquete RPM para OpenSSH, y Red Hat hace públicas de manera periódica versiones actualizadas.) En estos casos, el empaquetador nativo del sistema se encargará de actualizar los archivos binarios sin alterar los de configuración.

Sin embargo, si sigue el ejemplo de este libro e instala a partir del código fuente, entonces está confiando en los guiones de instalación del paquete del código fuente para que copien los nuevos archivos binarios. Estos guiones también instalan copias por defecto de los archivos de configuración. Si el guión no es lo suficiente "avispado" como para ver las copias que ya están instaladas, puede sobrescribirlas. Por ello, siempre es una buena política hacer una copia de seguridad de sus archivos de configuración antes de

instalar una nueva pieza de software. En el caso de OpenSSH, los guiones son de hecho tan inteligentes como para evitar sobrescribir las copias existentes de los archivos de configuración, pero un buen administrador siempre estará alerta.

Configuración del cliente ssh

El archivo de configuración para el programa cliente ssh es /usr/local/etc/ssh_config. (Este archivo también lo utilizan otros programas clientes, como el programa scp o sftp.) El archivo contiene una serie de líneas, cada una de las cuales fija una opción diferente que controla el comportamiento en tiempo de ejecución del cliente OpenSSH.

El formato del archivo es muy sencillo. En primer lugar, se especifica una máquina mediante la línea "Host", y después aparece la lista de las opciones para esa máquina. Si se necesitan máquinas adicionales, se puede añadir otra línea Host para cada máquina seguida por las opciones para esa máquina en particular. Una línea de máquina que consista en "Host*" valdrá para cualquier máquina y por eso será efectivamente la máquina por defecto.

Tras la instalación, /usr/local/etc/ssh_config contendrá una configuración por defecto típica. Esta será un conjunto de parámetros bastante genéricos optimizados para maximizar la seguridad; es decir, sólo deja activadas opciones básicas e inhabilita las características "más arriesgadas". Por ejemplo, por defecto /usr/local/etc/ssh_config inhabilita el reenvío por agente SSH, ya que puede suponer una vulnerabilidad de la seguridad si no se utiliza con cuidado.

OpenSSH soporta un gran número de opciones, la mayoría relacionadas con los matices de los sistemas y algoritmos criptográficos que puede utilizar. Cubrir todas estas opciones y sus matices es un trabajo que es mejor dejar a la documentación sobre OpenSSH, como su propia página de manual (que es bastante detallada). Sin embargo, merece la pena comentar dos ejemplos comunes sobre cómo cambiar una opción desde su valor por defecto.

Habilitar el reenvío por agente SSH

Recuerde que OpenSSH inhabilita por defecto el reenvío por agente SSH. Un agente SSH es un programa especial (normalmente llamado ssh-agent) que se ejecuta en segundo plano y administra las claves privadas para un usuario. El agente SSH recibe el nombre de un programa que debe ejecutar, y después gestiona las claves para ese programa y todos los hijos que ese programa genere. El modo más común de utilizar este mecanismo es ejecutar el programa ssh-agent para que administre una shell de entrada al sistema, como se haría mediante el comando `ssh-agent bash`. Una vez que el programa especial ssh-add se ha utilizado para añadir una clave que sea gestionada por ssh-agent, todos los programas ejecutados desde esa shell podrán hacer uso de esa clave automáticamente.

En esencia, esto significa que el usuario no tiene que introducir la contraseña secreta que permite el uso de la clave cada vez que se utiliza; el programa ssh-agent se encargará de ello. Un ejemplo de uso de ssh-agent viene a continuación. Si el usuario tiene sus llaves configuradas correctamente en los servidores local y remoto (como se describe en la documentación de SSH), entonces los siguientes comandos prepararán y utilizarán un agente SSH:

```
$ exec ssh-agent bash
$ ssh-add
```

(ssh-add pedirá al usuario su contraseña.)

```
$ ssh mi.maquina.remota
```

(ssh cogerá la clave del ssh-agent y entrará en el sistema automáticamente.)

Tras entrar en la máquina `mi.maquina.remota` (lo que sucederá automáticamente en el ejemplo precedente), el usuario puede desear entrar en un segundo sistema remoto, como si se tratara de un repetidor o un punto de salto. Si el reenvío por agente SSH está habilitado (y el servidor remoto también lo permite), entonces el agente que se configuró inicialmente en la máquina local puede volver a usarse para entrar en la segunda máquina remota. Si el reenvío por agente está inhabilitado, entonces el usuario tendrá que introducir una contraseña.

Si esto parece algo confuso, ahora comprenderá una razón por la que el reenvío por agente está inhabilitado por defecto: es demasiado fácil equivocarse. Otro asunto potencialmente importante es que habilitar el reenvío por agente SSH expone su cuenta a debilidades adicionales conocidas como ataques del "intermediario" (*man-in-the-middle*), en los que un servidor de confianza intermedio (un "intermediario") ve su seguridad comprometida. La consecuencia de estas debilidades podría ser que un usuario pusiera accidentalmente su cuenta en un estado en que un atacante pudiera obtener acceso a un sistema remoto sin ni siquiera necesitar la clave del usuario o la contraseña SSH. Antes de utilizar agentes SSH, debería leer cuidadosamente y comprender la documentación de OpenSSH y configurar su sistema y su cuenta adecuadamente.

Si desea utilizar el reenvío por agente SSH, habrá que habilitarlo en su sistema local. Para activarlo, edite la siguiente línea en /usr/local/etc/ssh_config para que el valor sea "*yes*" en lugar del valor por defecto de "*no*". El cambio surtirá efecto la siguiente vez que un programa cliente se ejecute.

```
ForwardAgent no
```

Realmente, todo esto es un rodeo para describir cómo realizar un cambio en el archivo ssh_config. Cada opción tiene un valor, y configurar el programa es tan simple como modificar los valores de las opciones. Sin embargo, OpenSSH es sólo tan seguro como su configuración, y es muy sencillo configurar OpenSSH de un modo inseguro.

Un administrador sabio y competente leerá la documentación de OpenSSH con detenimiento antes de realizar ningún cambio.

Inhabilitar la comprobación estricta de claves de máquina

Otra opción que está habilitada en el archivo /usr/local/etc/ssh_config por defecto es la comprobación estricta de claves de máquina. OpenSSH implementa los sistemas de criptografía de clave pública RSA y DSA, de manera que tanto el cliente como el servidor tienen claves secretas que permiten que se identifiquen de forma no ambigua. Sin embargo, si dos ordenadores nunca se "han encontrado" antes (por ejemplo, si no han intercambiado aún sus claves), entonces no hay ningún modo de que puedan verificar sus respectivas identidades.

Los clientes de OpenSSH tienen dos opciones para hacerse cargo de este caso. Pueden cerrar las conexiones de máquinas con claves desconocidas o simplemente añadir la clave de la máquina a su lista de claves conocidas y advertir al usuario de esta circunstancia.

La primera opción se conoce como comprobación estricta de claves de máquina y habilitarse o inhabilitarse gracias al archivo /usr/local/etc/ssh_config. Utilizar la comprobación estricta de claves de máquina es más seguro, ya que ayuda a prevenir ataques de "intermediario" similares al mencionado con anterioridad; no obstante, es menos conveniente, ya que obliga a que el usuario importe manualmente las claves de los servidores en los que confía (o que anule la opción manualmente en la línea de comandos). Como siempre, supone un compromiso entre la seguridad y la comodidad, y debe tener cuidado al valorar sus opciones. Si desea más información sobre la comprobación estricta de claves de máquina, consulte la documentación en www.openssh.org.

Si decide inhabilitar la comprobación estricta de claves de máquina, el procedimiento es el mismo que para habilitar el reenvío por agente SSH (comentado en la sección anterior). Es decir, localice la línea siguiente en /usr/local/etc/ssh_config y cambie el valor por defecto de "*yes*" que la habilita a "*no*".

```
StrictHostKeyChecking yes
```

Una vez que ha realizado este cambio, los clientes OpenSSH comenzarán automáticamente a añadir las claves de los servidores remotos a la lista de claves conocidas del usuario. Se generará una advertencia, pero será responsabilidad del usuario el tener cuidado con las máquinas remotas de poca confianza. (Fíjese, sin embargo, que los usuarios pueden anular este comportamiento en sus propias configuraciones. En general, ssh_config modifica sólo las opciones por defecto para los clientes.)

Si inhabilitar la comprobación estricta de forma tan rotunda no le resulta agradable, hay también una tercera opción, denominada "*ask*" (preguntar), que puede utilizarse como compromiso. Cuando los clientes ssh se encuentran en este modo, ni añadirán automáticamente las claves desconocidas ni las rechazarán. En su lugar, cada vez que se encuentren con una clave desconocida preguntarán al usuario si debería aceptarse. Esto necesita interacción con el usuario, pero resulta menos inconveniente que dejar activada la comprobación estricta de claves de máquina. Para habilitar esta opción, simplemente utilice "*ask*" en lugar de "*no*" o "*yes*" en la línea anterior.

Nota: Configuraciones en paquetes binarios

Los ejemplos encontrados en esta sección acerca de cambiar las configuraciones por defecto de los clientes y el servidor OpenSSH muestran cómo realizar este tipo de cambios en general. Los procedimientos seguidos funcionarán en cualquier instalación de OpenSSH, ya provenga de una instalación a partir del código fuente o de una instalación binaria como un archivo RPM o un paquete de Slackware. Sin embargo, es importante recordar que estas instalaciones binarias pre-empaquetadas están configuradas de un modo específico, que puede no ser el mismo que en el caso de la instalación por defecto a partir del puro código fuente de OpenSSH. (Por ejemplo, la instalación de OpenSSH, que Red Hat incluye con su distribución, ya tiene inhabilitada la comprobación estricta de claves de máquina, mientras que en la instalación inicial estaba habilitada.) Por eso, compruebe siempre que los cambios que va a hacer no se han realizado aún, y no se sienta confundido si descubre que ha sido así. "¡Instalador, cuidado!", podría decirse.

Configuración del servidor sshd

Configurar el servidor sshd es muy parecido a configurar los programas cliente. El archivo /usr/local/etc/sshd_config contiene múltiples líneas, donde cada una de las líneas fija el valor de una opción en particular. Al igual que con ssh_config, sshd_config contiene un conjunto bastante razonable de valores por defecto. Modificar el valor de una opción SSH también es tan simple como editar la línea apropiada en el archivo.

Una vez más, OpenSSH puede volverse muy inseguro debido a unas opciones de configuración mal escogidas, así que es vital leer y comprender la documentación de OpenSSH (como la página man para sshd) antes de realizar ningún cambio en la configuración. Sin embargo, un cambio bastante popular y habitual es habilitar el reenvío de conexión con X Window.

Habilitar el reenvío de conexión con X Window

Frecuentemente, un usuario entrará en una máquina remota desde un xterm u otro shell que esté ejecutándose dentro de un entorno X Window. Normalmente, los programas de X que el usuario ejecuta en la máquina remota están pensados para mostrarse en la máquina local físicamente presente frente al usuario. (Después de todo, el programa no sirve de mucho si el usuario no puede ver lo que hace.) El usuario tendrá que fijar una variable de entorno DISPLAY para que el programa pueda comportarse correctamente; sin embargo, OpenSSH soporta el reenvío de la conexión a X (*X Window Connection Forwarding*) para manejar esto automáticamente.

Es decir, el servidor sshd y el cliente ssh pueden establecer automáticamente un "túnel" a través del cuál los programas de X de la máquina remota pueden ejecutarse. Cuando un cliente entra en el sistema, el servidor sshd en el sistema remoto establece un servidor X Window virtual o fantasma al que se pueden conectar los programas que se ejecuten en ese sistema.

Este servidor simplemente retransmite a cualquier programa que se conecte a él de vuelta a lo largo de la conexión SSH hasta el propio sistema local del usuario. Además de la conveniencia de configurar automáticamente la variable DISPLAY, la conexión estará cifrada. Por eso, el reenvío de conexión con X Window es una característica muy útil y popular de OpenSSH. Sin embargo, bajo ciertas circunstancias puede ser un agujero de seguridad (ya que podría permitir el acceso no deseado al servidor X del usuario, o cosas incluso peores), así que la configuración por defecto de OpenSSH tiene inhabilitado el reenvío de conexión con X. Para habilitarlo, simplemente hay que encontrar la siguiente línea en el archivo /usr/local/etc/sshd_config y cambiar el valor por defecto de "*no*" a "*yes*":

```
X11Forwarding no
```

Fíjese en que esto habilitará el reenvío de conexión con X en su máquina local, lo que significa que permitirá a los usuarios que entren en su sistema a través de SSH utilizar el reenvío de conexión con X. Si es un usuario que desea utilizar el reenvío de conexión con X en un servidor remoto, hay que realizar una modificación similar en el servidor remoto.

Resumen de la configuración de OpenSSH

Una vez que está instalado, configurar OpenSSH es en general bastante fácil. No obstante, es importante comprender todas las consecuencias de un cambio en la configuración antes de realizarlo. Después de todo, OpenSSH es el responsable de dejar que los usuarios entren en un sistema, así que un error en la configuración podría proporcionar potencialmente a un atacante las llaves del sistema. Sin embargo, el acto real de la configuración es muy sencillo: sólo hay que editar la línea apropiada para la opción que se va a actualizar.

Instalación de OpenSSH en diferentes distribuciones

La sección previa comentó la configuración de los programas cliente y servidor de OpenSSH. Sin embargo, como OpenSSH tiene un componente servidor, hay que enganchar ese servidor al proceso de arranque del sistema operativo. Las siguientes secciones describen como realizar esto en cada una de las distribuciones de muestra.

Instalación de OpenSSH en Red Hat Linux

Como se comentó en la sección "Guiones de arranque del sistema" del capítulo 4, Red Hat Linux utiliza el modelo SysV de guiones de arranque del sistema. En este modelo, un directorio especial contiene muchos guiones, cada uno de los cuales gestiona un servidor o servicio específico. Cuando el sistema arranca, está configurado para ejecutar estos guiones. Esta sección describe cómo configurar el servidor OpenSSH para utilizarlo junto con la implementación de Red Hat del modelo SysV. Realmente, la distribución de código fuente de OpenSSH incluye un guión como el necesario, ya listo para ser usado; sin embargo, resulta útil ver cómo funciona uno de estos guiones, por eso vamos a presentar una versión menos sofisticada y más simple.

Para que sea compatible con el modelo SysV, el guión necesita soportar como mínimo tres argumentos: `stop`, `start` y `restart`. Entonces, instalar OpenSSH sobre Red Hat Linux implica escribir un guión de shell que entienda estos tres comandos. El comando `stop` detendrá el demonio sshd, el comando `start` lo ejecutará y el comando `restart` lo detendrá y volverá a ejecutarlo al mismo tiempo. El guión del listado 8.1 recoge un guión de inicio muy básico compatible con el sistema SysV.

Listado 8.1. Guión compatible con SysV para el demonio sshd.

```
#!/bin/sh
# chkconfig:  345 25 25
# description:Gestiona el servidor OpenSSH sshd
[ -x /usr/local/sbin/sshd ] || exit 0

RC = 0

start () {
    echo $"Arrancando sshd."
    /usr/local/sbin/sshd
    return 0
}

stop () {
    echo $"Deteniendo sshd."
    [ - e /var/run/sshd.pid ] && kill -TERM `cat /var/run/sshd.id`
    RC=$?
   return $RC
}

restart () {
    stop
    start
    RC=$?
    return $RC
}

# Veamos cómo hemos sido llamados.
```

```
case "$1" in
    start)
       start
       ;;
    stop)
       stop
       ;;
    restart)
       restart
       ;;
    *)
       echo $"Uso: $0 {start|stop|restart}"
       RETVAL=1
esac

exit $RETVAL
```

Son particularmente notables en el listado 8.1 las dos primeras líneas, que permiten que el guión pueda utilizarse junto con las herramientas de Red Hat chkconfig y service. Esto significa que el archivo puede simplemente colocarse (o crearse un enlace a él) en el directorio /etc/rc.d/init.d, y entonces se podrá utilizar chkconfig para gestionarlo. Los siguientes comandos muestran cómo instalar y utilizar el guión sshd del listado 8.1:

```
$ cp listado-8.1.sh /etc/rc.d/init.d/sshd
$ chkconfig --add sshd
$ service sshd start
```

El listado 8.1 es mínimo: no incluye ningún detalle, pero realiza el trabajo básico. Un buen programador de guiones verá inmediatamente modos adicionales en los que el guión podría ser más robusto o útil. Sin embargo, el objetivo del listado 8.1 es mostrar cómo construir un guión básico compatible con SysV, no enseñar a programar guiones avanzados, por eso es muy básico.

Recuerde que el propio OpenSSH incluye una versión mucho más robusta del listado 8.1 en el archivo /contrib/redhat/sshd.init del paquete del código fuente. Este guión es el que Red Hat utiliza en su versión empaquetada como RPM de OpenSSH, y no hay ninguna razón para no utilizar la versión oficial del guión de OpenSSH en lugar del listado 8.1. El objetivo del listado 8.1 es mostrar un guión básico, así que si alguien le ofrece un guión mejor, utilícelo, por favor.

Instalación de OpenSSH en Slackware Linux

Como se comentó en la sección "Guiones de arranque del sistema" del capítulo 5, Slackware Linux utiliza el modelo BSD de guiones de arranque del sistema. En este modelo, hay esencialmente un único guión que es el responsable de la mayoría de las tareas del arranque del sistema; sin embargo, la versión de Slackware de este sistema también incluye un soporte muy básico para guiones compatibles con el modelo SysV. Esta sección comenta cómo configurar el servidor OpenSSH sshd para que inicie durante el proceso de arranque de un sistema Slackware Linux.

Recuerde que hay dos modos de configurar el servidor de OpenSSH para que se ejecute en Slackware Linux. Una manera es modificar los guiones del sistema /etc/rc.d/rc.M o /etc/rc.d/rc.inet2 e incluir directamente código de guión similar al del listado 8.1. Así el servidor comenzaría a ejecutarse cuando el sistema arrancara.

Sin embargo, como se mencionó en la sección anterior, OpenSSH ya incluye un guión compatible con SysV para ejecutar y detener el servidor. Como Slackware también ofrece un soporte básico para SysV, podría tener sentido utilizar simplemente el archivo proporcionado por Red Hat Linux junto con el mecanismo /etc/rc.d/rc.sysvinit de Slackware. Lamentablemente, el archivo que se proporciona con OpenSSH necesita archivos específicos de la implementación de Red Hat Linux del modelo SysV, así que ese archivo no se puede utilizar con Slackware Linux sin realizar modificaciones. No obstante, el listado 8.1 es mucho más simple, y por eso podría utilizarse junto con Slackware sin ninguna modificación. Simplemente coloque el listado 8.1 en un archivo dentro del subdirectorio apropiado de /etc/rc.d y el proceso de arranque de Slackware se encargará automáticamente de él.

Nota

Véase el capítulo 5 para obtener más información sobre /etc/rc.d.

Cualquier opción (editar directamente /etc/rc.d/rc.M o /etc/rc.d/rc.inet2, o utilizar el enfoque de SysV) es una solución perfectamente útil. La que prefiera un administrador es realmente una cuestión de gusto; por ejemplo, un administrador puede preferir que todas las actividades de arranque se lleven a cabo en /etc/rc.d/rc.M para impedir que accidentalmente se pase por alto una, mientras que otro puede preferir reciclar los guiones de SysV que ya han sido escritos para ahorrarse trabajo. Hay más de un modo de hacer las cosas, como es típico de Slackware Linux.

Instalación de OpenSSH en Debian GNU/Linux

Instalar OpenSSH en Debian GNU/Linux es muy parecido a instalarlo en Red Hat Linux. Como Debian GNU/Linux utiliza el mismo modelo de guiones de inicio que Red Hat Linux (el modelo SysV) entonces el material comentado con anterioridad que se aplicaba a Red Hat Linux también se aplica a Debian GNU/Linux.

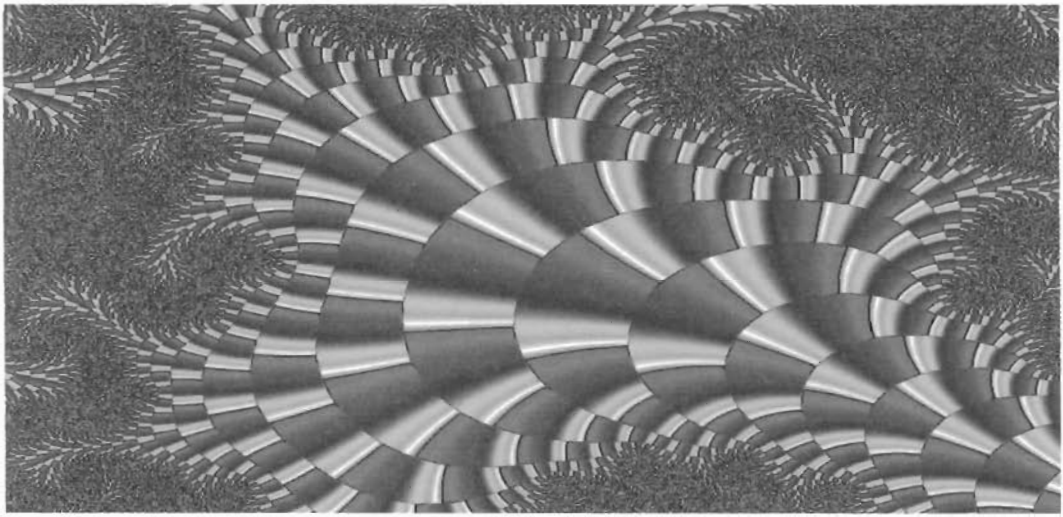
Al igual que Red Hat Linux, Debian GNU/Linux también incluye un paquete para OpenSSH con la distribución básica. Esto significa que probablemente pueda utilizar simplemente el paquete Debian (con sus propias configuraciones particulares, como se ha comentado a lo largo de este capítulo, por supuesto) tal cual, a no ser que necesite actualizarlo por sí mismo.

La única diferencia sustancial entre instalar OpenSSH en Debian GNU/Linux y Red Hat Linux es que Debian no incluye las herramientas service y chkconfig de Red Hat; en su lugar, tendrá que utilizar el programa update-rc.d de Debian, como se comentó en la sección "Trabajar con las herramientas de Debian" del capítulo 6.

Resumen

El programa OpenSSH del Proyecto OpenBSD es una implementación altamente compatible y robusta del protocolo SSH estándar del IETF. Instalar la forma adaptable de OpenSSH válida para los sistemas Linux es bastante sencillo, aunque muchas distribuciones ofrecen una forma pre-empaquetada de un modo más conveniente. Configurar OpenSSH es en general tan simple como editar el archivo apropiado, dependiendo de si va a tratar de configurar el programa servidor o los clientes. Sin embargo, es importante tener cuidado ya que es fácil configurar erróneamente OpenSSH provocando que alcance un estado inseguro.

De manera más general, este capítulo ha mostrado cómo instalar un programa servidor independiente genérico o un demonio y cómo engancharlo al sistema operativo. Al completar la lectura este capítulo, ha adquirido una cierta perspectiva sobre cómo se instalan otros programas similares, y también ha conseguido algo de experiencia práctica en la instalación y configuración adecuada de un programa bastante complicado. El siguiente capítulo mostrará una técnica diferente de instalación de software, utilizando los Módulos de Autenticación Enlazables como ejemplo.



9
Módulos de autenticación enlazables (PAM)

El sistema de módulos de autenticación enlazables (*Pluggable Authentication Modules*, PAM) es una API generalizada para la autenticación del sistema. Fue desarrollada originalmente por Sun Microsystems y se realizó una versión de este sistema para los sistemas Linux. Este capítulo muestra una breve perspectiva general sobre PAM y describe como se trabaja con este sistema.

El término autenticación se refiere a la tarea de confirmar que un determinado usuario es quien dice ser (por ejemplo, si la identidad del usuario es "auténtica"). En general es una bonita palabra nueva para el viejo concepto de la combinación de nombre de usuario y contraseña a la que casi todos los usuarios de ordenadores se han acostumbrado. Sin embargo, hay formas adicionales de credenciales que pueden utilizarse para la autenticación además de las contraseñas, como las claves criptográficas o dispositivos hardware como las tarjetas inteligentes o los objetos llave. Además, incluso el método tradicional de autenticación de usuario/contraseña tiene formas más sofisticadas, como los sistemas Kerberos y de servicios de información de red (*Network Information Services*, NIS).

Cualquier programa que necesite utilizar un mecanismo de autenticación en particular necesita ser escrito de tal modo que pueda soportarlo. Por ejemplo, para poder soportar el sistema Kerberos, un programa necesita codificarse de acuerdo con la API de Kerberos. Del mismo modo, cualquier sistema en el que se vaya a instalar ese programa necesitará que Kerberos esté instalado correctamente. Dado que hay múltiples esquemas de autenticación distintos de uso común, esto se convierte rápidamente en un problema tanto para los desarrolladores de software como para los administradores de sistemas.

El propósito de PAM es facilitar el uso y la gestión de los esquemas de autenticación. PAM proporciona una API software que los desarrolladores usan cuando escriben sus aplicaciones. Esta API es genérica y no implica ningún conocimiento sobre los detalles de un esquema de autenticación en particular. Además, PAM ofrece un mecanismo para instalar módulos (denominados, de forma suficientemente intuitiva, módulos PAM) que ofrecen sus servicios para un determinado esquema.

Por ejemplo, habría un módulo PAM para el sistema Kerberos comentado anteriormente y otro para el esquema Unix tradicional de usuario/contraseña. Sin embrago, un programa como login, que necesite realizar un proceso de autenticación no está escrito para utilizar un sistema, sino para utilizar PAM. El mismo programa login puede utilizar el esquema de Kerberos o el más básico de usuario/contraseña en función de los módulos PAM que haya instalado el administrador del sistema.

PAM es conceptualmente bastante sencillo. Las siguientes secciones comentarán los matices de la instalación y configuración de PAM. Hay esencialmente tres tareas de este estilo: instalar el propio sistema PAM, configurar el sistema PAM con módulos PAM específicos y configurar las aplicaciones de usuario para que utilicen los módulos PAM instalados.

Nota: Sólo los hechos: Módulos de autenticación enlazables

Propósito: La arquitectura Linux de módulos de autenticación enlazables ofrece una solución modular y de fácil configuración para la autenticación flexible de un sistema.
Autores: Andrew Morgan y otros.
Sitio web: www.kernel.org/pub/linux/libs/pam
Descripción: La arquitectura PAM proporciona dos servicios. En primer lugar, permite que las aplicaciones software se escriban con la idea en mente del soporte de una autenticación genérica. Esto permite al software cliente utilizar transparentemente cualquier mecanismo de autenticación instalado en el sistema sin necesidad de conocer los detalles específicos del mecanismo, lo que hace que el software cliente sea más flexible. En segundo lugar, PAM permite que los administradores del sistema instalen fácilmente módulos para mecanismos de autenticación específicos sin tener que recompilar o reconfigurar ninguna otra aplicación. Entre los ejemplos de mecanismos de autenticación que soporta PAM se incluyen los simples pares usuario/contraseña de Unix, Kerberos, credenciales de usuario almacenadas en LDAP, y muchos más.

Instalación de PAM

Probablemente esté bastante claro que PAM es un sistema de bajo nivel. Es decir, PAM es utilizado por muchos otros programas, y por eso es una parte importante

de una distribución. PAM está realmente al mismo nivel que otras bibliotecas centrales del sistema, como las bibliotecas glibc del Proyecto GNU o XFree86.

Por esta razón, la instalación de PAM es algo de lo que tiene que encargarse realmente el fabricante de la distribución. Para ver por qué, considere lo que tendría que hacer un administrador para instalar PAM. Obviamente, el primer paso sería instalar las propias bibliotecas PAM; sin embargo, después de eso habría que recompilar cualquier programa que tuviera que utilizar PAM para que pudiera hacerlo. Lo que significa tener que reinstalar un número potencialmente significativo de aplicaciones. Esto supone un esfuerzo notable, y por eso si se debe utilizar PAM en una distribución, tienen que ser instalado en las primeras fases del proceso.

Por ello, este libro no describirá propiamente el proceso de instalar PAM. Realmente es un proceso sencillo y similar a la instalación de cualquier otro paquete de software. No obstante, implica tanto esfuerzo y manejar a un nivel tan bajo la distribución Linux que es una tarea que es mejor dejársela al fabricante. Si su distribución viene con las bibliotecas de PAM instaladas, entonces las siguientes secciones pueden resultarle de interés. Si no es así, probablemente no merezca la pena que intente instalarlas.

Por supuesto, este es un capítulo sobre la instalación de software, ¿verdad? ¿Si no voy a comentar la instalación de PAM, para que lo menciono siquiera? El motivo es que la instalación de otros programas frecuentemente implica la interacción con PAM. Es decir, puede que necesite realizar algunos cambios de configuración relacionados con PAM cuando instale un programa que necesite realizar autenticación de usuario (como en el caso de OpenSSH o incluso el servidor web Apache si se cargan los módulos apropiados). La siguiente sección tratará sobre esta tarea.

Configuración de PAM

Esta sección se centra en la configuración de PAM para que soporte y trabaje adecuadamente con una aplicación. Esta sección tiene dos objetivos. Obviamente, el primero es explicar cómo configurar y utilizar PAM. El segundo objetivo es un poco más profundo. El objetivo global de este capítulo es mostrar y comentar una serie de técnicas de configuración habitualmente utilizadas por los programas. PAM utiliza una técnica determinada, y por eso es un buen ejemplo que añadir a la mezcla. Tras la lectura de esta sección, no sólo comprenderá PAM, si no que también habrá visto una técnica de configuración distinta que algún día le ayudará a comprender por sí mismo cómo funciona un nuevo programa.

Construcción del archivo de configuración

PAM puede utilizar dos mecanismos de archivos de configuración distintos. El primero es un simple archivo de configuración "plano" en el directorio /etc, y el segundo es el enfoque de directorios de depósito de archivos de configuración. El formato del archivo es parecido en ambos casos.

Nota

El enfoque de directorios de depósito de archivos de configuración se comentó extensamente en los capítulos 3 y 4. El enfoque del archivo plano se comentará a lo largo de este capítulo.

En el primer caso, todas las reglas que gobiernan el comportamiento de PAM en un sistema en particular están contenidas en el archivo /etc/pam.conf. En el segundo caso, los archivos que contienen las reglas de PAM para un servicio en particular se encuentran en el directorio /etc/pam.d. Si el directorio /etc/pam.d se encuentra en el sistema, entonces anula el comportamiento del archivo /etc/pam.conf, si es que este último existe; es decir, PAM ignora /etc/pam.conf a favor del directorio /etc/pam.d, si es que existe.

En ambos casos, PAM se configura con reglas que se encuentran en los archivos, donde cada una de las líneas define una regla. Como es tradicional en los archivos de configuración, el carácter almohadilla (#) indica un comentario en el archivo (significa que cualquier texto tras el símbolo # se ignorará). Cada regla está ligada a un servicio en particular (como login para el servicio de entrada en el sistema o ftp para un servidor FTP).

Un programa que utiliza PAM se identifica a sí mismo mediante un nombre de servicio. Entonces PAM inspecciona su configuración y extrae y aplica cualquier regla que se haya definido para ese servicio. PAM también tiene un servicio que hace de cajón de sastre llamado "*other*" (otros) y se ejecuta cuando un programa se identifica a sí mismo como un servicio que PAM no puede reconocer.

Estas reglas extraídas se ejecutan para determinar si el uso de la aplicación está autenticado adecuadamente o para realizar otras tareas como la gestión de cuentas o sesiones. La siguiente sección comentará el modo en que las reglas trabajan conjuntamente e interactúan para realizar estas tareas: el resto de esta sección describe el formato de la línea de una regla.

Cada regla tiene el siguiente formato:

```
servicio    tipo    control    módulo    argumentos
```

Cada elemento se comenta en las secciones siguientes.

El elemento servicio

El servicio es el nombre con el que se identifica a sí mismo un programa. Este nombre se utilizará para encontrar las reglas pertinentes para la aplicación. Si se utiliza el archivo /etc/pam.conf, entonces el elemento servicio indica a qué servicio se aplica la regla.

Si se utiliza el directorio /etc/pam.d en lugar del archivo único /etc/pam.conf, sin embargo, las reglas para un determinado servicio se encontrarán entonces en un archivo cuyo nombre será el mismo que el del servicio, y no se utilizará el elemento servicio.

Por ejemplo, las reglas para el servicio login estarían contenidas en un archivo con el nombre /etc/pam.d/login; ya que el nombre del servicio está implícito en el nombre del archivo, se puede omitir el elemento servicio de las reglas individuales contenidas en el archivo. Por el contrario, si se utiliza el archivo /etc/pam.conf, las reglas de los distintos servicios están totalmente mezcladas y se necesita el elemento servicio para distinguirlas.

El elemento tipo

El tipo define la propiedad sobre la que actúa la regla. El elemento tipo debe ser uno de estos valores:

- **auth**: La regla afecta a la autenticación del usuario de la aplicación.
- **session**: La regla afecta al entorno de la aplicación durante toda la sesión de entrada en el sistema.
- **account**: La regla gobierna algún aspecto permanente de la cuenta del usuario (diferente de la contraseña), como si la cuenta.
- **password**: La regla gobierna la comprobación, cambio, u otro tipo de interacción con la cuenta del usuario.

El tipo *auth* se utiliza para determinar si el usuario es auténtico. Típicamente, las reglas *auth* se utilizan para validar físicamente credenciales. Esto podría implicar comparar una contraseña para ver si es la misma que la contraseña oficial, validar un "símbolo" (*token*) generado por un sistema basado en una tarjeta inteligente, o efectuar la autenticación contra algún sistema externo como un servidor Kerberos. Una regla de autenticación también puede realizar otras comprobaciones; por ejemplo, es habitual encontrar reglas *auth* que permiten el acceso automático si el usuario ya es el usuario *root*.

El tipo *session* típicamente realiza trabajos auxiliares necesarios para el funcionamiento de un programa. Por ejemplo, los programas que necesitan utilizar X pueden tener una regla *session* que configure una entrada xauth para un usuario recién autenticado. Una regla de sesión alterna el entorno de la aplicación.

El tipo *account* indica que la regla está relacionada con la administración de la cuenta de usuario actual. Las reglas de cuentas rigen estrictamente la cuenta y comprueban cosas como si la cuenta existe, a qué horas del día o de la noche se permite acceder a esa cuenta a un usuario y demás. Un uso habitual de una regla *account* es comprobar si la cuenta del usuario ha sido inhabilitada y notificárselo al usuario (y denegar el acceso) si se da ese caso.

El tipo *password* indica que la regla está relacionada de algún modo con la administración de la contraseña del usuario, como la expiración de una contraseña o comprobar si una contraseña es nula (por ejemplo, está vacía). Un uso bastante común es incluir una regla *password* que compruebe si la contraseña de la cuenta ha expirado y solicite al usuario que modifique la contraseña o incluso deniegue completamente el acceso.

El elemento *control*

El elemento control define la acción que debe llevar a cabo una regla, como permitir o rechazar el acceso. El elemento `control` tiene dos formas: una forma breve (también denominada histórica), y una forma más complicada y potente. La forma más larga es un poco más potente, pero generalmente sólo se utiliza en las ocasiones en que se necesitan reglas muy complicadas. Ya que este no es generalmente el caso de las estaciones de trabajo ni incluso de los servidores, este libro comentará sólo la forma corta. Los lectores que estén interesados deberían consultar la página man sobre PAM para obtener más información sobre la forma larga.

La forma corta del elemento control debe ser una de estas cuatro palabras siguientes:

- **required**: Indica que el módulo PAM debe de tener éxito para que toda la pila de módulos tenga éxito.
- **requisite**: Es igual que *required* pero finaliza la pila PAM inmediatamente (las pilas PAM se comentarán en la siguiente sección).
- **sufficient**: Indica que el éxito de esta única regla es suficiente para que la pila tenga éxito, y no se necesitará ejecutar más reglas de la pila.
- **optional**: Indica que el éxito o el fallo de esta regla no es relevante (a no ser que sea la única regla).

Una regla *required* (necesaria) debe de tener éxito, o si no fallará toda la pila PAM. Con una regla *requisite* (requisito) sucede lo mismo, pero un requisito fallado causa que todas las reglas subsiguientes sean ignoradas, mientras que una regla *required* permite que se ejecuten incluso si la pila en su conjunto fallará. (Véase la siguiente sección para obtener más información acerca de las pilas PAM.) Típicamente, es mejor utilizar *required* que *requisite*, ya que el propio acto de finalizar un pila muy pronto puede proporcionar información a un atacante. Por ejemplo, si nunca solicita una contraseña porque falló la comprobación de si una cuenta existe, entonces el atacante sabe inmediatamente que tiene que dejar de perder el tiempo con esa cuenta.

Una regla *sufficient* (suficiente) es capaz de "hablar en lugar" de la pila al completo (es decir, el éxito de esta regla es "suficiente" para aprobar la pila completa). Una regla *optional* (opcional) siempre se ejecuta, pero su éxito o fallo no resulta relevante para la pila. (Sin embargo, si la regla opcional es la única regla en la pila, entonces obviamente es la única regla que puede aplicarse, y por eso no es realmente "opcional" y el éxito de la pila se basa en el de la única regla opcional.)

En general, la mayoría de las reglas *auth* en una pila se marcarán como necesarias (*required*) para asegurar que tienen éxito antes de permitir el acceso. La mayoría de las reglas *session* se marcarán como opcionales (*optional*) ya que normalmente sólo realizan trabajo adicional por motivos de conveniencia. Las reglas que se marcan como suficientes (*sufficient*) son típicamente reglas *auth*, ya que en esencia circunvalan al resto de la pila.

El elemento módulo

El módulo es una ruta hacia el módulo PAM que se va a utilizar para ejecutar físicamente la regla. Un módulo PAM es realmente un objeto dinámico compartido estándar que implementa la API PAM. La biblioteca principal PAM (que es llamada por la aplicación) carga la biblioteca compartida y después ejecuta su contenido. Los módulos PAM suelen encontrarse en el directorio /lib/security. (Fíjese en que están en la partición raíz en lugar de en la partición /usr, ya que el sistema los necesita durante el proceso de arranque.)

Los módulos PAM pueden ser instalados por el administrador del sistema para soportar alguna característica deseada, y cada módulo generalmente tiene una función separada. Por ejemplo, podría haber un módulo PAM que se encargase de la autenticación estándar de Unix de pares usuario/contraseña, y otro que se encargase de la autenticación Kerberos. Además, también podría haber módulos que manejasen reglas de sesión y de cuenta. Básicamente, el elemento módulo define la acción específica de autenticación, de sesión o de cuenta que hay que realizar.

Nota

Un determinado módulo PAM puede soportar múltiples tipos de pila; la mayoría realmente lo hacen.

El elemento argumentos

Este elemento se utiliza para pasar argumentos adicionales al módulo PAM especificado en la regla. El formato y el significado de cualquier argumento es específico del módulo PAM. Cada módulo puede definir sus propios argumentos específicos, y el elemento argumentos simplemente proporciona una manera para que los módulos PAM accedan a sus propios parámetros particulares.

Las pilas PAM

Cuando una aplicación intenta autenticar a un usuario, PAM busca las reglas para el servicio y tipo pertinente y las ejecuta en orden. Este conjunto de reglas que se aplican a una ejecución dada se denomina una pila de reglas PAM. Esta sección comenta cómo se ejecutan las pilas PAM y cómo circula el flujo de control entre ellas. La siguiente sección comentará un archivo de ejemplo en detalle.

Una pila PAM es simplemente una lista de reglas. Cuando todo funciona de forma normal, PAM se limita a procesar cada regla de acuerdo con su turno, en el orden en que se definió. Si cada regla PAM se ejecuta con éxito, entonces la pila al completo tiene éxito, y las llamadas de la aplicación a PAM tienen éxito. Sin embargo, cuando no todo es normal, puede suceder una variedad de cosas.

El caso más obvio acontece cuando falla una regla marcada como necesaria (*required*). Por ejemplo, el usuario puede introducir una contraseña incorrecta para

la cuenta, lo que provocaría que PAM fallara en su ejecución. En ese caso, cualquier regla que quede por ejecutarse aún lo haría, pero la propia fila fallaría, sin importar que las reglas posteriores tuvieran éxito. Este comportamiento es útil ya que habitualmente no querrá anular otras reglas simplemente porque una falló; por ejemplo, una regla podría ser una regla *account* que debe crear una entrada en un archivo de anotaciones, sin importar si la entrada en el sistema fue correcta.

Un caso similar ocurre cuando una regla requisito (*requisite*) falla. Este tipo de regla funciona exactamente como una regla necesaria, pero finaliza la ejecución de la pila inmediatamente, y no se ejecutarán las reglas posteriores. Este concepto de "retorno anticipado" es útil para casos en que se sabe que el fallo de una regla PAM permite obviar el resto. Por ejemplo, si una pila consiste en varias reglas de cuenta y la primera falla porque la cuenta no existe, está garantizado que el resto fallará y no hay ningún motivo para ejecutarlas. (No obstante, en la práctica esta es una mala idea, ya que el simple acto de finalizar una pila prematuramente puede proporcionar a un atacante información útil, como se mencionó anteriormente.)

Las reglas que son suficientes (*sufficient*) son similares a las reglas requisito en el sentido de que finalizan la ejecución de la pila; sin embargo, el significado de una regla suficiente es precisamente el opuesto de una regla requisito. Una regla suficiente sirve por sí sola para que la pila completa tenga éxito, y no hay razón alguna para molestarse en ejecutar el resto de la pila. El uso más común de este tipo de reglas probablemente sea permitir el acceso inmediato e incondicional para el usuario *root* (o a otro usuario, como la cuenta del usuario personal de administración en una estación de trabajo). Después de todo, si ya tiene la cuenta más poderosa del sistema, no hay ningún motivo para comprobar si puede volverse más poderosa.

Una regla opcional (*optional*) es, en cambio, algo diferente. La mayor parte del tiempo, las reglas opcionales no tienen impacto en las operaciones de la pila. Una regla opcional puede fallar y no repercutirá en el éxito global de la pila. Sin embargo, hay una excepción a ese comportamiento, que es cuando la regla opcional es la única regla de la pila. En ese caso, PAM no tiene otra regla para determinar el éxito o el fallo de la pila, y por eso no tiene otra opción más que utilizar la regla opcional.

La clave para la comprensión y utilización de las pilas de reglas PAM es entender las condiciones que pueden utilizarlas para desviarse del caso "normal". La idea general puede resumirse como sigue:

- PAM agrupa las reglas en /etc/pam.conf (o /etc/pam.d/*) en "pilas" según su servicio y tipo.
- PAM ejecuta la pila apropiada cada vez que una aplicación solicita un servicio en particular.
- Cada regla de la pila se ejecuta, a no ser que una regla requisito (*requisite*) o suficiente (*sufficient*) interrumpa el proceso.

Este modelo utilizado por PAM es realmente bastante común, especialmente en los sistemas relacionados con la seguridad. Por ejemplo, el núcleo Linux (y de hecho la mayoría de los sistemas capaces de configurar cortafuegos) utiliza un

paradigma a grandes rasgos similar para ejecutar sus guiones de cortafuego basados en iptables. Otro ejemplo es la funcionalidad de lista de control de acceso que soportan muchos sistemas de archivos de discos (como el sistema de archivos de red AFS). Este tipo de configuración puede ser notablemente sutil y delicada de comprender, precisamente porque es extremadamente potente. El truco para trabajar con este tipo de configuración es comprender las condiciones que pueden causar una desviación de la norma y considerar detenidamente todas las posibilidades. Además, cuantas menos reglas, mejor, ya que el menor número de reglas ayuda a la simplicidad, y la simplicidad lleva a la robustez del sistema.

Análisis de un archivo de ejemplo

El listado 9.1 es una copia del archivo /etc/pam.d/up2date procedente de un sistema Red Hat Linux 7.3. El programa up2date es una herramienta de mantenimiento del sistema desarrollada por Red Hat, y se comenta en el capítulo 4. up2date es una herramienta de mantenimiento del sistema, de modo que necesita permisos del usuario *root* para ejecutarse, y por eso tiene necesidades de seguridad que constituirán un buen ejemplo de uso de PAM. Este archivo es un ejemplo típico sobre cómo se utiliza PAM en la práctica. Cada línea del archivo se describe individualmente. Fíjese, sin embargo, en que los propios módulos PAM no se comentan exhaustivamente.

Listado 9.1. Ejemplo de archivo de configuración de PAM.

```
#%PAM-1.0
auth       sufficient    /lib/security/pam_rootok.so
auth       required      /lib/security/pam_stack.so service=system-auth
session    required      /lib/security/pam_permit.so
session    optional      /lib/security/pam_xauth.so
account    required      /lib/security/pam_permit.so
```

La primera línea es simplemente un comentario, e incluye información sobre la versión de PAM que necesita el archivo. El resto de las líneas define tres pilas: una pila *auth*, una pila *session*, y una pila *account*. Si se da cuenta, estas líneas no incluye el elemento servicio; es así porque la distribución de Red Hat utiliza el mecanismo /etc/pam.d. Si Red Hat hubiera elegido utilizar un único archivo /etc/pam.conf en su lugar, entonces cada línea tendría "up2date" como prefijo.

La pila *auth* comprueba en primer lugar el módulo PAM pam_rootok.so y después el módulo pam_stack.so. El primer módulo comprueba si el usuario ya es *root* y resulta exitoso si es así. Sin embargo, el segundo módulo provoca una ramificación a una segunda pila, denominada servicio "system-auth". Este es un archivo separado (en /etc/pam.d/system-auth) que contiene una pila PAM completa que simplemente extrae y autentica un par usuario/contraseña (al igual que realiza una gran variedad de comprobaciones en busca de contraseñas que hayan expirado y cosas parecidas).

Esta técnica permite que las pilas que utilizan habitualmente muchos servicios PAM (como la verificación de contraseñas de este ejemplo, que es tan común) se

puedan almacenar en un único archivo y reutilizarse, en lugar de tener que copiar sus reglas en cada archivo que las necesite.

En esta pila para up2date, el caso normal es que el usuario debe autenticarse mediante un nombre de usuario y una contraseña, que es algo manejado por la pila PAM system-auth mediante la línea de pam_stack.so. No obstante, antes de que esta regla pueda ejecutarse, se da a la primera regla (marcada como suficiente) la posibilidad de interceptar la ejecución. Esta regla simplemente comprueba si es el usuario *root* el que ejecuta el programa up2date, en cuyo caso la pila tiene éxito inmediatamente.

La pila de sesión es un poco extraña a primera vista. Normalmente, se ejecuta el módulo PAM pam_permit.so, al igual que el módulo opcional pam_xauth.so. Si el primero falla, entonces el segundo se ejecutará pero la pila global fallará. La regla opcional intenta llevar a cabo la autenticación de sesión xauth para X. Sin embargo, el módulo PAM pam_permit.so siempre tiene éxito; no puede fallar. El motivo de que esté ahí es porque el módulo pam_xauth.so puede fallar con gran facilidad (por ejemplo, si el usuario no está utilizando X). Ya que esta es una regla opcional, si fuera la única regla de la pila, podría provocar que la pila global fallara innecesariamente. Por eso, ahí está la regla pam_permit.so para asegurar que la pila al completo siempre tenga éxito. (Recuerde que el éxito o fallo de una regla opcional se ignora si hay otras reglas en la pila.)

La tercera pila es la pila *account*, que sólo tiene una regla. Esta regla vuelve a referirse al módulo pam_permit.so, que siempre tiene éxito. Esta regla sólo está aquí ocupando espacio, para asegurarse de que la pila de cuenta no falle. (Esta regla realmente no hace mucho, ya que no se necesita obligatoriamente en absoluto una pila *account*, pero es una buena medida que esté ahí.)

Este es un ejemplo bastante sencillo de un archivo de configuración de servicio PAM. Otros archivos PAM pueden volverse más complicados, pero mientras recuerde los conceptos básicos sobre cómo funciona PAM, no deberían ser muy difíciles comprender. En la mayoría de los casos, no tendrá que escribir sus propios archivos de configuración PAM, pero alguna vez sí tendrá que ser así, y una buena comprensión de PAM es algo muy positivo para aquellos casos en los que necesite leer un archivo PAM.

Nota: ¡No olvide el servicio *other*!

Hay un servicio especial llamado "*other*" (otro) que PAM define como el servicio por defecto. Es decir, si una aplicación especifica un nombre de servicio que PAM no puede reconocer, entonces entra en juego el servicio *other*. Mediante la configuración del servicio *other*, puede controlar los derechos de acceso que se conceden a programas desconocidos.
Es una buena idea adoptar un enfoque conservador cuando se enfrente con un sistema como PAM y fijar su regla por defecto de un modo extremadamente restrictivo. En primer lugar deniéguelo todo, y luego apruebe sólo permisos específicos para servicios específicos; de ese modo, no otorgará accidental-

mente permisos que no quería ofrecer. (Esta regla también se aplica generalmente a cosas como reglas de cortafuego y controles de acceso a archivos.) En el contexto de PAM, es una buena idea hacer muy restrictivas las reglas para el servicio *other*. Por ejemplo, las siguientes reglas provienen de la definición por defecto del servicio *other* para Red Hat Linux. Estas reglas forman un conjunto por defecto bastante razonable.

```
auth            required    /lib/security/pam_deny.so
account         required    /lib/security/pam_deny.so
password        required    /lib/security/pam_deny.so
session         required    /lib/security/pam_deny.so
```

Instalación de PAM en diferentes distribuciones

Recuerde que en general PAM es un mecanismo de un nivel demasiado bajo como para instalarlo en un sistema una vez instalado éste, al menos sin una cantidad de esfuerzo significativa para recompilar los programas existentes. En lugar de describir cómo instalar PAM, las siguientes secciones comentarán algunos matices del uso de PAM en las distribuciones de ejemplo.

Instalación de PAM en Red Hat Linux

PAM viene instalado por defecto en Red Hat Linux 7.3. Siguiendo con la misma tónica del uso extenso de Red Hat de los directorios de configuración mediante depósito de archivos, la instalación de PAM en Red Hat utiliza el enfoque de /etc/pam.d para configurar los servicios individuales.

Típicamente, cada RPM que instala un servicio que utiliza PAM instala también su propio archivo de configuración de servicio en /etc/pam.d. Esto posibilita que la herramienta rpm instale y desinstale archivos de configuración de PAM adicionales. Otros programas que instale a partir del código fuente podrían incluir un archivo de configuración para PAM para ese programa; en estos casos, sólo necesitará copiar el archivo en /etc/pam.d.

Instalación de PAM en Slackware Linux

Slackware Linux 8.0 no incluye PAM junto con la distribución. Obviamente es posible instalar PAM después de la instalación de Slackware Linux, pero ese es un compromiso que sólo se puede dejar al administrador. Hay un punto de retornos decrecientes, e instalar PAM en una distribución ya existente podría estar más allá de este punto.

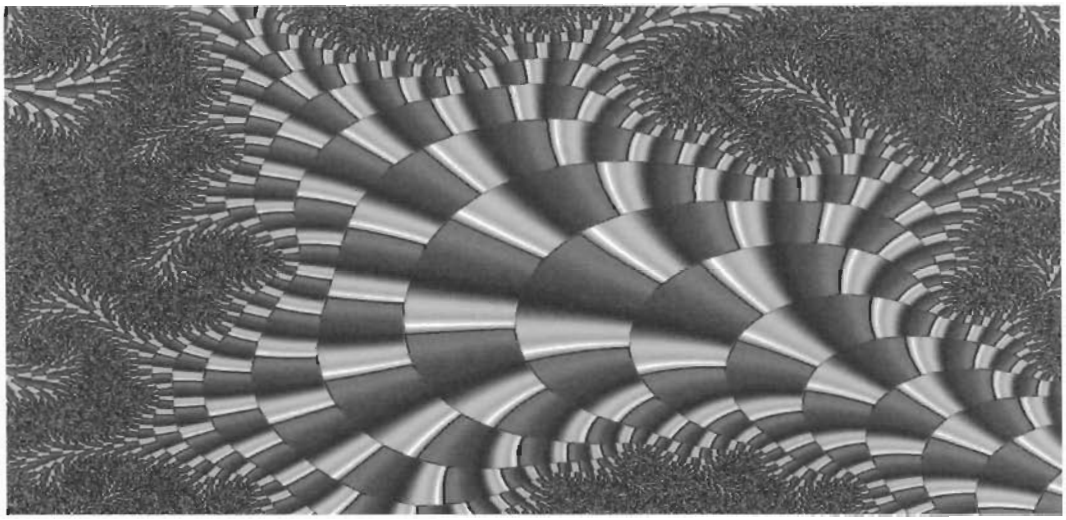
Instalación de PAM en Debian GNU/Linux

Como Red Hat Linux, Debian GNU/Linux viene junto con una instalación de PAM. De hecho, la configuración de Debian de PAM es muy similar a la de Red Hat: también utiliza el directorio /etc/pam.d y el paradigma de depósito de configuraciones. En otras palabras, todas las notas de la sección anterior que comentaba el uso de PAM con Red Hat Linux también se aplican a Debian GNU/Linux. Con toda probabilidad, la única diferencia que encontrará es que la lista real de servicios que se refleja en /etc/pam.d es algo diferente en las dos distribuciones y también dependen obviamente de los paquetes que se hayan instalado. Además, los contenidos de los archivos de servicios podrían ser sustancialmente diferentes.

Resumen

En este capítulo ha leído acerca de cómo funciona PAM junto con el sistema operativo y cómo configurarlo. Ha aprendido acerca de los tipos de información que se necesita para la configuración de PAM y el modo en que esa información se estructura en los archivos. Llegados a este punto, debería ser capaz de leer y comprender el funcionamiento de cualquier archivo de configuración de PAM que pueda encontrarse.

Hasta ahora ha visto dos capítulos referentes al tema de la configuración. En ellos ha visto dos aplicaciones software muy diferentes, que ofrecen dos perspectivas distintas ante problemas similares. Cuando finalice la lectura de los cuatro capítulos siguientes, habrá visto ejemplos de los métodos más habituales de configuración de software y estará bien preparado ante cualquier tarea de instalación que se le presente.



10
Biblioteca SOCKS Dante

Este capítulo describe cómo instalar la biblioteca cliente de SOCKS Dante. Dante proporciona soporte para el protocolo de cruce de cortafuego (*firewall-traversal*) SOCKS para aplicaciones cliente. Un cortafuego (*firewall*) es un dispositivo de red que usan frecuentemente empresas o usuarios individuales para proteger una red de ordenadores de atacantes hostiles. Típicamente, se fuerza a todo el tráfico de red que va y viene desde Internet a pasar por un único punto. En este punto se instala un cortafuego y se bloquea todo acceso a la red protegida, excepto quizás a un conjunto de servicios específicos y férreamente controlados. El inconveniente potencial de un cortafuego es que suele funcionar en ambos sentidos: también impide que los usuarios que estén dentro del cortafuego accedan directamente a recursos de Internet.

Normalmente se utiliza un servidor proxy (un servidor representante) para ofrecer a los usuarios internos acceso a la World Wide Web. Sin embargo, los servidores proxy sólo son útiles para la web, y no ayudan a otras aplicaciones. Por ejemplo, como el protocolo SSH no funciona sobre el protocolo de web http, podría resultar bloqueado, con lo que se impide que los usuarios que se encuentren tras un cortafuego puedan utilizar un programa como OpenSSH para acceder a una máquina de Internet. El protocolo SOCKS viene a ocupar este hueco, actuando como una especie de servidor proxy genérico para todos los servicios (no sólo la web).

Los administradores de red pueden instalar un servidor SOCKS en una red protegida por cortafuegos. Este servidor debe "eludir" el cortafuego, en el sentido de que tiene acceso tanto a la red interna como a Internet. Los programas clientes que son capaces de comunicarse mediante el protocolo SOCKS pueden configurar-

se para conectarse al servidor SOCKS, que funciona como repetidor. El cliente solicitará una conexión a un recurso en particular y el servidor SOCKS actuará como intermediario, retransmitiendo los mensajes entre el cliente y la máquina de Internet. Todo lo que sabrá la máquina de Internet es que se está comunicando sólo con el servidor SOCKS, pero en cualquier otro sentido la conexión será transparente. De este modo, el protocolo SOCKS es un protocolo de "túnel" (*tunneling protocol*) que permite al programa cliente "abrir un túnel" a través del cortafuego.

Puesto que el protocolo SOCKS permite que las máquinas de una red protegida que estén tras un cortafuego puedan acceder arbitrariamente a servicios de Internet, puede verse como un riesgo de seguridad, aunque un riesgo muy reducido porque sólo permite las conexiones salientes y no las entrantes. Por esta razón, no todas las redes con un cortafuego disponen de un servidor SOCKS, así que si se encuentra tras un cortafuego, tendrá que preguntar a su administrador de red acerca del acceso SOCKS. Incluso aunque se esté ejecutando un servidor SOCKS, puede estar configurado para restringir los tipos de conexión que realmente permite.

Entonces, ¿qué es Dante? Para decirlo llanamente, Dante es tanto un servidor SOCKS como una biblioteca de cliente SOCKS. El programa servidor Dante puede instalarse en un sistema y utilizarse para preparar un servidor SOCKS que soporte el cruce de cortafuegos, tal cual se describió anteriormente. La biblioteca de cliente, por su parte, ofrece a los programas cliente un modo de atravesar el cortafuego. Ya que la tercera parte de este libro contiene varios ejemplos más sobre la instalación de software de servidores, comentar el servidor SOCKS Dante resultaría redundante. Como resultado, este capítulo se centra en mostrar la instalación de una biblioteca de cliente y no se comenta el servidor Dante. Con la lectura de este capítulo, verá un ejemplo muy típico sobre cómo se configura el software que funciona como biblioteca compartida. Este tipo de software no es especialmente habitual, pero, aún así, es probable que se encuentre ante este problema en algún momento, así que resultará útil verlo.

La biblioteca de cliente Dante viene como un objeto compartido de Unix. (Es una criatura similar a las bibliotecas de enlace dinámico de Windows [las DLL].) Específicamente, las bibliotecas de cliente Dante implementan una versión de la API de sockets BSD estándar compatible con SOCKS. Esta API es la API que utiliza la amplia mayoría de las aplicaciones de red de Unix, e incluso es utilizada en muchos otros sistemas operativos. Al reemplazar esta biblioteca con una versión que es capaz de utilizar transparentemente SOCKS, Dante puede habilitar casi cualquier programa para que utilice SOCKS.

Nota: Bibliotecas compartidas de Unix y Dante

Una biblioteca compartida (denominada en ocasiones objeto compartido o "shlib" [*shared library*]) es un archivo, normalmente con una extensión ".so", que contiene código software compilado que comparten múltiples programas en ejecución. El motivo para compartir código entre programas es reducir el uso de memoria: como cada programa comparte la biblioteca, hay una única

copia en memoria, lo que reduce la necesidad de memoria de los programas en ejecución, y por ello conserva la memoria RAM. Se dice de los programas que utilizan un objeto compartido dinámico están enlazados dinámicamente contra la biblioteca. (De los programas que enlazan su propia copia de una biblioteca se dice que están enlazados estáticamente.)
Otro beneficio de los objetos compartidos es que pueden actualizarse independientemente de las aplicaciones que los utilizan, por supuesto con algunas restricciones. No puede modificar completamente una API y esperar que los viejos programas sean capaces de utilizarla. Se dice que las bibliotecas compartidas son compatibles binariamente cuando pueden utilizarse sin modificaciones por parte de los programas que se compilaron contra una versión distinta. La biblioteca SOCKS Dante se aprovecha de esta característica y ofrece una versión consciente del protocolo SOCKS de la API de sockets BSD que es binariamente compatible con la versión estándar (no SOCKS). Esto hace posible habilitar el protocolo SOCKS para casi cualquier programa que utilice la red.

Algunas distribuciones incluyen Dante con la instalación estándar; por ejemplo, Red Hat Linux ha incluido el software como un paquete opcional en versiones anteriores (aunque este no es el caso de Red Hat Linux 7.3). Como siempre, compruebe si un paquete ya está instalado en su sistema o está disponible para el proceso de instalación antes de molestarse en realizar todo el trabajo comentado en este capítulo. Sin embargo, incluso aunque Dante ya esté instalado, lo más probable es que no esté configurado de manera adecuada. Como verá pronto, configurar Dante significa saber bastante acerca de la configuración su red, y por eso la configuración por defecto tenderá a resultar inadecuada. Por este motivo, la sección "Configuración de Dante" de este capítulo le parecerá útil, incluso aunque la biblioteca Dante haya sido instalada junto con su distribución.

Nota: Sólo los hechos: Biblioteca SOCKS Dante

Propósito: Proporciona soporte para el protocolo de cruce de cortafuego SOCKS.
Autores: Inferno Nettverk A/S, Norway
Sitio web: http://www.inet.no/
Dependencias: Ninguna
Descripción: Dante es una implementación del protocolo cruce de cortafuego SOCKS, que soporta las versiones 4 y 5. El protocolo SOCKS permite a los programas cliente que lo utilizan acceder a recursos de Internet en general, incluso si se encuentren tras un cortafuego. Dante ofrece una biblioteca genérica que puede hacer que casi cualquier programa se "socksie", ofreciendo un acceso transparente a Internet.

Instalación de Dante

En cuanto a paquetes de software se trata, Dante es bastante "limpio", en el sentido de que no tiene ninguna dependencia y no necesita ninguna opción en tiempo de compilación extraña. (Bueno, técnicamente Dante necesita varios archivos de bibliotecas y cabeceras de sockets BSD, pero hablaríamos de un sistema Unix muy extraño si no tuviera ya instalados estos archivos.) Por eso, instalar Dante es un caso de libro de texto sobre el uso del sistema autoconf de GNU para configurar e instalar un paquete de software. El resto de esta sección comenta esta instalación.

Preparación de la ruta de instalación

La única opción que necesita fijar es la ruta de instalación. Como se comentó en la sección "Decidir la ubicación" del capítulo 7, generalmente hay dos sitios buenos para instalar el software construido de manera particular: el directorio /usr/local o el directorio /opt. Como con OpenSSH (cuya instalación se comentó en un capítulo precedente), probablemente no espera actualizar frecuentemente Dante, ni necesitará tener instalada más de una versión al mismo tiempo. Por ello, /usr/local parece una buena ubicación para instalar Dante.

Un fallo menor en la configuración de Dante es que no fija automáticamente la ubicación de sus archivos de configuración para que se corresponda con su directorio de instalación. Normalmente, si un programa se instala en /usr/local se esperaría encontrar su archivo de configuración en /usr/local/etc. Sin embargo, Dante utiliza por defecto el directorio de configuración /etc. Se puede alterar esto si se utiliza la opción `--with-socks-conf=` en el programa ./configure, pero no se hace de manera automática, lo que resulta algo inusual. Para este ejemplo, ya que va a instalar el programa en /usr/local, quiere que el archivo de configuración sea /usr/local/etc/socks.conf (lo que se mostrará en breve).

> **Nota**
>
> Como no va a utilizar el servidor SOCKS Dante, no se moleste en fijar el archivo de configuración del servidor mediante la opción `--with-sockd-conf=`.

El siguiente comando, cuando se ejecute desde el directorio de mayor nivel del paquete de código fuente de Dante, configurará el software con la ruta /usr/local:

```
$ ./configure --prefix=/usr/local --with-socks-conf=/usr/local/etc/
socks.conf
```

Este comando configurará tanto la biblioteca de cliente como el servidor de Dante; sin embargo, como se mencionó anteriormente, aquí sólo estará interesado

en la biblioteca de cliente Dante. Resultaría ideal inhabilitar el componente de servidor de modo que ni siquiera se compile, pero lamentablemente los guiones autoconf de Dante no proporcionan un medio para hacer eso. Aún así, jamás se ejecutará el servidor, por eso aunque suponga un desperdicio de espacio de disco, no es un gran agujero de seguridad. Si aún está preocupado por esto, puede modificar los guiones por sí mismo o construir un paquete (como un paquete RPM o Debian) para excluir los archivos del servidor.

Compilación e instalación

Tras la configuración, Dante está listo para ser compilado e instalado. Una vez más, este es un ejemplo de libro de texto de instalación mediante autoconf de GNU, por eso seguir los siguientes comandos es muy simple. El primero compila el software, y el segundo realmente lo instala:

```
$ make
```

(Aparecerá información sobre el proceso de compilación.)

```
$ make install
```

(Aparecerá información sobre el proceso de instalación.)

Resumen de la instalación

Dante es un ejemplo clásico de un paquete que utiliza el mecanismo autoconf de GNU. Se aplica la secuencia canónica de comandos `./configure; make; make install`, y no hay ninguna dependencia u opción en tiempo de compilación importante de las que los administradores tengan que ser conscientes. Sin embargo, al igual que es sencillo instalar Dante, su configuración puede ser bastante sutil, ya que necesita un cierto grado de conocimiento detallado acerca de la red. La siguiente sección explica la configuración de Dante.

Configuración de Dante

La instalación de Dante tal y como se describió anteriormente en este capítulo incluirá tanto el servidor SOCKS Dante como las bibliotecas de cliente Dante. Sin embargo, el servidor SOCKS Dante es un ejemplo de instalación de servidor bastante típico, y su proceso de configuración muy parecido a los otros servidores comentados en esta tercera parte (como los servidores Apache y OpenSSH). En consecuencia, este capítulo se centra en las bibliotecas de cliente Dante, y esta sección trata sobre su configuración.

Recuerde que las bibliotecas de cliente Dante son un sustituto binariamente compatible con la API de sockets BSD estándar. Esta es la API que utilizan casi

todos los programas de red de Unix. Las versiones de Dante de estas bibliotecas son "conscientes" del protocolo SOCKS, y por eso los programas que utilicen estas versiones tendrán automáticamente acceso al servidor SOCKS, incluso aunque no tengan ni idea de que no están utilizando las bibliotecas BSD estándar. Las propias bibliotecas de cliente Dante utilizan un archivo de configuración para encontrar el servidor SOCKS y extraer otros parámetros necesarios. Esta sección comenta esa configuración.

Encontrar el archivo de configuración

La ubicación estándar de Unix para los archivos de configuración es un subdirectorio de la ruta de instalación llamado "etc"; si instaló Dante a partir del código fuente tal y como se describe en la sección anterior, entonces la ubicación del archivo de configuración de Dante será /usr/local/etc. (Si está utilizando una instalación de Dante que se incluyó junto con su distribución, entonces el directorio de configuración puede variar, pero casi seguramente sea simplemente /etc.) El nombre estándar del archivo de configuración es socks.conf, aunque esto puede modificarse mediante una opción en tiempo de configuración del programa ./configure. Por eso, el nombre del archivo a lo largo de este ejemplo debería ser /usr/local/etc/socks.conf. Sin embargo, el proceso de instalación de Dante por defecto no instala un archivo de configuración (ni siquiera una copia por defecto) así que habrá que crear uno desde la nada.

Afortunadamente, Dante incluye archivos de configuración de ejemplo; sólo que no están instalados por defecto. Se encuentran en el subdirectorio *examples* en el paquete de código fuente de Dante. El archivo de configuración para los programas clientes se llama socks.conf, pero hay otros ejemplos, como socks-simple.conf. Para utilizar uno de estos archivos como plantilla para la creación de su propio archivo de configuración, simplemente cópielo como /usr/local/etc/socks.conf. Por ejemplo, el siguiente comando utilizará el archivo de ejemplo más simple como punto de partida.

```
$ cp examples/socks-simple.conf /usr/local/etc/socks.conf
```

También puede crear un archivo socks.conf a partir de la nada, aunque probablemente sea más fácil comenzar con una de las plantillas de ejemplo. En cualquier caso, ahora puede modificar la configuración simplemente editando /usr/local/etc/socks.conf.

Obtención de los parámetros de red

Configurar Dante correctamente requiere un poco de conocimiento acerca de cómo está configurada la red. Por ejemplo, ya que el protocolo SOCKS necesita la presencia de un servidor SOCKS, el nombre y número de puerto de la máquina que hace de servidor SOCKS deben de ser conocidos. Además, la versión del protocolo

SOCKS soportada por el servidor también debe conocerse. La tabla 10.1 incluye la lista de los campos de datos necesarios y explica brevemente sus significados; por lo general, deberá preguntar esta información al administrador de su red local. Sin embargo, este no es un libro sobre las redes o sobre el protocolo SOCKS, así que si necesita más detalles, deberá consultar la documentación existente sobre Dante.

Tabla 10.1. Campos típicos de configuración de Dante.

Opción	Valores posibles	Significado
versión de SOCKS	v4, v5, o msproxy	La versión del protocolo que se utiliza; normalmente se utiliza SOCKS v4 o SOCKS v5, o alternativamente el protocolo msproxy.
máscara de red local	Una máscara IP estándar (x.x.x.x)	El rango de direcciones IP que se usa dentro de la red local protegida (dentro del cortafuego).
dirección IP del servidor	Una dirección IP estándar	La dirección IP (no el nombre de la máquina) del servidor SOCKS.
número de puerto del servidor	Normalmente 1080	El número de puerto en la máquina del servidor en la que se ejecuta el proceso del servidor SOCKS.

Conceptualmente, el funcionamiento de Dante es bastante simple. Para cada conexión, compara la dirección IP de destino para ver si se corresponde con un patrón (la máscara de red), y si es así redirige la conexión a través del servidor SOCKS (o no) dependiendo de la configuración. Para la mayoría de las redes, la máscara de red es la red local y las peticiones locales no se redirigen a través del servidor SOCKS, mientras que las conexiones no locales (por ejemplo, a Internet) sí son redirigidas. Por eso, una vez que se ha reunido la información relevante sobre la red local, la dirección y el puerto del servidor y demás, se puede proceder a modificar el archivo de configuración.

Formato del archivo de configuración

El archivo de configuración de Dante es un poco más complejo que el de muchos otros software, pero sigue siendo bastante sencillo. Consiste en una sección delimitada por "llaves" (es decir, los caracteres "{" y "}"), donde cada sección habilita o inhabilita una característica en particular o define una "ruta". Cada línea dentro de cada uno de estos "bloques de ruta" configura una opción particular para esa ruta. También hay un conjunto de parámetros opcionales de una línea que pueden colocarse fuera de un bloque para configurar cosas como el soporte de anotaciones y de depuración. El archivo de configuración también permite incluir comentarios en el

archivo, que están destinados al usuario humano con el propósito de ofrecer documentación, y se ignoran durante el procesamiento del archivo. Como la mayor parte de los archivos de configuración de Unix, cualquier línea que comience con el carácter almohadilla (#) se considera un comentario y es ignorado.

Opciones globales

Dante soporta algunas opciones globales (es decir, opciones que son generales y existen independientemente de cualquier ruta particular). Estas opciones globales indican dónde apuntar la información de las anotaciones, si habilitar el soporte de depuración y el tipo de servidor de nombres de red que se utilizará. Por lo general, no querrá activar el soporte a la depuración, y dejar que el software tome sus anotaciones en la ubicación estándar resulta perfectamente correcto. Por eso, estas opciones pueden dejarse con sus valores por defecto (lo que quiere decir que no estarán presentes, o que estarán comentadas); sin embargo, si se necesitan pueden ser habilitadas tal y como se indica a continuación:

```
debug: 1
logoutput: stdout
```

Nota: Cuidado con los formatos de archivo

Los lectores observadores pueden haberse dado cuenta de que el formato para las opciones en el archivo /usr/local/etc/socks.conf de configuración de Dante es "opción: valor"; por ejemplo, la depuración (*debugging*) se activa mediante `debug: 1`, (Un valor de 1 es bastante habitual en los archivos de configuración para significar "auténtico" o "habilitado".) Compare este formato con el formato de OpenSSH descrito con anterioridad, donde cada línea tiene un formato como "opción valor" (como, por ejemplo, `StrictHostKeyChecking no`). Lamentablemente, no hay una manera estándar de fijar valores en los archivos de configuración de Unix, así que debe tener procurar seleccionar el formato correcto cuando trabaje con un archivo en particular. Si encuentra un error en su software que piensa que está relacionado con la configuración, es aconsejable comprobar el archivo para asegurarse de que el formato es correcto.

La opción restante es `resolveprotocol`, que indica a Dante si utilizar un protocolo TCP (basado en flujos), UDP (basado en datagramas), o ninguno para resolver los nombres de máquinas. Este es un asunto bastante críptico y varía de acuerdo con el servidor SOCKS y la red que se utilizan. Normalmente, los servidores SOCKS v5 deberían tener `resolveprotocol: tcp`, los servidores SOCKS v4 deberían utilizar `resolveprotocol: udp`, y las configuraciones sin servidor de nombres de red deberían utilizar `resolveprotocol: fake`. Aún así, hay excepciones, de modo que si necesita más información tendrá que consultar la propia documentación de Dante.

Las tres opciones de `debug`, `logoutput` y `resolveprotocol` son todas las opciones globales. Si no resultan necesarias (`debug` y `logoutput` no suelen serlo), pueden omitirse o comentarse en el archivo /usr/local/etc/socks.conf. El ejemplo de la sección siguiente muestra una configuración real como ejemplo, incluidas estas opciones.

Creación de un bloque de ruta

Las partes más importantes de un archivo de configuración socks.conf de Dante son los bloques de ruta. Un bloque de ruta consiste en un cierto número de líneas encerradas entre las líneas route { y }. (El uso de llaves para indicar "bloques" de líneas relacionadas es bastante común; más adelante en otro capítulo verá que el programa xinetd utiliza un formato similar.) Cada línea dentro de un bloque configura una opción diferente para ese bloque. La tabla 10.2 resume las opciones que se pueden fijar dentro de un bloque.

Tabla 10.2. Opciones habituales de bloques de ruta de Dante.

Nombre de opción	Significado
`from`	El patrón IP empleado para corresponderse con el origen de una conexión.
`to`	El patrón IP empleado para corresponderse con el destino de una conexión.
`via`	La dirección IP de un servidor SOCKS que se usará para manejar la conexión.
`prot`	Un número de puerto TCP/IP; puede utilizarse con `from`, `to` o `via`.
`proxyprotocol`	El protocolo que se usará (como SOCKS v4 o v5) para el bloque de ruta.

Hay ciertas opciones adicionales de bloques de ruta que son algo menos comunes que las de la tabla 10.2; no obstante, no se comentan aquí por razones de simplicidad. Si necesita más información, consulte, por favor, la documentación sobre Dante, y en especial la página del manual para el archivo socks.conf (a la que se puede acceder mediante el comando `man socks.conf`) y el archivo de configuración detallado de ejemplo llamado example/socks.conf que se encuentra en el paquete de código fuente de Dante.

Recuerde que la labor esencial de un bloque de ruta es indicar a Dante qué servidor SOCKS utilizar para una conexión dada (si es que se usa alguno). Las opciones `from` y `to` definen el origen y el destino, y la opción `via` especifica el servidor SOCKS que se utilizará (si se usa). El resto de las opciones simplemente indican qué protocolo SOCKS habrá que usar.

Un archivo de muestra

Esta sección mostrará un archivo socks.conf de ejemplo para ilustrar los puntos comentados anteriormente. Otras configuraciones de ejemplo se incluyen con el propio paquete de código de Dante dentro del subdirectorio example. También debería consultar estos ejemplos, ya que este no cubre todos los matices de la configuración de Dante. En su lugar, este ejemplo pretende mostrar la configuración para una estructura de red habitual. Específicamente, este ejemplo permite manejar este caso:

- La red local está en el rango IP 192.168.0.0 y todos los ordenadores de la red local están tras el cortafuego.
- El ordenador cortafuego está en la dirección IP 192.168.0.1.
- El cortafuego ejecuta un servidor SOCKS v4 en el puerto 1080.
- Todas las conexiones a la red local deberían ser conexiones directas (es decir, circunvalarían el servidor SOCKS), y todas las demás conexiones deberían pasar a través del servidor SOCKS.
- La depuración debería estar inhabilitada, y la salida de las anotaciones debería realizarse por defecto.

El listado 10.1 es un archivo socks.conf de muestra que cumple estos criterios. Esta estructura de red es realmente bastante típica en la red de una casa o una pequeña ofician que esté protegida mediante un cortafuego. (El rango de direcciones 192.168.0.0 suele usarse por parte de cortafuegos NAT [Traducción de Direcciones de Red, *Network Address Translation*].)

Listado 10.1. Ejemplo de archivo socks.conf.

```
debug: 0
# logoutput: stdout
resolveprotocol: tcp

# todo el tráfico del servidor de nombres (i.e. port=domain) debería
circunvalar el servidor SOCKS
route {
        from 0.0.0.0 to: 192.168.0.9 port=domain via: direct
}

# todo el tráfico hacia la máquina local debería circunvalar el servidor
SOCKS
route {
        from 0.0.0.0 to: 127.0.0.0/8 via: direct
}

# todo el tráfico hacia la red local debería circunvalar el servidor SOCKS
route {
        from 0.0.0.0 to: 192.168.0.0/8 via: direct
}
```

```
# todo el tráfico debería dirigirse al servidor proxy
route {
        from 0.0.0.0 to: 0.0.0.0 via: 192.169.0.1 port=1080
        proxyprotocol: socks_v4
}
```

Para la instalación de ejemplo de Dante, podría colocar el contenido del listado 10.1 en el archivo /usr/local/etc/socks.conf. Sin embargo, es muy posible que no funcionara, porque no es probable que su red se corresponda exactamente con la reflejada en el listado 10.1. Asegúrese de tener los datos de red correctos antes de intentar crear cualquier archivo de configuración de Dante, o probablemente acabará desperdiciando una gran cantidad de tiempo.

Instalación de Dante en diferentes distribuciones

Las siguientes secciones describen técnicas y tareas necesarias para la instalación de Dante en las tres distribuciones de muestra. El material anterior es genérico; resulta aplicable a cualquier distribución. Las secciones que vienen a continuación adaptan la información general y muestran instalaciones reales sobre distribuciones reales.

Instalación de Dante en Red Hat Linux

Recuerde que Dante es un ejemplo de libro de texto de una instalación. Sin embargo, la instalación de ejemplo de este capítulo instala el software en /usr/local, y el administrador de paquetes, RPM, no tiene modo de saber acerca de ella. Consecuentemente, RPM no puede administrar o proteger los archivos instalados. Como siempre, sería conveniente tener un paquete RPM real que se pudiera instalar en lugar de tener que construir uno manualmente. Por ello, si puede encontrar un paquete RPM preconstruido para Red Hat Linux, tal vez prefiera utilizarlo. (Una buena fuente para comprobar la existencia de ese tipo de paquetes es rpmfind.net.) Si no dispone de un RPM binario preconstruido, puede construir Dante a partir del código fuente y configurarlo como se describe en este capítulo. De manera alternativa, podría crear su propio paquete RPM de Dante creando un archivo de especificación RPM (.spec) para Dante. En este caso, si quiere más información debería ver la sección "Formato de empaquetado" en el capítulo 4.

Instalación de Dante en Slackware Linux

La filosofía de Slackware es mantener el sistema tan simple como sea posible y en casos como este es donde esa filosofía realmente merece la pena. Instalar Dante

en un sistema Slackware es una tarea casi tan sencilla como podría esperar. Los detalles proporcionados en este capítulo deberían ser aplicables directamente a un sistema Slackware.

La única excepción potencial se da si desea instalar Dante de modo que pueda ser administrado mediante las herramientas de empaquetado de Slackware, tal y como se comentó en el capítulo 5. Por fortuna, esto es tan simple como instalar Dante en un directorio separado (mediante el argumento `--prefix` en el guión `./configure`) y después utilizar las herramientas de Slackware para crear e instalar paquetes. Debería leer la sección "Formato de empaquetado" del capítulo 5 si quiere detalles completos a este respecto.

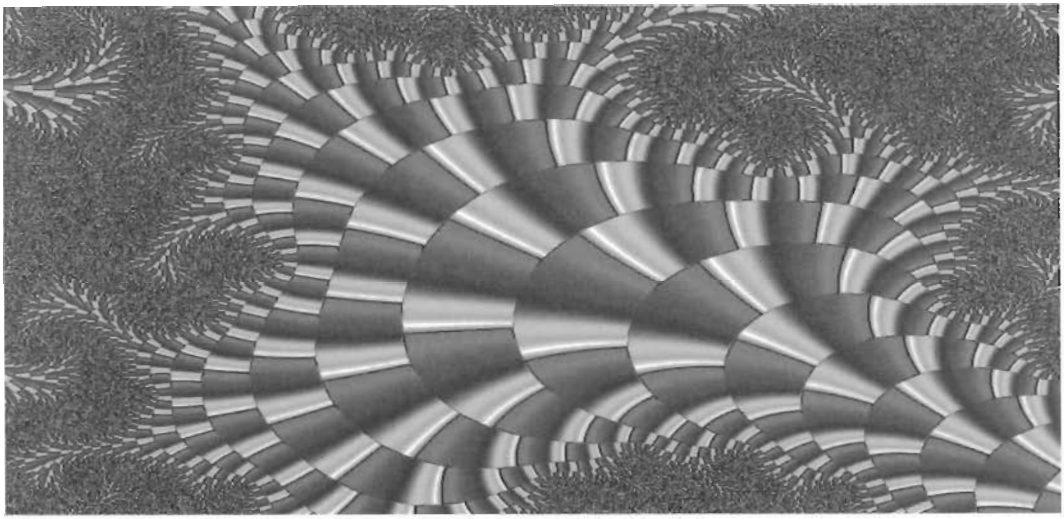
Instalación de Dante en Debian GNU/Linux

Al igual que con Red Hat Linux y Slackware Linux, instalar Dante en Debian GNU/Linux es extremadamente sencillo. Sin embargo, Debian incluye una instalación binaria preconstruida de Dante que puede utilizar, ahorrándose el esfuerzo de construir e instalar Dante por sí mismo. El nombre del paquete es dante-client y puede conseguirlo mediante el programa `apt-get` del modo habitual (como se comentó en el capítulo 6 en las sección "Uso del programa apt-get"). Si también necesita el servidor SOCKS Dante (que no se comenta en este libro), puede obtenerlo del mismo modo mediante apt-get si instala el paquete dante-server.

Resumen

En este capítulo, ha leído acerca de cómo instalar y configurar software que viene en forma de bibliotecas de cliente. Esta técnica no es tan habitual como configurar un servidor o un programa cliente tradicional, pero, aún así , se la encontrará de vez en cuando, por lo que resulta útil haberla visto. En esencia, implica instalar una biblioteca compartida y crear un archivo de configuración para ella. Esta técnica es de algún modo parecida a la utilizada por otro software de bibliotecas de cliente, como las bibliotecas centrales de los escritorios KDE y GNOME. Un ejemplo más presente de esta técnica es el omnipresente archivo /etc/resolv.conf, que es utilizado por las bibliotecas de red centrales. En cada uno de estos casos, el propio programa cliente no tiene ningún conocimiento de que el archivo de configuración (y a veces ni siquiera la biblioteca) existan.

El resto de los capítulos de esta segunda parte continuarán mostrando instalaciones adicionales de software. El siguiente capítulo comentará este proceso para el servidor HTTP Apache.



11
El servidor HTTP Apache

La Fundación de Software Apache (*Apache Software Foundation*) es una organización paraguas sin ánimo de lucro que desarrolla un número significativo de proyectos de código abierto; su producto bandera es el servidor HTTP Apache. El servidor HTTP Apache es, con razón, uno de los proyectos de código abierto con más éxito. Es ciertamente uno de los mejor conocidos; muchas personas no iniciadas frecuentemente sólo reconocen como proyectos de código abierto a Linux y Apache. Este capítulo describe cómo instalar y configurar el servidor HTTP Apache (al que nos referiremos a lo largo de este capítulo simplemente como "Apache").

Como OpenSSH, casi cualquier distribución Linux existente incluye Apache. Sin embargo, también como en el caso de OpenSSH, Apache es una parte tan importante y útil de muchos sistemas Linux (y es un ejemplo tan bueno de un estilo particular de configuración) que lo incluyo aquí como ejemplo.

El servidor web Apache comenzó su vida como una variante del servidor web httpd original del *National Center for Supercomputing Applications* (NCSA). Era una colección de parches para el software central del NCSA, y era así un servidor parcheado ("patchy", de donde viene el nombre de "Apache"). En un determinado momento el NCSA dejó de mantener su servidor, pero por entonces los desarrolladores de Apache habían reescrito de una manera efectiva todo el código original del NCSA. Apache se convirtió en un proyecto separado, y su desarrollo activo continúa aún hoy.

Los objetivos del servidor Apache son la fiabilidad, la flexibilidad y la robustez, en lugar de una gran velocidad. Como resultado, Apache es un producto increíblemente sofisticado y repleto de características. Apache puede utilizarse para una

gran variedad de necesidades; además de las tareas básicas de servicio web, Apache puede integrarse con diversos lenguajes de desarrollo web, como Perl, PHP, y Java. Apache también ofrece una extensa API que permite a los desarrolladores expandir sus capacidades de modo que pueda hacer casi cualquier cosa.

El equipo Apache no se centra en la velocidad bruta, aunque no se puede decir que Apache no sea rápido. De hecho, es bastante veloz. Sin embargo, ya que los desarrolladores de Apache no están interesados en estirar el sistema para conseguir cada gramo de rendimiento que puedan, hay algunos productos más rápidos que Apache. Aún así, pocos son más estables o disponen de más características.

El servidor HTTP Apache es una pieza de software extremadamente flexible y sofisticada. Este capítulo describe el proceso de instalación de Apache y cómo se pone en funcionamiento. Este capítulo también comenta los aspectos básicos de la particularización de Apache para que cumpla las necesidades de un sitio específico. Sin embargo, Apache es demasiado sofisticado como para entrar en mucho detalle. En lugar de hacer esto, este capítulo se centra en hacer generalizaciones sobre Apache que muestren técnicas que serán útiles también con otro software.

Nota: Sólo los hechos, servidor web Apache

Propósito: El servidor HTTP Apache es un servidor web robusto, extensible y con muchas características que puede servir páginas web estáticas y aplicaciones dinámicas escritas en una amplia variedad de lenguajes.
Autores: La Fundación de Software Apache
Sitio web: httpd.apache.org
Descripción: La Fundación de Software Apache es la organización padre de un importante número de paquetes de software. El más notable de estos es el servidor HTTP del mismo nombre Apache. Este servidor está entre los servidores web más robustos existentes, y es la plataforma escogida para más de la mitad de los sitios web de Internet. Apache proporciona la base para diseñar sitios web utilizando una gran variedad de tecnologías.

Instalación de Apache

Esta sección comenta la instalación del servidor HTTP Apache a partir del código fuente. La mayoría de las distribuciones Linux contienen una versión preconfigurada de Apache. Sin embargo, Apache es un producto muy sofisticado y complejo, y por eso sucede frecuentemente que un usuario o un administrador necesita realizar una instalación particularizada de Apache con opciones y funcionalidades no incluidas en la configuración por defecto de la distribución. En casos así, el administrador tendrá que construir un Apache personalizado a partir del código fuente, que es el tema de esta sección.

Apache es en realidad un producto muy complejo (de hecho tan complejo que se han escrito libros completos sobre él). De ningún modo esta sección puede pretender cubrir exhaustivamente todo lo referente al servidor HTTP Apache. En su lugar, esta sección tiene dos objetivos menos ambiciosos:

- Mostrar la instalación práctica de un servidor muy complicado.
- Proporcionar las bases de la configuración de Apache.

Apache es uno de los paquetes de software más sofisticados que se pueden encontrar, y por eso al completar este capítulo dará un paso importante en la comprensión de las instalaciones de software.

Según su papel y el motivo que le lleva a leer este libro, este capítulo probablemente le será útil en una de las dos maneras. Quizás sólo es un usuario medio que no tiene más que un interés muy relativo por Apache, o puede que sea un administrador de sistemas responsable de la implantación a nivel de producción en un sitio web real. En el primer caso, leer este capítulo supondrá un gran tutorial académico sobre cómo instalar una pieza compleja de software. En el segundo caso, este capítulo será una buena introducción sobre la instalación de Apache, pero probablemente no va a ser lo suficientemente exhaustivo como para que arriesgue su trabajo jugando a esta carta. Si necesita más información, debería consultar la documentación de Apache en httpd.apache.org o conseguir una copia de uno de los excelentes libros de referencia sobre Apache.

Opciones en tiempo de compilación

Incluso aunque Apache es un paquete muy complejo, es un ejemplo de instalación mediante la herramienta autoconf muy simple. La tercera parte de este libro ya ha comentado diversos ejemplos de instalación mediante autoconf, así que ya debe resultar un tema conocido. Aún así, como todas las instalaciones mediante autoconf, el peligro se esconde en las opciones en tiempo de compilación. Esta sección comenta las opciones en tiempo de compilación para el servidor HTTP Apache.

A riesgo de ser repetitivo, recuerde que Apache es un producto muy complicado. La ejecución del comando `./configure --help` muestra que tiene una lista muy larga de opciones en tiempo de compilación, y muchas de ellas están interrelacionadas e incluso son mutuamente exclusivas. En otras palabras, comprender estas opciones puede ser muy complicado.

Sin embargo, recuerde que la intención de los desarrolladores de Apache no es hacer la vida innecesariamente complicada, y por eso son realmente necesarias todas las opciones. Debido a esto, desafortunadamente, no hay modo de librarse y no hay ningún truco mágico para configurar Apache. Para instalar Apache, tiene que entender sus opciones en tiempo de compilación, y el único modo de hacerlo es consultar la documentación para el software. Por fortuna, la documentación de Apache es excepcionalmente buena y exhaustiva; el sitio web que se encuentra en httpd.apache.org contiene todo lo que se puede saber sobre la instalación (y la

ejecución) de Apache, y se incluye una copia del manual junto con el código fuente y las instalaciones.

Dicho esto, puede extraer unas pocas generalizaciones sobre los tipos de opciones que soporta Apache. El resto de esta sección comenta las siguientes opciones en tiempo de compilación:

- Directorio de instalación
- Soporte para objetos de carga dinámica (*Dynamically Loadable Objects*)
- Selección de un módulo de multiproceso (*Multi-Processing Module*, MPM)
- Otras opciones importantes

Con estas opciones, puede crear una configuración de Apache básica y razonable. Casi cualquier entorno de producción tendrá que particularizar estas opciones, pero serán un buen punto de partida. (Realmente, la configuración tal y como se comentará será muy similar a la configuración por defecto de Apache, pero merece la pena ver un poco desglosado el proceso.)

Elección de un directorio de instalación

Como ya habrá llegado a esperar, el primer paso en la instalación de un paquete de software es decidir dónde instalarlo. En el capítulo 7 se comentaron dos opciones generales: el directorio /usr/local y el directorio /opt. El directorio /usr/local es un buen lugar para colocar el software necesitado por muchos usuarios que se actualiza muy raramente, mientras que /opt es un buen lugar para poner software que necesita estar autocontenido o que necesita actualizarse frecuente o sistemáticamente.

Las otras instalaciones de software de ejemplo que se comentan en este libro incluyen muestras de las dos posibilidades; el paquete OpenSSH, por ejemplo, lo necesitan muchos usuarios y se instaló en /usr/local. El entorno Java (que se comentará más adelante) se instalará en /opt/java, ya que puede ser un paquete bastante complicado y será de ayuda tenerlo organizado en su propia ubicación.

Entonces, ¿dónde debería instalarse Apache? Bien, la respuesta a eso depende del tipo de sistema en el que se instale Apache. Si el objetivo es una máquina real a nivel de producción, entonces lo más probable es que el sistema vaya a ser vigilado muy de cerca y que se vayan a efectuar actualizaciones, como mucho, infrecuentemente; en este caso, probablemente nunca necesitará tener más de una copia instalada al mismo tiempo, y puede que sea suficientemente correcto instalar Apache en /usr/local. La mayoría de las distribuciones que incluyen Apache suelen suponer que la máquina va a utilizarse más o menos como un servidor de producción, o quizás sólo como el servidor web personal de alguien, y que las actualizaciones pueden realizarse mediante el mecanismo de paquetes (como RPM). Por eso, la mayoría de las distribuciones instalan Apache en /usr; si va a instalar una versión de Apache a partir del código fuente personalizada para un sistema de este tipo, probablemente debería usar /usr/local.

Nota

Véanse los capítulos 3 y 7 para obtener más información sobre /usr/local.

Hay otros casos de uso comunes de Apache. Por ejemplo, podría ser un desarrollador que instala Apache en su sistema de escritorio. En este caso, podría necesitar seguir los últimos y más importantes cambios de Apache o tener instaladas distintas versiones, cada una con una configuración diferente. En este caso, probablemente quiera que sus instalaciones de Apache estén más aisladas, y por eso es mejor ponerlas en un subdirectorio de /opt; por ejemplo, /opt/apache. Este es el caso que se supondrá en este libro.

Entonces, para los propósitos de este capítulo el directorio de instalación será /opt/apache. Para fijar realmente esto como opción en tiempo de compilación mediante el programa `./configure`, necesita utilizar la opción `--prefix`; a saber, `--prefix=/opt/apache`. (Si piensa que necesitará actualizar frecuentemente Apache, otra opción sería incluir el número de versión en la ruta, como mediante la opción `--prefix=/opt/apache-2.0.36` para Apache v2.0.36, y creará /opt/apache como un enlace simbólico que apunte a su versión. Esto le permite actualizar Apache mucho más fácilmente, mediante la instalación de la nueva versión por separado y el reemplazo del enlace simbólico una vez que se haya comprobado.)

Nota: Directorio de instalación frente a ubicación de los archivos de datos

Esta sección sugiere que las instalaciones "de producción" de Apache se instalen en /usr/local y las instalaciones "de desarrollo" en /opt. Esa es una buena regla general, pero hay un problema adicional: dónde instalar los archivos de datos reales, como el contenido del sitio para el que utiliza Apache. Esto podría depender en realidad de (o estar dictado por) qué lenguaje o herramientas emplea para desarrollar su sitio, pero típicamente hay al menos un archivo que especifica los contenidos de su sitio. Por defecto, Apache creará un directorio llamado htdocs que se dedica a contener estos archivos de datos, aunque puede cambiar esto fácilmente en la configuración en tiempo de ejecución. Sin embargo, este directorio htdocs estará entre los demás archivos y directorios de Apache; por ejemplo, podría acabar siendo /opt/apache/htdocs. Esto significa que sus archivos de datos estarán "mezclados" con los binarios del programa y con los archivos de configuración. No pasa nada, está bien, pero podría resultar una idea mejor colocar sus archivos de datos en una partición aparte, incluso en un disco aparte, del que se puedan obtener fácilmente copias de seguridad y que pueda restaurarse en caso de un fallo hardware en su servidor. Esto puede hacerse tras la instalación, por supuesto (véase la sección "Configuración de Apache" si desea más información). ¡Sólo recuerde planificarlo por anticipado!

Habilitación del soporte de módulos de carga dinámica

Apache tiene una gran cantidad de prestaciones, y eso significa que si todas sus características se habilitan en tiempo de compilación, el espacio necesario en memoria en tiempo de ejecución será muy grande. La mayoría de las instalaciones, mientras tanto, no necesitan toda esa funcionalidad y por eso es un desperdicio de memoria. Esta sección comenta cómo habilitar los módulos cargables que Apache soporta para resolver este asunto.

Apache utiliza objetos compartidos dinámicos (*Dynamic Shared Objects*, DSO), que también suelen indicarse como módulos Apache o simplemente módulos. Cada módulo es un archivo en el disco que contiene alguna funcionalidad específica para el servidor Apache. Por ejemplo, hay módulos DSO por separado para vigilar el estado del servidor y para realizan correcciones de ortografía.

Normalmente, si desea la funcionalidad de un módulo específico, tiene que compilar ese módulo dentro de la instalación de Apache. (Por ejemplo, tiene que compilar Apache para que incluya el módulo de comprobación de errores ortográficos si desea esa funcionalidad.) Sin embargo, si después cambia de idea, no puede librarse de ese soporte para la comprobación de errores sin tener que recompilar Apache. Siempre puede dejarlo dentro de la instalación y no utilizarlo, por supuesto, pero en ese caso es un desperdicio de memoria y puede que de tiempo de proceso de la CPU, e incluso podría ser un riesgo de seguridad si un atacante descubre un modo de explotarlo. Por estos motivos, generalmente no querrá que Apache incluya módulos que no se van a utilizar, incluso aunque se queden ahí sin hacer nada.

Apache se enfrenta a este asunto permitiendo que cada módulo se cargue dinámicamente cuando arranca el servidor. De este modo, eliminar el soporte para un módulo dado es tan sencillo como editar el archivo de configuración de Apache para eliminar la línea que carga el módulo y reiniciar el proceso del servidor Apache después. Por ejemplo, inhabilitar el módulo de comprobaciones ortográficas es tan fácil como comentar o borrar una línea en un archivo, y después el módulo de comprobación ortográfica no estará sin hacer nada, ni siquiera estará.

Otro beneficio importante de los módulos cargables es el caso inverso, es decir, añadir un módulo para incluir soporte para una funcionalidad en la que no pensó o que ni siquiera existía cuando construyó Apache. Sin el soporte DSO, por ejemplo, si quisiera disponer de la comprobación de errores ortográficos pero olvidó incluirla en la compilación, tendría que recompilar Apache para que estuviera presente. Aunque, si habilitó el soporte DSO puede añadir fácilmente este módulo más adelante y activarlo mediante el reinicio del servidor. Esta técnica es útil también para utilizar módulos de terceras compañías que no se incluyen de manera adecuada con Apache: puede compilar simplemente como módulos DSO los módulos de estas terceras compañías y cargarlos en Apache en tiempo de ejecución. La alternativa sin los módulos de carga dinámica sería provocar de algún modo que Apache compilara estos archivos de terceras personas junto con los archivos estándar de Apache. Apache realmente puede hacer esto, pero es algo bastante pesado, como verá más tarde.

Utilizar realmente módulos (compilarlos, instalarlos, activarlos en el archivo de configuración y demás tareas) se comentará más adelante en esta sección. La belleza de los DSO, después de todo, reside en que una vez que ha construido la parte central de Apache, puede añadir y eliminar más tarde módulos a placer. Por ahora, todo lo que tiene que hacer es habilitar el soporte para los módulos DSO en Apache. Esto se consigue mediante dos opciones: las opciones `--enable-mods-shared` y `--enable-so`.

La opción `--enable-so` simplemente activa el soporte para los DSO dentro de Apache; sin embargo, no provoca que ningún módulo específico se construya como DSO. La opción `--enable-mods-shared` que se pasa a `./configure` es la que le permite añadir o habilitar módulos específicos. Por ejemplo, si quisiera añadir ese módulo de comprobaciones ortográficas como un DSO, usaría `--enable-mods-shared=mod_speling`. Sin embargo, también hay una forma conveniente que puede utilizar (específicamente, `--enable-mods-shared=all`) que le ahorra el trabajo de tener que introducir todos los módulos individuales explícitamente. Esta es la opción que se utilizará en este libro, y su resultado es que se habilitan todos los módulos que pueden habilitarse como módulos DSO se habilitan.

Nota: Módulos DSO y seguridad

Puede argumentarse que habilitar módulos DSO con Apache supone un riesgo de seguridad, y que por ello no debería utilizarse en sistemas de producción reales. Es concebible que un atacante pueda provocar que Apache cargue dinámicamente un módulo DSO designado para comprometer el servidor Apache o el propio sistema. Si el soporte DSO está inhabilitado, esto sería obviamente imposible.

Esto es un riesgo. Sin embargo, en la práctica el riesgo es probablemente bastante remoto, y de cualquier modo es muy irrelevante. Después de todo, si un atacante puede obtener acceso al sistema de archivos en que está instalado Apache (para colocar su propio módulo hostil para que lo cargue Apache), entonces ciertamente puede hacer mucho más daño que solamente cargar un módulo en Apache.

Por eso, en cuanto riesgo de seguridad, el de habilitar los módulos DSO para Apache es bastante pequeño. Sin embargo, es un riesgo, y los usuarios y administradores siempre necesitan comparar los riesgos con sus impactos potenciales. ¡Estar avisado de antemano significa estar preparado de antemano!

Elección de un MPM

Algunos módulos Apache no pueden cargarse como módulos DSO. Entre estos módulos se incluye el módulo central, que contiene cosas como la capacidad de cargar realmente los módulos DSO, y el módulo de multiproceso (*Multi-Processing*

Module, MPM) que se usará para la instalación. Esta sección comentará las opciones de MPM disponibles para Apache.

Un MPM es un módulo que maneja las conexiones con los clientes y gobierna el modo en que Apache se comporta cuando completa las peticiones de esos clientes. Cada MPM puede utilizar una técnica diferente para este "multiproceso" y cada uno tiene un perfil de rendimiento distinto. Al permitir que se particularice el MPM, Apache puede utilizar una técnica diferente para cada sistema operativo en el que se ejecute, permitiendo obtener el mejor rendimiento posible de cada plataforma. La funcionalidad MPM se añadió en la versión 2.0 de Apache.

Ya que el MPM que se utilice son los cimientos para el resto de Apache, el MPM no puede construirse como un módulo de carga dinámica; tiene que compilarse junto con el programa central de Apache. En otras palabras, una vez que se ha compilado e instalado Apache, para cambiar a un MPM distinto será necesario recompilar el propio Apache.

En general, el programa `./configure` puede detectar sobre qué sistema operativo se va a compilar Apache y seleccionará el MPM más adecuado para esa plataforma. Sin embargo, algunas plataformas soportan varios MPM de entre los que puede escoger. En estos casos, el programa `./configure` selecciona un MPM por defecto razonable, que suele ser el más estable.

Conmutar a un MPM distinto no es un asunto pequeño, ya que tiene implicaciones importantes sobre el rendimiento. Si quiere utilizar un MPM alternativo, debería consultar la documentación de Apache sobre este tema. Esta sección sólo está aquí para avisarle de la presencia de los módulos MPM en Apache; en la configuración real utilizada en este ejemplo simplemente se aceptó el MPM por defecto que Apache seleccionó para el sistema. No se pasó ninguna opción particular relacionada con MPM a `./configure`.

Otras opciones

Como ya he mencionado en repetidas ocasiones, Apache soporta una muy amplia variedad de opciones en tiempo de compilación. Cubrir exhaustivamente estas opciones queda más allá de la intención de esta sección y, además, explorar las características que ofrece Apache es de por sí una cuestión de varias horas. Si verdaderamente quisiera dominar Apache, debería consultar la documentación de Apache u otro libro. Sin embargo, hay unas pocas opciones bastante comunes que los usuarios necesitarán frecuentemente, algunas de las cuales se resumen en la tabla 11.1.

Tabla 11.1. Algunas opciones comunes en tiempo de compilación de Apache.

Opción	Descripción
`--enable-ssl`	Activa el soporte al cifrado SSL/TLS de conexión.
`--enable-proxy`	Activa el soporte para que Apache actúe como un servidor proxy HTTP.

(Continuación)

Opción	Descripción
`--enable-dav`	Activa el soporte para el estándar de administración de contenidos WebDAV.
`--enable-speling`	Incluye la corrección de errores ortográficos de direcciones URL.

Completar la instalación

Ahora que ya ha seleccionado las opciones en tiempo de compilación, puede instalar Apache. Este es un ejemplo muy simple de una instalación mediante autoconf de GNU, queriendo decir con esto que consiste esencialmente en tres comandos: `./configure` con los argumentos apropiados, `make`, y `make install`. Aquí está el comando `./configure` que se usó realmente en este ejemplo:

```
./configure --prefix=/opt/apache --enable-so --enable-mods-shared=all
```

Una vez que el comando `./configure` y los comandos `make` se hayan completado, Apache estará instalado en el directorio /opt/apache.

Si ha leído los otros ejemplos de instalación de software de este libro, debería de comenzar a ver algunos patrones. Tras la lectura de este capítulo sobre Apache, habrá visto la instalación práctica de un servidor software bastante complicado. Será capaz de aplicar muchas de las mismas técnicas y formas de pensar en la instalación de cualquier otro programa servidor, como el servidor de directorios OpenLDAP o el servidor de compartición de archivos y medios de impresión Samba. Instalar estas cosas es bastante fácil (sólo tiene que leer la documentación para descubrir qué opciones necesita, después ejecute `./configure`, `make`, y `make install`, y ¡ya está! ¡Ya tiene configurado su software!) Por supuesto, incluso en esa situación, el trabajo sólo está a medio hacer, también hay que configurarlo. Ese es el tema de nuestra siguiente sección.

Configuración de Apache

Esta sección comentará la configuración de Apache una vez instalado. La sección anterior comenta la configuración de las opciones en tiempo de compilación de Apache, lo que realmente se reduce a habilitar o inhabilitar características específicas. Sin embargo, las opciones en tiempo de ejecución controlan el comportamiento real de Apache cuando se está ejecutando.

Apache utiliza una configuración de servidor típica; es decir, lee un archivo de configuración que fija una variedad de opciones y modifica su comportamiento de acuerdo con ellas. No obstante, Apache es una pieza de software bastante complicada, así que viene con un buen montón de archivos. Esta sección resume en primer

lugar los diversos archivos y directorios relevantes a Apache y después comenta cómo utilizar estos archivos para modificar la configuración.

Nota

Si desea más información acerca de las opciones en tiempo de compilación frente a las opciones en tiempo de ejecución, vea la sección "Opciones de configuración del software" en el capítulo 7.

Exploración del directorio de instalación

En la sección anterior se instaló Apache en el directorio /opt/apache. Un rápido vistazo a ese directorio revelará un cierto número de subdirectorios. Esta sección describe estos directorios, centrándose en los más importantes. Después de esta "visita guiada" de una instalación Apache, la siguiente sección comenta cómo configurar el servidor. Para comenzar, la tabla 11.2 muestra una lista de los directorios de configuración estándar de Apache.

Tabla 11.2. Subdirectorios de la instalación de Apache.

Directorio	Contenidos
bin	Archivos binarios del programa para el servidor y otras utilidades.
build	Guiones de utilidad y otros archivos para construir y añadir módulos DSO adicionales.
cgi-bin	Guiones de web utilizados para generar contenido web dinámicamente.
conf	Archivos de configuración para el servidor.
error	Archivos con mensajes de texto que el servidor muestra cuando se producen diversos errores.
htdocs	Archivos de contenido ofrecidos por el servidor HTTP.
icons	Pequeños icono gráficos utilizados por el servidor cuando presenta diversas páginas generadas dinámicamente (como listados de directorios).
include	Archivos necesarios para compilar módulos DSO y otros programas.
lib	Bibliotecas compartidas (no módulos DSO) necesitados para el servidor HTTP.

(Continuación)

Directorio	Contenidos
logs	Archivos que contienen información de anotaciones sobre diversos eventos HTTP.
man	Páginas de manual estándar de Unix para el servidor y los programas de utilidades.
manual	El manual completo de instalación y uso de Apache en formato HTML.
modules	Archivos de módulos de carga dinámica (DSO).

Algunos de los directorios que aparecen en la tabla 11.2 se comentarán de un modo individual en las siguiente sección. Otros que son menos importantes se comentarán al unísono al final de esta sección.

Programas en el directorio bin

Como probablemente esperará tras la lectura del capítulo 3, el subdirectorio bin en el nivel superior del directorio de instalación de Apache contiene los programas binarios. Esto incluye tanto el programa del servidor HTTP Apache real (httpd) como varios programas de gran utilidad. La tabla 11.3 resume estos programas.

Tabla 11.3. Programas en el directorio bin de Apache.

Programa	Propósito
httpd	El servidor HTTP Apache.
ab	Una utilidad de desarrollador para medir el rendimiento de un servidor HTTP (como Apache).
apachectl	Proporciona un modo conveniente de arrancar, detener, y administrar el servidor HTTP.
apxs	Una utilidad de desarrollador y administrador utilizada para compilar e instalar nuevos módulos DSO en una instalación de Apache ya existente.
checkgid	Utilizado internamente por los programas de Apache.
dbmmanage, htdigest, htpasswd	Programas utilizados para administrar las cuentas de usuario y contraseñas de Apache en diversos formatos.
logresolve	Una utilidad conveniente para convertir direcciones IP numéricas en los archivos de anotaciones de Apache en nombres de máquinas legibles.
rotatelogs	"Rota" los archivos de anotaciones renombrando los archivos de anotaciones actuales y creando nuevos archivos de anotaciones.

Programas del servidor

Los programas hhtpd, ab, apachectl y checkgid están todos ellos relacionados con la administración del servidor y las pruebas de rendimiento. El programa httpd es, por supuesto, el propio servidor HTTP Apache; es el plato fuerte, por así decirlo. El programa apachectl es realmente un guión de shell, y se utiliza para arrancar, detener, reiniciar, y administrar de cualquier modo el proceso httpd. El programa apachectl es parecido a los guiones de servicio utilizados por el modelo de inicio SysV en Red Hat Linux, que se comentaron en los capítulos 3 y 4.

Los programas checkgid y ab son menos importantes. El programa checkgid es simplemente una pequeña utilidad invocada por Apache; los usuarios y administradores no necesitan preocuparse por de él (¡ni siquiera tiene una página man!). El programa ab es el programa "*Apache Bench*" (banco de pruebas Apache), y se utiliza para medir el rendimiento de un servidor HTTP. (Incluso aunque ab se incluye junto con Apache, puede funcionar realmente junto con muchos servidores HTTP.) El programa ab lo utilizan los administradores y desarrolladores para probar configuraciones de ajuste del rendimiento y el rendimiento de aplicaciones web. En general, la mayoría de los usuarios no necesitan preocuparse de ab a no ser que haya un problema con la instalación.

Programas de administración

Además de los programas del servidor y los programas de apoyo, más importantes, también se incluyen unas cuantas utilidades de conveniencia para hacer más sencilla la vida de los usuarios. El programa apxs es la utilidad de "Extensión de Apache" (*Apache Extension*). Este programa se utiliza para compilar e instalar módulos DSO de Apache. (Recuerde de la sección anterior que los módulos DSO son archivos que Apache puede cargar cuando arranca para incluir determinadas características en el servidor.) Los módulos DSO proporcionan un modo de añadir prestaciones a Apache sin tener que recompilar el servidor al completo. Si descubre que necesita instalar un módulo DSO de Apache nuevo, apxs resultará muy útil. Es bastante fácil de usar, pero si tiene algún problema, consulte la documentación de Apache sobre él.

El programa logresolve se utiliza para convertir direcciones IP numéricas en nombres de máquinas dentro de los archivos de anotaciones de Apache. Un típico servidor Apache recibe una gran cantidad de peticiones desde navegadores web a través de Internet. Cada petición contiene información sobre el cliente, incluida su dirección IP. Apache anota estas peticiones (al igual que otras cosas, como los errores), pero las entradas anotadas sólo incluyen la dirección IP. Es así porque el proceso de buscar el nombre de la máquina de un cliente en servidores de nombre de dominio (*Domain Nameservers*, DNS) puede resultar muy costoso y desperdiciar innecesariamente ancho de banda. El programa logresolve se incluye como un modo sencillo de tomar los archivos de anotaciones de Apache (que sólo contienen direcciones IP) y volverles a dar formato con nombres de máquinas obtenidos desde el servicio DNS. Puede utilizar logresolve en cualquier momento en que necesite o quiera ver las anotaciones de su sitio con algo más que sólo direcciones IP.

El programa rotatelogs administra los archivos de anotación de Apache. A medida que el servidor HTTP funciona durante mucho tiempo, va anotando diversos eventos (en especial peticiones procedentes de navegadores web) en sus archivos de anotaciones. Estos archivos pueden volverse muy grandes como resultado de esta anotación continua. El comando `rotatelogs` "rota" los archivos de anotaciones de Apache; en esencia, sólo renombra los archivos de anotaciones que se usan actualmente y después hace que Apache comience a tomar sus anotaciones en ficheros nuevos. El usuario puede entonces borrar los antiguos archivos de anotaciones después de revisarlos, para ahorrar espacio en el disco.

Nota

El comando `rotatelogs` no se suele usar realmente desde la línea de comandos; en su lugar, se incrusta en el mecanismo de anotaciones de Apache mediante los archivos de configuración. Si desea más información al respecto, vea la página man acerca de rotatelogs que se incluye junto con la instalación de Apache.

Programas de administración de la base de datos de usuario

Una característica habitualmente necesitada por los sitios web es la restricción de acceso a ciertas páginas o conjuntos de páginas para usuarios específicos. (Estas páginas y conjuntos de páginas se conocen como "reinos" [*realms*].) Apache soporta esta funcionalidad mediante diversas directrices en sus archivos de configuración. Sin embargo, otro componente clave en el soporte de la autenticación de usuario es, por supuesto, tener una base de datos de usuarios.

Hay un cierto número de formas de preparar una base de datos para usuarios. Los sitios web comerciales, con los que la mayoría de los lectores se sentirán familiarizados, utilizan algún tipo de base de datos relacional (o un directorio) de alta capacidad para almacenar información sobre las cuentas de usuario. No obstante, la autenticación que Apache soporta por defecto es más modesta. Soporta bases de datos de usuarios almacenadas en discos locales, en unos pocos formatos diferentes. (Hay, por supuesto, módulos preparados por terceras personas que puede utilizar para proporcionar a Apache un almacenamiento de datos de alta capacidad; sólo que no se incluyen con la distribución estándar de Apache.)

Los tres programas dbmmanage, htdigest, y htpasswd son utilidades que gestionan estas bases de datos de usuario. Los matices acerca de qué formato de datos es mejor para una determinada aplicación, y cuándo puede utilizar el mecanismo básico de Apache o cuándo cambiar a una base de datos de alta capacidad son temas avanzados y no son realmente relevantes para este libro. Si necesita más información, por favor consulte las páginas del manual para esos programas (que se incluyen con la instalación de Apache) o la documentación que puede encontrarse en el sitio web de Apache en httpd.apache.org.

Archivos de configuración en el directorio conf

El directorio conf contiene todos los archivos de configuración para Apache. (En realidad, al programa httpd, que es el auténtico programa del servidor HTTP Apache, se le puede indicar qué archivo de configuración debería usar en forma de parámetro en línea de comandos. El directorio de configuración de Apache no está realmente incrustado en el código fuente ni se fuerza a que tenga un valor específico.) Esta sección comenta brevemente los contenidos de este directorio.

El nombre canónico para el archivo de configuración de Apache es httpd.conf. (Una vez más, sin embargo, no tiene por qué ser así; al programa httpd puede pasársele un nombre de archivo arbitrario.) Una instalación típica de Apache tendrá varios archivos httpd.conf de ejemplo, como una configuración ajustada para conseguir el máximo rendimiento y otra que habilite el soporte de SSL, al igual que un archivo httpd.conf estándar.

Como resultado, normalmente habrá varios archivos con la extensión .conf dentro del subdirectorio conf de la instalación de Apache; sólo se necesita que esté ahí uno de estos archivos. Además, por cada archivo .conf habrá un archivo con la extensión -std.conf. Los archivos -std son copias de los archivos originales; su propósito es actuar como copias de seguridad en caso de que un administrador con tendencia a cacharrear llene de basura un archivo .conf. Sin embargo, los archivos -std tampoco son necesarios para que funcione Apache.

En realidad puede eliminar todos los archivos .conf y -std.conf excepto httpd.conf si no desea que abarroten del directorio de configuración. Eso dejará únicamente el archivo de configuración de Apache tradicional como punto de partida para comenzar a personalizar el servidor. Sin embargo, merece la pena echar un vistazo a esos otros archivos de ejemplo antes de borrarlos si es que escoge este camino; hay una gran cantidad de configuraciones de ejemplo e información muy valiosa en esos archivos. Por supuesto, aún así, no hay ningún sustituto para la lectura de la documentación real. Los archivos de configuración de muestra son simplemente eso, y no son exhaustivos.

Hay otros dos archivos en el subdirectorio conf: mime.types y magic. MIME es la extensión multimedia de correo de Internet (*Multimedia Internet Mail Extension*), y es un estándar de la industria para la especificación del tipo de un archivo, como por ejemplo "ejecutable" o "texto sencillo" o "HTML". (¡Que no le confunda el "*Mail*" en el nombre de MIME, el estándar MIME pude usarse en casi cualquier parte!) El archivo mime.types de Apache es sólo una correspondencia que asigna tipos MIME a extensiones de archivo. (Por ejemplo, la extensión ".txt" se hace corresponder con el tipo MIME "text/plain" [texto sencillo].) Normalmente, no tendrá que modificar este archivo, ya que contiene valores por defecto bastante razonables. Si lo hace, lea cuidadosamente la documentación de Apache al respecto, ya que modificar mime.types puede tener sutiles consecuencias. El archivo magic funciona de un modo parecido a mime.types, excepto en que se utiliza a para asignar tipos MIME a archivos cuya extensión puede resultar poco fidedigna, basándose en su lugar en "números mágicos" contenidos en el archivo. Esto se utiliza frecuentemente para cosas tales como los archivos de audio.

Archivos de anotaciones en el directorio logs

El subdirectorio logs contiene, de manera suficientemente intuitiva, los archivos de anotaciones para el servidor HTTP Apache. El mecanismo de anotaciones de Apache es muy flexible y puede configurarse para anotar una gran variedad de eventos en cualquier número de archivos con cualquier formato. Rara vez será necesario modificar esta configuración y por eso no comentaré este proceso aquí. Por fortuna, Apache tiene una configuración de anotaciones por defecto que casi siempre será adecuada. La tabla 11.4 resume los archivos involucrados en esta configuración por defecto.

Tabla 11.4. Archivos de anotación por defecto de Apache.

Nombre de archivo	Contenidos
error_log	Errores relacionados con el funcionamiento del servidor o con peticiones fallidas o erróneas provenientes de navegadores web clientes; se utiliza para depurar errores.
accesss_log	Información sobre cada petición recibida desde un cliente; se utiliza para recoger estadísticas sobre el sitio web y la utilización de los usuarios.
httpd.pid	Contiene el identificador de proceso Unix del proceso httpd que se ejecuta en ese momento.

Técnicamente, el archivo httpd.pid no es realmente un archivo de anotaciones; sólo contiene el identificador de proceso (PID) de Apache. Este archivo facilita descubrir rápidamente el PID de Apache para poder reiniciarlo o detenerlo. Además, dependiendo de la plataforma sobre la que funcione Apache y de las opciones que se hayan habilitado, puede haber más o menos archivos en el directorio logs que los mostrados en la tabla 11.4.

Directorios de contenidos

El propósito del servidor HTTP Apache es, por supuesto, servir documentos como páginas web escritas en HTML. Apache también soporta lenguajes de programación muy diferentes, y entornos que los desarrolladores web puede utilizar para crear y ejecutar aplicaciones web dinámicas e interactivas. La instalación de Apache utiliza los directorios htdocs y cgi-bin para contener los archivos que realmente conforman del contenido del sitio. El directorio htdocs es el hogar de los archivos HTML "estáticos". Estos son archivos que no tienen ningún contenido generado dinámicamente, como tablas de contenidos, páginas de entrada por defecto al sitio web, archivos HTML con cabeceras y pies de página estándar y demás. El directorio htdocs también contiene otros archivos estáticos necesarios para el sitio, como archivos con imágenes gráficas, archivos *tarball* (.tar.gz) que se descargarán desde

el sitio, y muchos más. Literalmente, cualquier archivo que se encuentre en el directorio htdocs (y al que el proceso httpd tenga permiso de acceso) se vuelve disponible para su descarga desde el sitio web. En otras palabras, ¡no ponga en htdocs nada que no quiera que la gente pueda descargar!

El directorio cgi-bin contiene guiones y programas que utilizan la interfaz de pasarela común (*Common Gateway Interface*, CGI) estándar para las aplicaciones web dinámicas. CGI define una manera estándar y fija en la que un servidor web (como Apache) interactúa con los programas que se utilizan para generar el contenido. Los desarrolladores web escriben programas en un lenguaje de programación como Perl, Python, C, PHP, u algún otro lenguaje que entienda el estándar CGI, y esto les permite trabajar junto a un servidor web que soporte CGI, como Apache. El directorio cgi-bin de Apache contiene todos los programas CGI utilizados en el sitio web. El directorio está separado del directorio htdocs por cuestiones de seguridad y para hacer que el sistema se administre más fácilmente.

Dese cuenta de que los directorios htdocs y cgi-bin son para el modo "estándar" de uso de Apache. Apache es extremadamente flexible y soporta una amplia variedad de maneras de servir y generar dinámicamente contenido. Por ejemplo, Apache puede utilizarse con aplicaciones web que utilicen la tecnología de Servlets Java de Sun, que define un modo muy distinto de almacenar y servir los datos de la aplicación web. Los sitios basados en Servlets Java frecuentemente no usan ni el directorio htdocs ni cgi-bin.

Directorios de documentación

En una instalación estándar de Apache hay dos directorios con nombres parecidos: manual y man. ¡No, no es un error tipográfico! Estos directorios contienen documentación sobre apache y sobre las utilidades de apoyo relacionadas.

El directorio manual contiene una copia del manual de usuario de Apache, que está en formato HTML. Puede utilizar cualquier navegador web para explorar este manual, que es una copia de la documentación que se encuentra en el sitio web de Apache. Estos archivos no son necesarios para que Apache funcione, por eso si dispone de poco espacio en el disco duro puede borrarlos sin ningún problema (¡aunque resulta útil tenerlos a mano!). Dependiendo de la configuración de su instalación de Apache también puede que necesite eliminar la referencia que se hace al directorio manual en el archivo httpd.conf; compruebe ese archivo en busca de más detalles. En cualquier caso, puede acceder al manual desde su servidor accediendo al URL http://localhost/manual con su navegador web favorito.

El directorio man contiene, en cambio, las páginas del manual de Unix para los archivos binarios del programa que se encuentran en el directorio bin. Aquí es donde se encuentran las páginas del manual sobre httpd, apachectl y demás. Sin embargo, si ha instalado Apache en un directorio como /usr/local u /opt/apache puede que no sea capaz de acceder a estas páginas debido a que el propio programa `man` no sabe que debe buscar en estos directorios para dar con las páginas. Para poder acceder a estas páginas del manual, utilice un comando parecido a este:

```
man -M /opt/apache/man <nombre_de_comando>
```

Tendrá que sustituir `/opt/apache` por el directorio de su instalación de Apache.

Nota

En general, la opción `-M` del programa man suele ser una cosa muy útil que conviene conocer. Esta opción le permite especificar un directorio alternativo para que se busquen en él las páginas del manual. Resultará muy práctico cuando instale software en una ubicación que no sea /usr.

Nota: Uso de la variable MANPATH

El uso del parámetro `-M` en el comando `man` puede volverse agobiante si consulta frecuentemente páginas man. Por fortuna, el programa man soporta una variable de entorno llamada MANPATH. Si fija el valor de esta variable a una lista de rutas que contengan páginas man, separadas por dos puntos (como `/opt/apache/man:/usr/man`), puede añadir permanentemente directorios para que man busque en ellos cuando necesite encontrar páginas del manual. Al añadir este parámetro en los guiones globales de configuración de usuario descritos en los capítulos 4, 5 y 6 (para Red Hat Linux, Slackware Linux, y Debian GNU/Linux, respectivamente), puede volver accesibles a los usuarios las páginas de manual instaladas en ubicaciones que no sean la estándar.

Otros directorios

En una instalación Apache se incluyen también otros directorios (build, error, icons, include, lib y modules). En la mayor parte de los casos, estos directorios los utilizan Apache y las herramientas relacionadas, y los usuarios y administradores no necesitan hacer nada en o con estos directorios. Si quisiera más información sobre ellos, puede consultar el sitio web de Apache y otra documentación diversa.

Personalización de Apache

Hasta ahora, he comentado sólo la instalación de Apache y algunas tareas de configuración muy básicas necesarias para conseguir que funcione. Sin embargo, el propósito de Apache es hacer que funcione un sitio web, y por eso el siguiente paso es comentar cómo personalizar Apache para que ejecute su propio sitio web.

Resulta imprescindible recordar que Apache es un producto muy sofisticado, y que es capaz de mostrar un rango sorprendentemente amplio de comportamientos. El objetivo de esta sección no es describir exhaustivamente cómo personalizar Apa-

che para cada circunstancia posible. En su lugar, el objetivo es mostrar los mecanismos de personalización de Apache a la vez que ofrecer una metodología en particular para configurar un servidor. Después de que haya leído esta sección, no sólo dispondrá de una comprensión básica sobre cómo configurar Apache, sino que también entenderá cómo actuar para adquirir una comprensión similar de las configuraciones de otros servidores.

Quizás la particularización más habitual que necesitará realizar en un servidor HTTP Apache sea configurar directorios específicos. En general, un determinado sitio web tiene muchos directorios y a veces cada directorio requiere comportamientos o características específicas. Por ejemplo, la mayoría de los directorios podrían ser accesibles públicamente, pero un directorio necesita "bloquearse" en cuanto a seguridad se refiere y requerir una contraseña. Para soportar esta funcionalidad, Apache le permite especificar propiedades o habilitar características para directorios individuales mediante el uso de directrices dentro de bloques "*Directory*" (directorio) en el archivo httpd.conf.

Una directriz es, en el lenguaje de Apache, una instrucción en el archivo httpd.conf que indica al servidor que realice una acción determinada. Por ejemplo, una línea que habilita el soporte de cifrado SSL/TLS sería una directriz, al igual que una línea que desactivase la habilidad de generar listados de los contenidos de un directorio. Las directrices Apache, por su parte, están encerradas en diversos "bloques" que determinan el alcance de su efecto.

Hay muchos tipos de estos bloques, pero es más simple es el bloque *Directory*. Un bloque de directorio tiene un comienzo definido (marcado por una línea `<Directory ...>`) y un fin definido (marcado por una línea `</Directory>`). Y las directrices contenidas dentro del bloque sólo afectan a ese bloque (y a cualquier posible hijo del bloque).

Un archivo de configuración httpd.conf típico contiene una entrada para el directorio de mayor nivel del sitio. Esta entrada contiene algunas directrices que se aplican al directorio raíz. Ya que también se aplican a cualquier hijo del directorio raíz, de este modo se fijan efectivamente las directrices por defecto para todo el sitio. El texto siguiente es un ejemplo de bloque *Directory* que podría utilizarse para la instalación de Apache comentada en este capítulo:

```
<Directory "/opt/apache/htdocs">
   Options FollowSymLinks
   AllowOverride None
   Order allow, deny
   Allow from all
   DirectoryIndex index.html index.htm
<Directory>
```

La ruta /opt/apache/htdocs en el bloque de directorio anterior apunta hacia la ubicación de la instalación en la que se encuentra el contenido HTML, y las directrices englobadas en el bloque se aplicarán a ese directorio. Además, ya que todo el contenido tiene que estar dentro de ese directorio, este bloque también sirve efectivamente como bloque por defecto para todo el sitio. (Describir lo que hace cada una de esas directrices tomaría demasiado tiempo y no es realmente relevante, así que

si quiere más detalles, consulte la documentación de Apache mencionada con anterioridad.)

Sin embargo, es posible anular estas directrices por defecto según un criterio caso por caso. Por ejemplo, quizás desea añadir archivos llamados home.html a la directriz `DirectoryIndex`, pero sólo para el directorio /opt/apache/htdocs/home. El bloque de directorio siguiente llevaría a cabo esta labor: la directriz `DirectoryIndex` se actualizará para ese directorio específico, pero todas las otras directrices permanecerán inalteradas.

```
<Directory "/opt/apache/htdocs/home">
    DirectoryIndex index.html index.htm home.html
<Directory>
```

La mayoría de las particularizaciones de un sitio gestionado por Apache se basan en la creación de estos bloques *Directory* (y otros bloques) del modo apropiado y la posterior utilización de las directrices correctas dentro de cada bloque. Este modelo es realmente muy potente, ya que le permite particularizar el comportamiento de cada área individual del sitio. Lamentablemente, este es el máximo detalle en que puedo entrar, ya que no podría hacer justicia a la auténtica potencia del sistema de configuración de Apache en el espacio que tengo disponible. Mi consejo es que lea el manual de Apache citado anteriormente para tener detalles completos.

Dejando a un lado toda esta terminología sobre directorios y directrices, a lo que todo se reduce es a que Apache le proporciona un conjunto de opciones (directrices) y un mecanismo (bloques de directorio) para extender o reducir el alcance de las opciones a distintas partes del sistema. Este es un modo muy habitual para la configuración de software de servidor: crear bloques para un cierto ámbito de algún modo y después fijar opciones dentro de cada bloque de ámbito.

En el mundo Apache, estos bloques de ámbito se crean mediante parejas de líneas `<Directory>` y `</Directory>`; en otros programas podrían fijarse mediante pares de llaves ({ y }) o simplemente mediante la posición de las líneas en el archivo (como en el caso del servidor OpenSSH). Incluso aunque estos programas sean diferentes en su sintaxis, en cada caso se trata de la misma idea general. Cuando se encuentre intentando configurar un programa nuevo con el que no se haya familiarizado, pruebe e identifique los casos en que establezca "ámbitos" y cree "opciones" dentro de ellos.

Instalación de Apache en diferentes distribuciones

Esta sección comenta las cuestiones que surgen cuando se instala el servidor HTTP Apache en alguna de las tres distribuciones de muestra. En general, los procesos de instalación y arranque de Apache son bastante autocontenidos (que quiere decir que no confían demasiado en que otros programas se arranquen antes),

por eso no hay una gran cantidad de cambios entre las distribuciones. Sin embargo, hay algunas cosas que debería comprender.

Instalación de Apache en Red Hat Linux

Instalar Apache en Red Hat Linux es muy sencillo si se utilizan las técnicas comentadas en este capítulo. No obstante, hay un par de cosas de las que debe ser consciente relativas a la versión de Apache y a cómo administrar el software una vez que esté instalado.

Elección de la versión correcta

Red Hat Linux incluye un paquete con el servidor HTTP Apache. Sin embargo, en Red Hat Linux 7.3 este paquete es de una versión más antigua de Apache que la comentada a lo largo de esta sección. En general, configurar la versión más antigua de Apache es en esencia lo mismo (en el sentido de que necesita recurrir a muchas de las mismas técnicas), pero hay algunas diferencias. Si escoge utilizar la distribución estándar de Apache que se incluye con Red Hat, asegúrese de consultar y comprender la documentación del proyecto Apache para la versión más antigua que va a instalar.

Inicio del servidor Apache durante el arranque del sistema

Una necesidad habitual es arrancar el servidor Apache automáticamente cuando arranque el sistema. Como se comentó en el capítulo 4, esto se consigue en la distribución Red Hat Linux mediante el modelo SysV de guiones de inicio del sistema. Este modelo se basa en un guión "maestro" que invoca un cierto número de guiones específicos para cada servicio, cada uno de los cuales se adhiere a una determinada estructura. Estos guiones son a su vez responsables del inicio, detención, y reinicio de un servicio en particular.

Nota

Véase el capítulo 4 para más información sobre el modelo SysV.

El servidor HTTP Apache incluye un programa en forma de guión que automatiza el proceso de arranque y parada del servidor (es el programa apachectl mencionado anteriormente). Este guión de shell puede recibir los comandos `stop`, `start` y `restart`. Son los mismos comandos que los utilizados por los guiones compatibles con el sistema SysV, lo que significa que el programa apachectl puede utilizarse para gestionar el servidor HTTP Apache en un sistema Red Hat Linux.

Para fijar este modo de empleo, sólo necesita crear un enlace simbólico. El siguiente comando muestra cómo se haría esto si ha instalado y configurado el servidor del modo comentado a lo largo de este capítulo:

```
# ln -s /opt/apache/bin/apachectl /etc/rc.d/init.d/apache
```

Este comando configurará Apache de modo que el sistema pueda iniciarlo y detenerlo (es decir, que se pueda utilizar el comando `service`, tal y como se comentó en el capítulo 4). Sin embargo, el capítulo 4 también describe el modo especial de nombrar y ordenar los enlaces simbólicos que determina realmente si un servicio inicia cuando lo hace el sistema y en qué orden lo hace. Simplemente crear el enlace simbólico anterior no provocará que Apache inicie con el sistema.

Nota

Véase el capítulo 4 para más información sobre el uso de los enlaces simbólicos en el modelo de guiones de inicio SysV.

Hay dos opciones para hacer que Apache inicie durante el arranque del sistema. Uno es gestionar los enlaces simbólicos necesarios a mano (como se comentó en el capítulo 6) y el otro hacer que el guión apachectl sea compatible con el comando `chkconfig` de Red Hat, que se podrá utilizar entonces para administrar el guión.

Hacer apachectl compatible con chkconfig

Para integrar adecuadamente Apache con las herramientas que Red Hat ofrece para administrar los servicios del sistema, el guión apachectl necesita integrarse con el comando `service` de Red Hat (lo que se comentó en la sección anterior) al igual que con el comando `chkconfig`. Ambos comandos se comentan en detalle en el capítulo 4, y debe leer ese capítulo para comprender todos los detalles del material comentado en esta sección.

Hacer que un guión sea compatible con chkconfig significa añadir un comentario en el guión que contenga la información que utiliza chkconfig para configurar el servicio. Las siguientes líneas pueden añadirse al guión apachectl (contienen esa información necesaria):

```
# chkconfig: 345 99 99
# description: El servidor HTTP Apache
```

Esas líneas provocarán que apachectl quede configurado (por defecto) para ejecutarse en los niveles de ejecución 3 a 5, con una prioridad de 99 tanto para el arranque como para su parada (ya que Apache no depende de si otros servicios se están ejecutando). Una vez que haya hecho este cambio en el guión apachectl, puede activarlo ejecutando el comando `chkconfig --add apache`; después de eso, podrá gestionar normalmente el estado del arranque de Apache, empleando para ello las herramientas y técnicas descritas en la sección "Guiones de arranque del sistema" del capítulo 4.

A algunos administradores puede no gustarles la idea de modificar un guión proporcionado por Apache, por una buena razón: si se actualiza Apache, también habrá que realizar este cambio en el guión apachectl actualizado. Si esto supone una inquietud, puede merecer la pena escribir un guión de shell muy simple (simi-

lar al listado 8.1 del capítulo 8) que contiene la información de chkconfig y simplemente llama al guión apachectl. De este modo, el comando `service` y otras herramientas del sistema utilizarán el guión particularizado, y el guión apachectl no tendrá que ser modificado. Algunos administradores pueden preferir esta opción.

Instalación de Apache en Slackware Linux

Al igual que con Red Hat Linux, es fácil poner en marcha Apache en la distribución Slackware Linux. Slackware Linux 8.0 también contiene un paquete para Apache, pero una vez más, es una versión más antigua que la comentada en este capítulo, como en Red Hat. Configurar Apache para que lo arranque Slackware Linux es muy simple. Como se comentó en el capítulo 5, Slackware Linux utiliza el modelo BSD de guiones de arranque de sistema, pero también ofrece un "enganche" compatible con SysV mínimo por motivos de conveniencia. Configurar Apache para que inicie cuando lo haga Slackware Linux es tan simple como crear un enlace simbólico en el directorio apropiado, como sucedería con el siguiente comando:

```
# ln -s /opt/apache/bin/apachect /etc/rc.d
```

Este comando es muy parecido al comando equivalente ofrecido para el caso de Red Hat en la sección anterior. Sin embargo, como una demostración más del objetivo principal de Slackware de buscar la simplicidad, no hay nada más que lo que hay. No es necesario hacer nada más. Con ese comando, Apache iniciará cada vez que arranque el sistema.

Instalación de Apache en Debian GNU/Linux

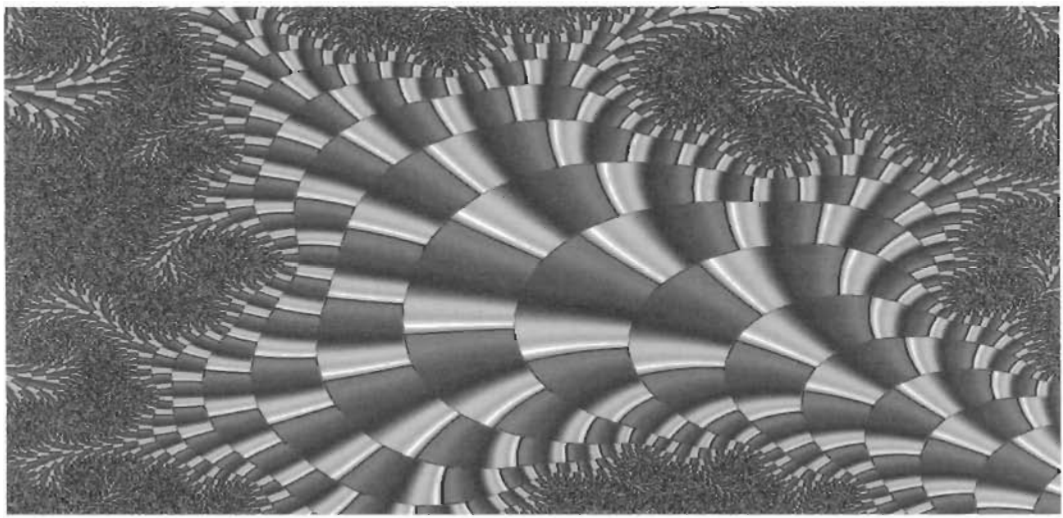
Como Red Hat Linux y Slackware Linux, Debian GNU/Linux incluye un paquete preconstruido para Apache, pero vuelve a ser una versión antigua del software. Si elige utilizar el material comentado en este capítulo para construir e instalar su propia instalación, encontrará que la mayor parte del material anterior referente a Red Hat Linux también se aplica a Debian GNU/Linux. Es decir, ya que las dos distribuciones emplean el modelo SysV para sus guiones de inicio, el proceso es muy similar. Sin embargo, tendrá que utilizar las herramientas de Debian para la administración de los guiones de inicio para los niveles de ejecución de su sistema, tal y como se comentó en el capítulo 6. Además, probablemente quiera crear ese guión separado (comentado anteriormente en el contexto de Red Hat Linux), que sencillamente invoca al guión apachectl.

Resumen

Tras la lectura de este capítulo, ya ha visto un ejemplo de cómo instalar y configurar el servidor HTTP Apache. No es una hazaña despreciable: Apache es una

pieza de software muy grande y flexible (y por ello complicada). Aunque probablemente aún no pueda añadir el grado "Administrador experto de Apache" a su currículum, debería de haber avanzado bastante en su camino hacia el dominio de Apache. Una vez que haya trabajado en la instalación de Apache en su propio sistema con su propia configuración, ya lo estará haciendo bastante bien. Aún más, si puede configurar con éxito Apache, entonces la mayor parte del resto del software comenzará a parecerle algo sencillo.

El siguiente capítulo de la tercera parte continúa la senda de mostrar diversas instalaciones de software, centrándose en el sistema de versiones concurrentes (CVS). El capítulo 12 mostrará la instalación y configuración de un servicio inetd.



12
Sistema de versiones concurrentes (CVS)

El sistema de versiones concurrentes (*Concurrent Version System*, CVS) es un miembro de una clase de herramientas conocidas con diversos nombres como herramientas de gestión de cambios, herramientas de gestión de software y configuración (*software and configuration management*, SCM) y algunos otros nombres más. CVS actúa como un repositorio que hospeda una copia maestra de un proyecto de software y que permite a los desarrolladores "sacar una versión verificada" (*check out*) para utilizarla como su propia copia de trabajo (conocida como caja de arena o *sandbox*) del mismo modo que una persona puede sacar un libro de una biblioteca. Al actuar como repositorio, CVS también registra, rastrea, y administra los cambios sobre el software a medida que los desarrolladores los van realizando.

CVS es una versión de nueva generación de una clase más primitiva de herramientas parecidas. La herramienta Unix original para estos cometidos se llamaba Sistema de Control de Código Fuente (*Source Code Control System*, SCCS). SCCS también hacía de repositorio y permitía a los desarrolladores extraer (*check out*) e introducir (*commit*) cambios, pero era bastante más restrictiva con respecto a lo que permitía hacer a los desarrolladores. Por ejemplo, sólo un desarrollador podía editar un archivo dado en un mismo momento, y el repositorio tenía que encontrase en un disco local. El proyecto GNU implementó una herramienta funcionalmente similar conocida como Sistema de Control de Revisiones (*Revision Control System*, RCS). CVS se basa en RCS y lo emplea en el núcleo, pero proporciona prestaciones mejoradas.

El mayor motivo para que CVS reclame la fama es su habilidad para hacerse cargo del desarrollo asíncrono. Recuerde que SCCS y RCS permiten a un único

desarrollador por vez editar un archivo, como por ejemplo un archivo de cabecera. Si este archivo de cabecera es uno muy habitual, compartido por todos los aspectos del software, entonces para muchos desarrolladores trabajar sobre distintas partes del software al mismo tiempo se convierte en algo muy difícil. El enfoque que CVS toma es mantener la caja de arena de cada desarrollador completamente disociada del repositorio maestro, hasta que el desarrollador introduzca un cambio. En ese momento, se sincroniza la caja de arena con el repositorio y se fusionan las discrepancias que pueda haber. CVS puede manejar y fusionar la mayoría de los tipos de discrepancias, pero en un pequeño conjunto de casos se necesita la intervención humana para resolver los conflictos. (Realmente, por lo general es una buena práctica revisar todas las combinaciones, incluso las que CVS realiza de manera automática, ya que ninguna pieza de software es perfecta.)

Además del gran aumento de la productividad que ofrece CVS, también proporciona algunas prestaciones adicionales. CVS funciona en varios modos diferentes, incluyendo tanto un modo cliente-servidor (y, de hecho, varios de estos modos), como el modo del repositorio local tradicional utilizado por SCCS. Estas características hacen de CVS una herramienta extremadamente útil para coordinar los cambios de un equipo de desarrolladores distribuido; por eso, CVS es inmensamente popular en los proyectos de desarrollo de código abierto, e incluso se está volviendo cada vez más habitual dentro de las grandes corporaciones globales como un modo de compartir sus proyectos por todo el mundo.

CVS es utilizado más habitualmente para el desarrollo de software, como repositorio para el código fuente. Sin embargo, no hay nada en CVS que sea específico a los lenguajes de programación o al código fuente. De hecho, CVS se puede utilizar para administrar cualquier tipo de proyecto. Funciona mejor para proyectos que consisten en archivos basados en texto, pero también puede utilizarse con archivos binarios (como imágenes o fotografías) a costa de parte de su habilidad para combinar cambios. Por eso comúnmente se utiliza CVS para administrar objetos que no son código fuente como archivos HTML para sitios web, datos XML, archivos de configuración de texto y demás.

Las principales virtudes de CVS son su modelo descentralizado (ya que los desarrolladores no están atados al repositorio, sino aislados hasta que intentan introducir sus modificaciones) y su apoyo al desarrollo asíncrono. Sin embargo, explicar cómo se utiliza CVS como parte de una metodología de administración de proyectos software con éxito es un tema suficientemente extenso como para merecer su propio libro. Los lectores interesados deberían consultar el excelente manual de referencia sobre CVS escrito por Per Cederqvist que se encuentra en www.cvshome.org/docs/manual/cvs.html.

Como muchos de los otros ejemplos de los capítulos anteriores referentes a la instalación, CVS está incluido en la mayoría de las distribuciones. Sin embargo, típicamente sólo instalan CVS en su modo cliente, en lugar de en su modo servidor. Esto tiene sentido, porque preparar un servidor CVS (o incluso un simple repositorio CVS) sólo es útil para los desarrolladores de software.

Este capítulo detallará cómo instalar CVS con sus prestaciones de servidor como un servicio inetd. Al leer este capítulo, también conseguirá una buena perspectiva

acerca de cómo instalar cualquier servicio basado en inetd, al igual que será capaz de instalar y configurar su propio repositorio CVS funcional. CVS es un ejemplo clásico de un servicio inetd, por eso incluso aquellas personas que no sean desarrolladores deberían considerar examinar con cuidado este capítulo.

Nota: Sólo los hechos: Sistema de Versiones Concurrentes

Propósito: El Sistema de Versiones Concurrentes es una herramienta que proporciona a los desarrolladores de software la habilidad de gestionar cambios en sus productos.
Autores: ¡Muchos!
Sitio web: www.cvshome.org
Descripción: CVS es una herramienta de gestión de cambios diseñada para ayudar a los desarrolladores de software (o a otros equipos que produzcan colectivamente un trabajo amplio) en la organización, seguimiento y administración de su trabajo. CVS ayuda a los desarrolladores a colaborar entre sí sin interferir ni destruir el trabajo del otro. CVS supera a una gran variedad de herramientas antiguas dedicadas a la misma área.

Instalación de CVS

Para instalar CVS en su modo servidor, se necesita instalar los programas cliente. Esto es así porque el modo servidor de CVS funciona como un servicio inetd. El "superservidor de Internet" inetd se comentó en el capítulo 4; en pocas palabras, inetd es un programa que proporciona un soporte de red básico a otros programas, de modo que no tengan que gestionar su propio soporte de red. Los programas cliente de CVS estándar utilizan esta funcionalidad para "transformarse" en software servidor. Esta sección describirá cómo instalar los programas CVS estándar con el soporte apropiado para el modo servidor. La sección "Configuración de CVS" describirá cómo configurar los programas estándar para el modo servidor más popular.

Opciones en tiempo de compilación

Como la mayoría de los programas de código abierto, CVS soporta varias opciones en tiempo de compilación. Como se comentó en el capítulo 7, una opción en tiempo de compilación es un parámetro que incrusta una propiedad particular en el software en el momento en que se construye. Típicamente, las opciones en tiempo de compilación se utilizan para habilitar o inhabilitar porciones particulares de funcionalidad.

La primera opción que se necesita fijar es la opción `--prefix`. Esta opción determina dónde se instalarán en última instancia los archivos binarios del programa. En el capítulo 7 comenté las posibles ubicaciones para el software (concretamente, los directorios /usr/local y /opt). Ya que la instalación de CVS es una instalación sencilla que consiste en archivos bastante simples que sólo necesitarán ser actualizados ocasionalmente (si es que alguna vez lo necesitan), una buena ubicación sería /usr/local.

Nota

Si desea más información acerca de la elección entre /usr/local y /opt, lea la sección "Decidir la ubicación" del capítulo 7.

CVS soporta opciones en tiempo de compilación que habilitan o inhabilitan características relacionadas con el comportamiento de los modos cliente y servidor. La tabla 12.1 resume las características principales soportadas por CVS y recoge una lista de opciones para habilitarlas e inhabilitarlas. Se puede obtener información adicional al ejecutar el comando `./configure --help` en el directorio raíz de CVS. (El comando `./configure` es la herramienta autoconf de GNU también comentada en el capítulo 7; la opción `--help` muestra información sobre las opciones soportadas, y la tabla 12.1 es en gran medida un resumen de esta información.)

Tabla 12.1. Opciones en tiempo de compilación de CVS.

Característica	Opción de habilitación	Opción de inhabilitación
Soporte para la autenticación mediante el protocolo Kerberos*	`--with-krb4=[ruta]`	`--without-krb4`
Soporte para la autenticación mediante el protocolo GSSAPI*	`--with-gssapi=[ruta]`	`--without-gsspai`
Soporte para el cifrado de datos en modo cliente/servidor cuando se utiliza Kerberos	`--enable-encryption`	`--disable-encryption`
Soporte para el modo cliente	`--enable-client`	`--disable-client`
Soporte para el modo servidor	`--enable-server`	`--disable-server`

* Opciones con un valor de [ruta] deben incluir una ruta hacia una instalación de la biblioteca correspondiente, por ejemplo, la biblioteca Kerberos. Consulte la documentación de CVS.

En general, las opciones se reducen a habilitar estas tres posibilidades:

- Modos cliente y servidor (mediante las opciones `--enable-client` y `--enable-server`)
- Cifrado de datos a través de la red (mediante la opción `--enable-encryption`)
- Mecanismo alternativo de autenticación como Kerberos o GSSAPI.

La amplia mayoría de las instalaciones CVS tendrán habilitados ambos modos cliente y servidor, muchas tendrán habilitado el cifrado de datos y algunas tendrán habilitados los mecanismos de autenticación.

Las redes que ya utilicen Kerberos o GSSAPI (como una red corporativa o universitaria) se beneficiarán de estas opciones; al habilitarlas se permitiría utilizar una alternativa al modo del servidor de contraseñas (*password server*, pserver) que es bastante inseguro. Además, si se está utilizando Kerberos, también se puede habilitar el cifrado. Esto impide que material valioso y secreto pueda ser interceptado a través de la red. (En realidad, todavía puede ser interceptado, pero como está cifrado no podrá leerse.)

La mayoría de las instalaciones necesitarán ambos modos cliente y servidor. Sin embargo, un administrador podría configurar las instalaciones CVS en estaciones de trabajo para que contuvieran sólo el soporte de cliente, mientras que las instalaciones CVS en servidores contendrían sólo soporte para el modo servidor. De este modo se impediría que los usuarios hicieran funcionar servidores CVS en sus estaciones de trabajo e impediría que los usuarios que entraran en un servidor pudieran ejecutar clientes CVS. Esta funcionalidad puede utilizarse para formar un conjunto de políticas de uso de CVS, aunque este tema no se trata explícitamente en este capítulo.

Para este ejemplo, mostraré cómo habilitar los modos cliente y servidor. No desea explícitamente el soporte para Kerberos o GSSAPI; sin embargo, si las bibliotecas ya están instaladas, también podría utilizarlas, por eso simplemente dejará que el programa ./configure detecte su presencia. (Por defecto, ./configure siempre autodetectará cualquier característica que no incluya específicamente.) El siguiente comando es el comando utilizado para configurar el software:

```
% ./configure --prefix=/usr/local --enable-client --enable-server
```

Instalación a partir del código fuente

Una vez que se han seleccionado las opciones en tiempo de compilación y se ha ejecutado el programa ./configure, se puede compilar el software. Esta sección describe cómo construir e instalar el software. Para CVS, este es un procedimiento bastante simple.

En esencia, todo lo que tiene que hacer es ejecutar el comando `make`, seguido de `make install`, como se muestra aquí:

```
make
make install
```

A medida que se compila el software, aparecerán diversos mensajes sobre el proceso de construcción en la pantalla, y aparecerán aún más mensajes cuando se copie CVS en /usr/local como parte del proceso de instalación.

Si se completan ambos comandos sin errores, ¡ya estará instalado CVS! La sección siguiente, llamada "Configuración de CVS", comentará cómo configurar CVS (mediante sus opciones en tiempo de ejecución) de cara a su uso real.

Nota: Selección del mejor modo de CVS

Este libro comenta cómo instalar y configurar CVS para el modo pserver. Sin embargo, CVS soporta un amplio número de modos diferentes e incluso varios modos distintos de servidor. (Por ejemplo, hay modos que utilizan el sistema de autenticación Kerberos en lugar de las contraseñas.) El modo pserver comentado en este libro es probablemente la manera menos segura de utilizar CVS. Sin embargo, el modo pserver es el más popular, que es por lo que se comenta en este libro. Los administradores preocupados por la seguridad (y en especial los administradores que se encarguen de un repositorio accesible en una red pública) deberían consultar la documentación de CVS sobre cómo utilizar modos más seguros.

Configuración de CVS

Esta sección describirá cómo configura CVS para el modo servidor una vez que se han instalado las herramientas CVS cliente estándar. El modo de servidor de contraseñas de CVS funciona como un servicio que inicia desde el "superservidor" inetd. Configurar CVS para que se pueda acceder como servidor es tan simple (¡y tan complicado!) como configurar correctamente inetd. Como ya se ha indicado, el modo pserver es una manera bastante insegura de hacer funcionar un repositorio CVS. Sin embargo, esta sección trata tanto acerca de mostrar cómo configurar un servicio inetd como acerca de instalar CVS, y por eso es por lo que se muestra el modo pserver.

Lamentablemente, a veces distribuciones diferentes utilizan versiones de inetd diferentes; por ejemplo, Red Hat Linux utiliza el programa xinetd, mientras que Slackware Linux utiliza una implementación de Unix más tradicional. Sea cual sea la implementación de inetd que se utilice, tiene que hacer funcionar el mismo servicio CVS. Las siguientes secciones describen las opciones de configuración relacionadas con CVS que son comunes para todas las implementaciones de inetd. La sección "Instalación de CVS en diferentes distribuciones" que se encuentra más adelante en este mismo capítulo describe cómo enganchar CVS y sus parámetros

del modo servidor en las implementaciones de inetd para las tres distribuciones de muestra.

Las diversas opciones en tiempo de compilación que soporta CVS se comentaron con anterioridad en este capítulo; si desea una lista de estas características, échele un vistazo a la tabla 12.1 De manera notable, varias opciones tienen que ver con la autenticación en el modo servidor, como Kerberos y las simples contraseñas. Esta sección sólo comentará la configuración de un servidor CVS en el modo de contraseña normal (modo pserver).

Muchas distribuciones podrían no soportar alguno de los otros modos (por ejemplo, si no tiene instalado Kerberos), y el modo pserver es el más común. Además, el modo pserver es más simple y por eso es el mejor ejemplo de una instalación de un servicio inetd. Consulte la documentación de CVS acerca de información sobre las otras opciones de configuración.

Una instalación de CVS en modo pserver consiste en unos pocos pasos separados, que se describen en la siguiente sección. Estos pasos realmente van más allá de lo estrictamente necesario para instalar un servicio inetd. Sin embargo, cada servicio inetd va a tener algún tipo de requisito de configuración específico, como estos pasos u otros. Por eso resulta útil rastrear la configuración global de CVS, no sólo los pasos relacionados con inetd, para ganar perspectiva sobre técnicas de configuración generales.

Creación del directorio repositorio

El primer paso en la instalación de CVS en modo pserver (o cualquier otro modo) es crear un directorio para los archivos de datos. Recuerde que CVS implica el concepto de un repositorio; este repositorio debe ser un directorio real en el que se almacenen los archivos que se envían (una vez verificados los cambios y resueltas las discrepancias), al igual que los historiales de esos archivos e información de control variada.

Este directorio puede encontrarse en cualquier lugar que se considere conveniente. Una posibilidad podría ser un subdirectorio de /opt o /usr/local. Sin embargo, el FHS especifica un directorio para cosas como estas bases de datos: el directorio /var. Entonces, una elección razonable sería el directorio /usr/local/var/cvs. (Se escogió el directorio /usr/local/var en lugar de /var porque, como se comentó en el capítulo 7, la mayoría de los gestores de paquetes nativos ignoran el contenido de /usr/local, tratándolo como un material específico del sistema.)

Se puede usar el siguiente comando para crear el directorio:

```
% mkdir -p /usr/local/var/cvs
```

El argumento `-p` del comando `mkdir` le indica que cree no sólo el directorio cvs, sino también cualquiera de sus directorios antecesores que no exista previamente. Por ejemplo, frecuentemente no hay un directorio /usr/local/var por defecto, de modo que el comando anterior fallaría si no tuviera la opción `-p`; con ella se creará el directorio /usr/local/var a la vez que su directorio hijo /usr/local/var/cvs.

Nota: Repositorios CVS críticos

Utilizar /usr/local/var (o incluso /opt/var, dependiendo de los gustos) para guardar los datos de CVS es una buena elección para una instalación personal o para un uso limitado. Sin embargo, usos más importantes de CVS, como el que puedan hacer organizaciones que producen software, presentan necesidades adicionales, como realizar copias de seguridad de datos de manera regular y posiblemente disponer de servidores con tolerancia ante fallos y "*hot backup*" (que realizan y restauran rápidamente copias de seguridad). Después de todo, un repositorio CVS representa la copia maestra de lo que sea que gestione; si pone todos sus huevos en una misma cesta, ¡tendrá que ser una cesta extremadamente segura! Este tipo de usuarios de CVS probablemente prefiera colocar el repositorio CVS en un volumen, partición, disco o servidor (u otro tipo de dispositivo similar) separado, y montarlo como algo parecido a /cvs. Estos usuarios también deberían asegurarse de realizar con frecuencia copias de seguridad del directorio de repositorio /cvs, y probablemente tendrían que usar uno de los modos de CVS más seguros para proteger mejor a su servidor de un ataque.

Creación de cuentas de usuario

Una vez que se ha seleccionado y creado el directorio que va a hospedar al repositorio CVS y los datos relacionados, se debe crear al menos una cuenta de usuario para acceder a él. Puede realizar esta tarea mediante una gran variedad de métodos. En esta sección describiremos someramente esos métodos.

Confiar en las cuentas del sistema

Por defecto, CVS permitirá el acceso a cualquier usuario que disponga de una cuenta válida en el sistema subyacente. Por ejemplo, cualquier usuario con un identificador de usuario y una contraseña válidos será capaz de entrar en el sistema y acceder a los contenidos del repositorio CVS (aunque ese usuario también debería disponer de los permisos Unix sobre el sistema de archivos correctos). Por eso, el modo más simple de gestionar el acceso a CVS es simplemente pasar la pelota y dejar que los servicios estándar del sistema operativo se encarguen de ello.

Sin embargo, en este caso, los usuarios realmente estarán conectados al sistema como ellos mismos, y por eso accederán al sistema de archivos con sus permisos normales.

Esto podría convertirse en un engorro administrativo, ya que la administración de los permisos de archivo es algo complicado. (Determinar los permisos de archivo para este ejemplo se comenta en una sección posterior.) Además, utilizar este enfoque quiere decir que los usuarios tendrán que tener literalmente una cuenta en la máquina, lo que significa que pueden entrar en ella, al igual que utilizar CVS;

este puede ser un riesgo de seguridad si se pretende que la máquina sea estrictamente un servidor CVS.

La cara positiva de dejar que el sistema gestione las cuentas es que así es fácil administrar los usuarios (sencillamente porque no tiene que administrarlos en absoluto.) La cara negativa es que es más difícil administrar los permisos de archivos y es necesario que cada usuario del servidor CVS tenga una cuenta en el sistema. Esta técnica es más útil cuando el servidor CVS en modo pserver lo utiliza un grupo de trabajo pequeño y funciona con un servidor (quizás un servidor de desarrollo) al que todos los miembros del grupo tienen acceso. Es menos útil para proyectos más grandes con una base de usuarios más extensa.

Compartir una única cuenta de usuario

CVS ofrece una alternativa a los administradores que no quieren ceder las tareas de la autenticación al sistema operativo. En su lugar, se puede preparar una única cuenta para el sistema CVS, e incluso puede ser una cuenta inactiva (es decir, una cuenta que no permita entrar en el sistema). Se puede crear el directorio del repositorio CVS de modo que sólo sea visible para este único usuario real, y el administrador puede crear varias cuentas "fantasma" que sólo son visibles por parte de CVS.

Esto se lleva a cabo mediante el archivo passwd local de CVS. Este archivo es parecido al archivo /etc/passwd estándar de Unix, pero sólo lo utiliza CVS. El archivo permite a los administradores asignar cuentas CVS fantasma a cuentas físicas. Se puede añadir usuarios a CVS simplemente añadiéndolos al archivo passwd de CVS y asignando las cuentas fantasma a la única cuenta física real.

La parte positiva de este enfoque es que es un poco más seguro, ya que sólo hay que mantener una cuenta, y se puede conceder a la cuenta permisos restringidos para reducir el daño potencial en caso de una ruptura de la seguridad. Además, los permisos del directorio de repositorio son más fáciles de administrar, ya que sólo hay una cuenta física que necesite acceder a él. La parte negativa es que los administradores tienen que administrar un archivo passwd específico de CVS. Este enfoque es probablemente más apropiado para grandes proyectos con muchos desarrolladores, con al menos una persona designada como administrador oficial del servidor CVS.

Este enfoque es el que se tomará en este libro durante el resto del ejemplo. En realidad, también es posible utilizar ambas técnicas de manera simultanea, en una especie de modo híbrido. Establecer los permisos para ambas técnicas y los procedimientos para añadir usuarios al archivo passwd CVS fantasma es algo que se comentará en la siguiente sección.

Establecimiento de permisos

Esta sección describirá cómo configurar los permisos para el directorio de repositorio y ofrecerá algunos detalles sobre cómo administrar las cuentas de usuario. Cada uno de los dos casos delineados en la sección anterior se comentará en esta. El caso en el que la autenticación de CVS se delega al sistema operativo implica unos

permisos del directorio de repositorio más complicados pero una administración de usuarios más simple, mientras que lo contrario es cierto para el caso de las cuentas CVS virtuales.

Parámetros para el caso de autenticación de sistema

En el primer caso, CVS se configura simplemente para que el sistema operativo sobre el que se ejecuta se haga cargo de la autenticación de los usuarios. Es decir, cada vez que entre en el sistema un usuario, CVS esencialmente se limita a preguntarle al sistema operativo si tanto el nombre de usuario como la contraseña son válidos. Si lo son, el proceso que ejecuta CVS pasa a utilizar la identidad de ese usuario e intenta realizar los comandos CVS que el usuario haya solicitado.

Esto significa que el servidor CVS realmente está ejecutándose con los permisos Unix de la cuenta de usuario que se utilizó para entrar en el sistema. Eso, a su vez, significa que si una cuenta de usuario en particular no dispone de permisos para leer o escribir en el directorio del repositorio CVS real, ¡entonces la cuenta no podrá acceder a la instalación de CVS!

Por eso, se deben especificar los permisos de los archivos que se encuentren en el directorio. Típicamente, las instalaciones en este caso tienen tres propiedades principales:

- El directorio de repositorio pertenece a un grupo específico de CVS definido en el archivo /etc/group (o mediante otro mecanismo, como el servicio de información de red [NIS]).
- Todos los usuarios del sistema que necesitan acceder a CVS se incluyen como miembros del grupo CVS.
- El directorio de repositorio tiene activados los permisos de lectura y escritura para el grupo, y también tiene activado el "bit adhesivo" (sticky bit).

La primera propiedad es fácil de lograr. La mayoría de las distribuciones contienen herramientas para gestionar usuarios y grupos, y se puede usar el comando `chown` o `chgrp` para fijar el grupo asociado con un directorio dado. Una vez que existe el grupo, un comando como el que viene a continuación (que utiliza la ruta del ejemplo de /usr/local/var/cvs) concederá la propiedad del directorio al grupo cvsusers:

```
% chgrpd -R cvsusers /usr/local/var/cvs
```

La segunda propiedad de la lista también es fácil de conseguir. Para sistemas autónomos que utilizan los archivos /etc/passwd y /etc/group para la autenticación, hay típicamente utilidades (como `useradd` y `groupadd`) para añadir, borrar, y administrar los contenidos de usuarios y grupos. Esta es una tarea bastante básica, por eso si no sabe cómo hacer esto, consulte la documentación de su distribución. Los sistemas que están en una red pueden utilizar otro esquema de autenticación (como NIS, Kerberos u otro módulo PAM) que disponga de un mecanismo distinto para administrar grupos y miembros de grupos; si piensa que podría

tratarse de este caso, puede que tenga que ponerse en contacto con sus administradores de red en busca de ayuda para crear un grupo del modo apropiado.

La tercera propiedad (activar el bit adhesivo en el directorio) es igualmente simple, pero necesita una ligera explicación. Debería sentirse familiarizado con el esquema de permisos de archivos Unix tradicional; es decir, hay tres clases de acceso y cuatro permisos que pueden otorgarse individualmente a cada clase. Las clases son: la cuenta que posee el archivo (*user*, usuario), los miembros del grupo al que pertenece el archivo (*group*, grupo), y cualquier otra cuenta (*other*, otro). El comando `chmod` se utiliza para fijar los permisos para cada clase, y los permisos son de lectura (*read*), escritura (*write*) ejecución (*execute*, que se utiliza para los programas y guiones ejecutables y para ver el contenido de un directorio), y adhesivo (*sticky*).

El permiso adhesivo (a veces llamado bit adhesivo) indica que el archivo debe conservar los permisos que se le han fijado, sin importar lo que suceda. Normalmente, el bit adhesivo se utiliza para indicar que un programa ejecutable debe ejecutarse como el usuario o grupo que posee el archivo, en lugar del usuario que ejecutó realmente el programa. (Cuando se utiliza de esta manera, el bit adhesivo se llama bit de adherencia de usuario [*setuid bit*].) Sin embargo, cuando se aplica a un directorio, el bit adhesivo significa que todos los subdirectorios y archivos que se creen bajo el directorio conservarán los mismos permisos que el propio directorio. Los permisos correctos y el bit adhesivo pueden fijarse para el directorio /usr/local/var/cvs mediante los siguientes comandos:

```
% mkdir -p /usr/local/var/cvs
% chgrp -R cvsusers /usr/local/var/cvs
% chmod -R ug+rwx /usr/local/var/cvs
% chmod -R ug+s /usr/local/var/cvs
```

El parámetro `-R` pasado a `chgrp` y `chmod` les indica que sean "recursivos" y el comando se repita en todos los archivos y subdirectorios contenidos dentro del directorio pasado como parámetro.

El motivo por el que es importante utilizar el bit adhesivo en el directorio de repositorio CVS es porque hay múltiples usuarios, y el programa CVS en modo pserver realmente se ejecuta como si fuera cada uno de los usuarios individuales. Si un usuario añade un archivo a CVS, el archivo realmente será creado por el usuario que lo envió y adoptará los permisos por defecto de ese usuario. Esto significa que es posible que un usuario introduzca en el repositorio CVS un archivo que no está accesible para otro usuario, lo que, como poco, resulta inconveniente. Al activar el bit adhesivo en el directorio de repositorio CVS se anula ese comportamiento y se fuerza que cualquier archivo nuevo tenga los mismos permisos que el propio directorio /usr/local/var/cvs, así que se previene este problema.

Una vez que se han cumplido estas tres condiciones básicas, el repositorio CVS está listo para ser utilizado (aunque por supuesto aún no se ha configurado bastante el propio servidor; siga leyendo los demás pasos). Aún así, recuerde que delegar la autenticación de usuario en el sistema operativo es sólo uno de los modos en que se puede gestionar el CVS; la siguiente sección cubrirá el otro caso.

Nota: ¿Qué usuario posee el repositorio?

Los lectores avispados pueden preguntarse cuál es el usuario al que pertenece realmente el directorio de repositorio CVS (que en este ejemplo es /usr/local/var/cvs). Es decir, ¿qué usuario tiene que ejecutar el comando `mkdir` para crear el directorio? Bueno, la belleza de la técnica del bit adhesivo reside en que eso no importa. El usuario *root* puede crear el directorio, y mientras que los bits adhesivos se hayan fijado correctamente, como en el ejemplo, cualquier usuario del grupo de CVS tendrá entonces acceso completo al repositorio.

Parámetros para el caso de cuentas virtuales

La segunda técnica principal de autenticación para una instalación de CVS en modo pserver consiste en utilizar una lista de usuarios virtuales específica para CVS. En este caso, sólo se necesita crear un único usuario en el sistema operativo subyacente. Además, este usuario nunca necesitará una shell de entrada al sistema, por eso la cuenta puede tener una contraseña vacía o inhabilitada de algún otro modo, para evitar los riesgos de seguridad que implicaría permitir la entrada de un usuario remoto. En lugar de tener una cuenta en el sistema, los usuarios de CVS tienen cuentas fantasma o virtuales dentro del propio sistema CVS; las cuentas virtuales se asignan a la cuenta física en el sistema.

Preparar este escenario es bastante simple. Ya que sólo hay un usuario, no es necesario que exista un grupo de usuarios aparte, y no hay que gestionar ninguna tontería sobre bits adhesivos. En vez de eso, simplemente se crea el repositorio, que pertenece a la única cuenta de usuario real, y todos los usuarios de CVS virtuales se asignan a (y funcionan con los permisos de) la cuenta física. Por eso, crear correctamente el directorio de repositorio es tan simple como ejecutar estos comandos:

```
$ mkdir -p /usr/local/var/cvs
# chown -R cvs:cvs /usr/local/var/cvs
```

Este ejemplo asume que el nombre de usuario físico de CVS es `cvs`, y que se ha creado un grupo para él, llamado también `cvs`. No todas las distribuciones tienen un grupo aparte para cada cuenta de usuario; compruebe la documentación de su distribución acerca de cómo administrar usuarios y grupos.

Añadir usuarios virtuales a CVS es un poco más complicado. Una vez que el servidor CVS se encuentra activo, hay un módulo especial llamado CVSROOT que los usuarios pueden extraer como una copia de trabajo. Este directorio contiene diversos archivos que controlan el comportamiento de CVS; uno de esos archivos es CVSROOT/passwd. Este archivo contiene nombre de usuario y contraseñas para las cuentas virtuales, y un campo adicional que especifica el nombre de la cuenta física con el que se corresponde cada cuenta virtual. En este ejemplo, todas las cuentas se asignan al usuario físico cvs.

Lamentablemente, gestionar el contenido del archivo CVSROOT/passwd (y realizar tareas administrativas adicionales) está más allá de lo que se pretende comentar en este capítulo, que simplemente se centra en la instalación de CVS. Si está interesado, consulte la documentación sobre CVS que se puede encontrar en www.cvshome.org, en especial el manual de Cederqvist. El resto de esta sección se centrará en la instalación de CVS en modo pserver mediante el servicio inetd.

Nota: Uso de la autenticación CVS híbrida

Estos ejemplos describen dos modos de preparar la autenticación de usuario con CVS: ya sea delegando completamente la autenticación al sistema operativo o asignando cuentas de usuario CVS fantasma a una cuenta del sistema física real. En realidad, no se trata de dos alternativas exclusivas: se puede configurar CVS para funcionar en ambos modos al mismo tiempo. Es decir, CVS siempre comprobará la lista de cuentas virtuales en primer lugar; sin embargo, si no se ha encontrado en esta lista al usuario, entonces lo comprobará en el sistema operativo. Esto significa que algunos usuarios pueden asignarse a una cuenta física mientras que se puede dejar que otros accedan al sistema CVS a través del sistema operativo. Esto añade una capa adicional de flexibilidad al conjunto.

Establecimiento de parámetros en tiempo de ejecución

La sección anterior describió cómo decidir los valores de diversos parámetros que CVS necesita para funcionar. El siguiente paso es reunir estos parámetros y valores en forma de un comando real que se pueda ejecutar. Este será el comando que está configurado para ejecutar el servidor inetd para las conexiones CVS, y esta sección comentará cómo formular el comando específico.

En realidad, es bastante simple. Si tuviera que introducir el comando necesario para ejecutar CVS directamente en la línea de comandos, tendría este formato general:

```
cvs -f --allow-root=<ruta al repositorio CVS> pserver
```

La clave es la lista de parámetros. El argumento `-f` indica a CVS que no tenga en cuenta los parámetros que puedan haberse fijado en el archivo de configuración ~/.cvsrc del usuario. Esto es importante, porque este tipo de opciones podría modificar el comportamiento de CVS cuando se esté ejecutando en modo servidor. El parámetro `--allow-root` especifica el directorio (o directorios, ya que puede haber más de un parámetro de este tipo) del repositorio CVS (que en el ejemplo de este libro es /usr/local/var/cvs). El argumento `pserver` no tiene ningún valor y

simplemente indica al programa (que normalmente comenzará a ejecutarse como cliente) que conmute al modo servidor utilizando la autenticación mediante contraseñas.

Por eso, si sustituimos la ruta del repositorio por la ruta del ejemplo, necesitará el siguiente comando específico:

```
% cvs -f --allow-root=/usr/local/var/cvs pserver
```

Este es el comando que hay que introducir en la configuración para inetd. La mecánica exacta para hacer esto varía según la implementación de inetd. Las siguientes secciones comentarán cómo realizar esto en las distribuciones de muestra.

Instalación de CVS en diferentes distribuciones

La sección anterior describió cómo seleccionar los parámetros para configurar inetd de manera que ejecute CVS en modo pserver, y en ella se produjo el comando real que se necesita para ejecutarlo. Las siguientes secciones describirán cómo configurar las distribuciones de muestra para que ejecuten ese comando desde el interior de inetd.

Instalación en Red Hat Linux

Como se comentó en el capítulo 4, Red Hat Linux usa el programa xinetd como su implementación de inetd. Este programa es una versión modernizada del clásico programa inetd que utiliza técnicas más recientes para convertir la configuración en algo más simple. Las implementaciones clásicas de inetd (que se comentan en la sección "Instalación en Slackware Linux") utilizan un archivo de configuración único para dirigir todos los servicios inetd. En contraste, xinetd utiliza la técnica de "depositar configuraciones" (también comentada a lo largo del capítulo 4) para configurar servicios.

Actualización del archivo /etc/services

Esta sección describirá como actualizar el archivo /etc/services, que contiene información que asigna nombres a los servicios de red, para incluir CVS. El archivo /etc/services es una parte del sistema operativo, más que específico de una distribución o una implementación de inetd en particular. Por eso, incluso aunque se presente dentro de la sección específica de Red Hat Linux, se necesitará aplicar esta modificación a cada distribución.

El archivo /etc/services asigna nombres simbólicos a puertos TCP/IP de red. Por ejemplo, el número de puerto estándar para las conexiones telnet es 23; por eso hay una línea en /etc/services que asigna el nombre "telnet" al puerto 23, como esta:

```
telnet          23/tcp
```

Probablemente ya haya distinguido el formato general a partir de esa línea. Es muy simple: un nombre de servicio seguido por un número de puerto y el tipo del número de puerto, como TCP (orientado a flujos) o UDP (orientado a datagramas). (Que un programa utilice TCP o UDP queda determinado por cómo se escribió el programa.)

Los programas escritos para hacer uso de la red suelen determinar el número de puerto que usarán buscando el nombre del servicio mediante el sistema operativo. Por ejemplo, un programa que necesite saber el puerto para el servicio telnet buscaría "telnet" a través del sistema operativo, que a su vez consultaría /etc/services y devolvería el número de puerto 23.

Normalmente los números de puerto estándar los asignan entidades como la IETF (*Internet Engineering Task Force*) o la IANA (*International Association for Names and Addresses*), o se acuerdan por convención. La mayor parte de las distribuciones Linux incluyen un archivo /etc/services reciente con la mayoría de los puertos y nombres de servicio con los que los usuarios pueden interactuar. De hecho, el archivo /etc/services que se incluye con una distribución reciente del sistema Red Hat Linux incluye sin problemas más de 400 servicios de red predefinidos. Esta lista es bastante completa y contiene muchos servicios que no resultan estrictamente necesarios, pero que están presentes como referencia; por ejemplo, ¡contiene una entrada para el juego en red multijugador Quake!

Ya que la lista de puertos estándar está en constante evolución, puede que las distribuciones más antiguas no tengan una lista tan reciente, por eso, cuando quiera añadir un servicio a /etc/services, compruebe siempre que no esté ya presente bajo un nombre diferente. Por ejemplo, la versión de /etc/services que se incluye con Red Hat Linux 7.3 ya tiene una entrada para CVS en modo pserver, que puede verse en la siguiente línea:

```
cvspserver              2401/tcp
```

Por eso, en los sistemas Red Hat Linux 7.3, no tendrá que modificar realmente el archivo /etc/services. Sin embargo, en algunas versiones anteriores de Red Hat Linux y en las copias de otras distribuciones puede no estar incluida esta línea, o puede incluirse bajo un nombre distinto (como `cvs` en lugar de `cvspserver`). Sin embargo, a pesar de eso, el puerto será el 2401. Añadir dos nombres para el mismo servicio no destruye nada; es simplemente algo redundante. (De hecho, ni siquiera necesita en absoluto añadir realmente esta línea al archivo /etc/services y puede usar en su lugar el número de puerto en bruto, directamente. Sin embargo, añadir la línea a /etc/services hace las cosas más "legibles para las personas".)

El directorio /etc/xinetd.d

El programa xinetd utiliza la técnica del directorio de depósito de configuraciones para configurar los servicios. Es decir, cuando se arranca xinetd, examina un directorio en particular en busca de archivos que contengan información y después lee esos archivos. Cada archivo contiene información para un servicio diferente que debe gestionar xinetd. Por defecto, y tal y como está instalado en los sistemas Red

Hat Linux, el directorio que utiliza xinetd para estos archivos de configuración es /etc/xinetd.d.

Para configurar un servicio para que se ejecute con xinetd, el primer paso es simplemente crear un archivo en /etc/xinetd.d que contenga los datos de la configuración para el servicio. Por convención, cada archivo en el directorio de configuración tiene un nombre de acuerdo con el servicio que configura (aunque no resulte necesario); por ejemplo, /etc/xinetd.d/wu-ftpd es el archivo de configuración para el programa servidor FTP wu-ftp. Para CVS, el archivo debería llamarse /etc/xinetd.d/cvs.

Construcción del archivo de configuración

El siguiente paso es crear los contenidos del archivo de configuración para CVS. Este archivo contiene información como si el servidor está habilitado (ya que xinetd permite que un servicio en particular pueda estar configurado pero no esté realmente activado), si el servicio está orientado a flujos o a datagramas, qué usuario debe ejecutar el servicio y otra información parecida. El archivo tiene el siguiente formato general:

```
service <nombre del servicio>
{
    [parámetro de configuración] = [valor]
}
```

El parámetro `<nombre del servicio>` es el nombre del servicio tal y como está especificado en /etc/services. Las líneas con parámetros de configuración contienen cada una un parámetro y un valor, y especifican la mayor parte de los datos de configuración reales que utilizará xinetd. La tabla 12.2 resume los parámetros más comunes.

Tabla 12.2. Parámetros del archivo de configuración de xinetd.

Parámetro	Significado
`disable`	"yes" o "no"; (sí o no), indica si el servicio está activado o no.
`socket_type`	"stream" o "dgram"; (flujo o datagrama), indica si el servicio está orientado a flujo o a datagrama.
`wait`	"yes" o "no"; (sí o no), indica si el servicio esperará conexiones entrantes adicionales, o si finalizará tras servir una única conexión.
`user`	Nombre del usuario bajo el que se debe ejecutar el servicio (normalmente *root*).
`server`	Ruta hacia el archivo ejecutable del programa servicio.
`server_args`	Cualquier argumento que se necesite pasar al programa.

(Continuación)

Parámetro	Significado
`nice`	Puede utilizarse para determinar una prioridad para el servicio; así se puede reducir (o incrementar) la prioridad para un servicio en particular para que utilice una porción reducida (o incrementada) del tiempo de CPU.

La tabla 12.2 muestra sólo una parte de todas las opciones disponibles que se pueden utilizar en un archivo de configuración para los servicios xinetd. Se puede obtener la documentación completa al respecto a través de la página del manual para xinetd.conf (es decir, mediante el comando `man xinetd.conf`). Aunque la tabla 12.2 supone un buen punto de partida para configurar un servicio xinetd y puede resultar adecuada para la mayoría de los casos simples, servicios inetd más complicados pueden necesitar algunas de las opciones que aparecen en la página del manual.

Recuerde que Red Hat Linux 7.3 ya incluye una entrada en /etc/services para CVS ("cvspserver") así que eso es fácil. (En una distribución que utilice xinetd pero no tenga previamente una entrada para CVS, el administrador tendrá que añadir la entrada y hacer referencia a ella en el archivo de configuración de xinetd, o utilizar el número del puerto.) Obviamente, el parámetro `disable` necesitará ser "no", o de otro modo xinetd no arrancaría el servidor CVS. El parámetro `socket_type` debe ser "stream", y el parámetro `wait` "no"; estas son propiedades del programa CVS en modo pserver y no pueden alterarse. El argumento `nice` es en realidad opcional en este caso, puede dejarse con un valor "10". De este modo se mantendrá al servidor CVS funcionando con una prioridad razonable con respecto a los demás procesos del sistema. (Es decir, no monopolizará la CPU, pero tampoco será completamente ignorado por el sistema. Consulte la página del manual sobre el comando `nice` si desea más información al respecto.)

Las tres opciones restantes (`user`, `server` y `server_args`) son los valores que se decidieron antes en esta sección. En el ejemplo, el usuario bajo el que se ejecuta CVS es cvs.

El propio programa se compiló a partir del código fuente y se instaló en /usr/local/bin, así que el argumento `server` debe ser /usr/local/bin/cvs. Por último, el valor del parámetro `server_args` debe corresponder a los argumentos descritos antes: `-f --allow-root=/usr/local/var/cvs pserver`. El listado 12.1 recoge el contenido completo del archivo de configuración de xinetd para CVS.

Listado 12.1. Archivo de configuración de xinetd para CVS en modo pserver.

```
service cvspserver
{
    disable     = no
    socket_type= stream
    wait        = no
    user        = cvs
    server      = /usr/local/bin/cvs
```

```
    server_args= -f --allow-root=/usr/local/var/cvs
    nice        = 10
    port        = 2401
}
```

Administración de xinetd

Ahora que ya se ha configurado CVS para su uso junto con xinetd, xinetd comenzará a gestionarlo. Sin embargo, si xinetd ya está en funcionamiento, entonces necesitará reiniciarlo. Además, algunas instalaciones de Red Hat Linux pueden tener instalado xinetd, pero no en funcionamiento. En estos casos, se necesitará configurar xinetd para que inicie durante el arranque del sistema. Esta sección comentará cómo llevar a cabo estas tareas.

Si ya está funcionando, reiniciar xinetd provocará que vuelva a cargar sus archivos de configuración; como resultado, xinetd "reconocerá" el nuevo archivo de configuración para CVS y arrancará el servidor. Se puede utilizar el comando `kill` para finalizar el antiguo proceso y reiniciar después xinetd desde la línea de comandos.

Sin embargo, en los sistemas Red Hat Linux hay un modo más fácil de hacer esto: como se comentó en el capítulo 4, Red Hat incluye un comando llamado `service` que puede utilizarse para ejecutar, detener y reiniciar convenientemente servicios. Por eso, el comando `service xinetd restart` es mucho más simple que utilizar el comando `kill` y reiniciar xinetd manualmente.

Hay incluso una manera más de provocar que xinetd "reconozca" la instalación de CVS. Mediante el envío de la señal SIGUSR2 a xinetd, el administrador puede indicar a xinetd que recargue su archivo de configuración, sin finalizar realmente su ejecución, lo que es un poco más agradable que matar y reiniciar el proceso. Se consigue enviar la señal mediante el siguiente comando:

```
% kill -USR2 `pidof xinetd`
```

Hay realmente unas pocas señales especiales más que puede manejar xinetd; si desean más información sobre estas señales o sobre la señal SIGUSR2 ya mencionada, los lectores deberían consultar la página del manual para xinetd.

Otra utilidad conveniente más que ofrece Red Hat es el comando `chkconfig`. Este comando se mencionó en el capítulo 4, y permite a los administradores examinar, habilitar e inhabilitar servicios específicos (no sólo servicios xinetd) que se estén ejecutando en el sistema. El comando `chkconfig` puede habilitar o inhabilitar servicios xinetd, al igual que servidores estándar. Por eso, una vez que se ha configurado CVS en modo pserver como servicio xinetd, puede activarse y desactivarse mediante la herramienta chkconfig. De este modo, podría inhabilitar el servidor CVS de este ejemplo mediante el comando:

```
% chkconfig cvs off
```

Se puede rehabilitar más tarde el servidor CVS en modo pserver mediante un comando similar, reemplazando "off" por "on".

Instalación en Slackware Linux

Configurar CVS para que se ejecute en modo pserver en un sistema Slackware es parecido al espíritu de su instalación sobre Red Hat Linux, tal y como se comentó en la sección anterior. Sin embargo, Slackware utiliza una implementación tradicional del superservidor inetd, en contraste con la versión modernizada de Red Hat. Esto implica unas pocas diferencias en los detalles sobre cómo instalarlo.

Realmente, enganchar CVS en el sistema inetd de Slackware es bastante sencillo. En primer lugar, se necesita modificar /etc/service, precisamente del mismo modo en que era necesario para Red Hat Linux. Después, se necesita incluir en el archivo /etc/inetd.conf una línea que añada el servidor CVS en modo pserver a la lista de puertos que controla inetd. Una vez más, la filosofía de Slackware recibe su premio: esta operación sólo implica añadir una línea, en lugar de tener que enfrentarse con varios archivos en un directorio y un formato particularizado para el propio archivo de configuración. El código siguiente resulta por lo general suficiente para configurar CVS en el servidor inetd de Slackware (o, del mismo modo, en el servidor inetd de la mayoría de los sistemas similares a Unix).

```
cvspserver stream  tcp  nowait  root
     /usr/bin/cvs  cvs -f --allow-root=/usr/local/var/cvs pserver
```

Tenga en cuenta que las líneas anteriores deberían estar juntas como una única línea en su copia del archivo /etc/inetd.conf.

Nota

Véase el capítulo 5 para más detalles respecto a la configuración de aplicaciones para su funcionamiento junto con inetd.

La única dificultad que podría surgir con esta técnica es que muchas implementaciones tradicionales de inetd sólo soportan hasta cinco argumentos para el programa que se va a ejecutar (y uno de esos argumentos tiene que ser el propio nombre del programa). Si está encargado del funcionamiento de un repositorio CVS considerable con múltiples directorios raíz, podría llegar a este límite a través de la necesidad de los múltiples argumentos `--allow-root`. Sin embargo, para estos casos, puede configurarse inetd para que invoque un guión de shell muy simple que ejecute en su lugar el comando CVS necesario. Ya que el guión de shell no tendrá ningún límite impuesto sobre el número de argumentos, CVS se ejecutará con normalidad.

Instalación en Debian GNU/Linux

Debian GNU/Linux utiliza una implementación tradicional del programa inetd, como Slackware Linux. (Compare esto con el uso que hace Red Hat Linux de la

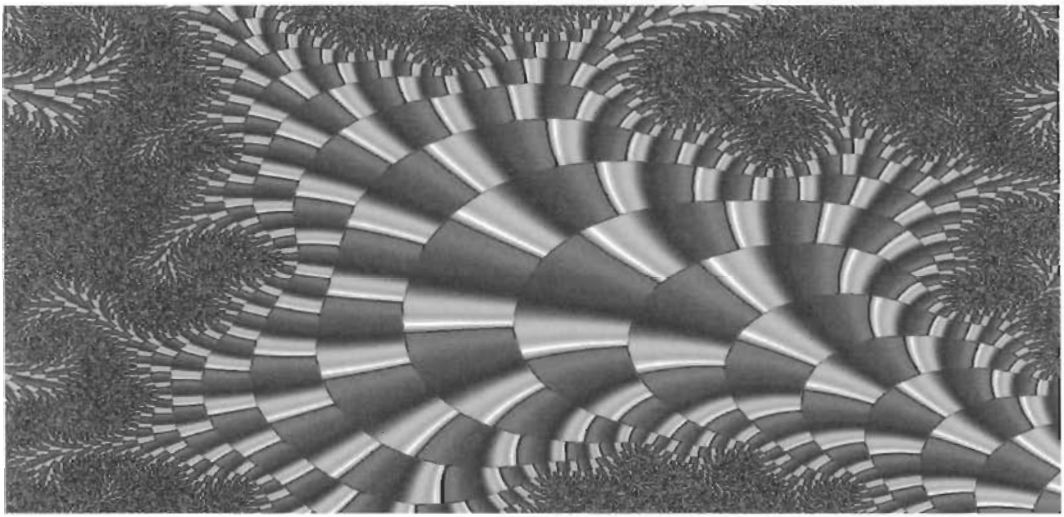
variante xinetd.) Obviamente, esto significa que preparar una instalación de CVS en modo pserver en un sistema Debian GNU/Linux es muy parecido a hacerlo en Slackware Linux; de hecho, los procesos son idénticos.

Sin embargo, Debian también ofrece una herramienta adicional que se puede utilizar para administrar los servicios que ejecuta su versión de inetd. Esta herramienta se llama `update-inetd` y está comentada en la sección titulada "Uso del programa update-inetd" del capítulo 6. Puede desear investigar las prestaciones de esta herramienta en lugar de hacer las cosas a mano, como tendría que hacer con Slackware Linux.

Resumen

En este capítulo, ha leído acerca de cómo instalar y configurar el sistema de versiones concurrentes. Ha aprendido acerca de su naturaleza dual como servidor y cliente, y también ha aprendido a configurar la faceta de servidor para que funcione como servicio inetd. Además, aprendió a utilizar las herramientas disponibles para cada una de las distribuciones de muestra para automatizar este proceso de configuración. Debería descubrir que las nuevas habilidades que ha adquirido resultan muy útiles para la configuración y administración de cualquier servicio basado en inetd que pueda encontrarse.

El próximo y último capítulo de esta tercera parte se ocupa del Java 2 Software Development Kit (J2SDK) de Sun Microsystems. Esta aplicación resulta vital para el desarrollo de software escrito en el lenguaje de programación Java, extremadamente popular. Al mostrar la instalación de J2SDK, este libro también le enseñará cómo instalar y gestionar un conjunto bastante complicado de herramientas en línea de comandos, habilidades que podrá aplicar en un amplio abanico de casos.



13
Kit de desarrollo de Java de Sun Microsystems

El lenguaje de programación Java fue creado por Sun Microsystems para facilitar el desarrollo tanto de aplicaciones basadas en clientes como basadas en servidor. El lenguaje Java puede verse realmente como una plataforma completa, ya que no sólo especifica un lenguaje de programación y las sintaxis y API asociadas, sino también un entorno de ejecución conocido como máquina virtual. La especificación de la máquina virtual Java (*Java Virtual Machine*, JVM) describe una especie de ordenador genérico (la máquina virtual) que en esencia se emula sobre el hardware real. El kit de desarrollo de software Java 2 de Sun (*Java 2 Software Development Kit*, J2SDK) consiste en un conjunto de herramientas (como un compilador) para el lenguaje Java y una JVM para ejecutar las aplicaciones. Se han escrito incontables libros sobre los igualmente incontables aspectos de la plataforma Java, y por eso este libro no va a ahondar más en el tema. Los lectores interesados pueden consultar cualquiera de los libros disponibles en el mercado (como "Java in a Nutshell", escrito por David Flanagan, y "Learn Java with JBuilder6", escrito por John Zukowski).

Nota: Distinguir entre J2SDK y JDK

El kit de desarrollo de software Java 2 (J2SDK) es un producto de Sun Microsystems. Sin embargo, otros fabricantes (como IBM) que producen software similar para ejecutar y desarrollar software Java frecuentemente denominan a sus productos kit de desarrollo Java (*Java Development Kit*,

JDK). Aún más, la propia Sun llamaba anteriormente a su J2SDK un JDK. Puede pensar en el término "JDK" como en una descripción de la clase de software que se utiliza para desarrollar y ejecutar software Java, mientras que "J2SDK" es simplemente el nombre de la versión particular de Sun de un JDK. Esta es la terminología que se utilizará en este capítulo.

El J2SDK es un producto de marca comercial. Es decir, incluso aunque Sun haga que las especificaciones del lenguaje de programación Java y la máquina virtual Java estén disponibles libremente, el producto real J2SDK es software registrado perteneciente a Sun. El código fuente para el J2SDK está disponible; sin embargo, no confunda esto con el software de código abierto, ya que la licencia Sun de código de la comunidad (*Sun Community Source License*, SCSL) bajo la que Sun ofrece el código ¡no es una licencia de código abierto! Para muchos de los lectores, por supuesto, esto no importará, y este libro adopta un enfoque práctico: a veces, el software comercial es simplemente una necesidad. Una gran cantidad de desarrolladores profesionales necesita utilizar J2SDK de Sun en sistemas Linux, por eso este capítulo está aquí para mostrarle cómo. No obstante, el resultado para usted es que para poder instalar este software debe aceptar una licencia de software (pero sin ningún coste). Examine esta licencia con detenimiento antes de aceptarla y realizar ninguno de los pasos comentados en este capítulo.

Nota

Véase el capítulo 2 para más información sobre el software de código abierto.

Java es un auténtico lenguaje de programación, y por eso puede no interesar por sí mismo a lectores que no sean ingenieros de software o programadores. Sin embargo, la inclusión del J2SDK como un ejemplo se debe principalmente a que permite mostrar la instalación de un paquete de software comercial que necesita extensas personalizaciones del entorno de usuario para funcionar adecuadamente. Es un gran modo de ilustrar algunas técnicas muy útiles, así que incluso los lectores no desarrolladores pueden desear leer esta sección.

Nota: Sólo los hechos, J2SDK de Sun Microsystems

Propósito: Proporciona un entorno de ejecución y herramientas de desarrollo para el lenguaje de programación Java.
Autores: Sun Microsystems, el Proyecto Blackdown
Sitio web: java.sun.com/j2se y www.blackdown.org
Descripción: El lenguaje de programación Java es la plataforma altamente adaptable de Sun para el desarrollo de muchos tipos de aplicaciones, incluidos

programas clientes con GUI, aplicaciones de servidor de empresas, y pequeños *applets* (aplicacioncitas) para la web.

Instalación de J2SDK

El primer paso, como siempre, es descargar el software. Puede encontrarse dentro de la sección "Products and APIs" de la página entrada al sitio web de Java de Sun: java.sun.com. Necesitarádescargar el Java 2 SDK para Linux. Ya que es un producto software comercializado, no hay disponible una opción de instalación a partir del código fuente. Por ello, es importante que se asegure de que el software se ejecutará en su distribución actual; la página de descarga presenta una lista con los requisitos que la distribución debe cumplir (que normalmente se reduce a si la distribución dispone de la versión necesaria del paquete de la biblioteca glibc de GNU). Dese cuenta de que aunque el código fuente está disponible para descargarlo a un precio nulo, la licencia no es una licencia de código abierto y compilar el código no es una hazaña pequeña; no es realmente una opción para usuarios finales como pueda serlo usted.

En el momento de la elaboración de este libro, la versión más reciente de J2SDK era la 1.4, por eso este ejemplo asume que va a instalar esa versión. En la práctica, sin embargo, es bastante habitual entre los desarrolladores Java conmutar entre distintas versiones del J2SDK, e incluso cambiar entre distintos JDK. Por ejemplo, pueden necesitar soportar una versión específica del J2SDK (que puede o no ser la última ofrecida por Sun). A veces los desarrolladores también necesitan cambiar a la implementación JDK de otro fabricante; IBM también dispone de una versión, y en ocasiones es útil durante la depuración del software escrito en Java el conmutar entre las versiones de IBM y Sun.

Estas necesidades implican que en cualquier momento dado un sistema puede necesitar tener instalado más de un JDK. Como se comentó en la sección "Decidir la ubicación" del capítulo 7, esta situación hace que sea más apropiado instalar el software en /opt en lugar de en /usr/local. Por ejemplo, un sistema podría tener instalados simultáneamente /opt/ibm-jdk-1.3 y /opt/sun-j2sdk-1.4. Sin embargo, hay más relacionado con Java que sólo el JDK; los desarrolladores y usuarios de software Java también necesitan una amplia variedad de bibliotecas Java (llamadas *Java Archive files* o archivos JAR) para poder utilizar realmente la mayoría del software. Por eso, este ejemplo lleva la técnica del directorio /opt un paso más allá: el software se instalará en /opt/java en lugar de simplemente en /opt. Esto se comentará en más detalle según avance la sección; por ahora, es suficiente decir que el directorio de instalación es /opt/java/sun-j2sd-1.4.

El siguiente paso es extraer físicamente el JDK. Realmente, el paquete se distribuye como un guión de shell. Esto puede parecer un poco extraño para aquellos lectores que nunca hayan visto este método; los usuarios preocupados por la segu-

ridad pueden incluso ser (¡con toda la justificación del mundo!) suspicaces ante paquetes que deban ser ejecutados para poder instalarlos. Sin embargo, este paquete es bastante inofensivo. Es un archivo de shell (*shell archive*, shar). Este formato es bastante similar en concepto a un archivo TAR, omnipresente en los sistemas Unix y que probablemente ya le resulte familiar. Sin embargo, un archivo shar está codificado realmente en un guión de shell, no como un archivo tar, que es un simple archivo con datos binarios. Hay en general dos motivos para utilizar un archivo shar en lugar de un archivo tar: uno es que ya que es un guión de shell, es simplemente un archivo de texto ASCII, y es más fácil enviarlo por correo electrónico que enviar un archivo binario.

El otro motivo para utilizar un archivo shar (y en el que está interesado Sun) es que ya que en el fondo el formato es un guión de shell, el empaquetador puede incluir un guión para que se ejecute cuando se extraiga el paquete. En el caso del JDK, Sun utiliza esta prestación para mostrar una copia de la licencia del software que se solicita al usuario que acepte antes de que el archivo shar continúe con el proceso de extracción. Por eso, cuando se desempaquete la distribución de JDK, se mostrará la licencia y se le pedirá que la acepte antes de proseguir.

En todo caso, el archivo de distribución de JDK puede extraerse simplemente mediante la ejecución del archivo. Sin embargo, el guión de shell extrae sus contenidos dentro del directorio actual, ¡por eso es importante desplazarse al directorio padre de la ubicación última deseada! Un potencial matiz adicional es que el JDK se extraerá dentro de un directorio llamado de una manera específica: j2sdk1.4. Si ya hay un directorio con ese nombre (que podría ser el caso si hay instalado más de un JDK), entonces sus contenidos se sobrescribirán, así que ¡compruebe este detalle con cuidado antes de proceder a la extracción! Por eso puede ser recomendable realizar la extracción en un directorio distinto del directorio destino, como /tmp. En cualquier caso, aquí están los pasos necesarios para extraer el J2SDK (asumiendo que la distribución del J2SDK se ha descargado como /tmp/[nombre de fichero]):

```
% mkdir /tmp/jdk; cd /tmp/jdk
% sh /tmp/[nombre de fichero]

% mkdir -p /opt/java
% mv j2sdk1.4 /opt/java/sun-j2sdk-1.4
```

Después de llevar a cabo el segundo paso, debe leer y aceptar la licencia, y entonces se extraerán los contenidos de la distribución.

El parámetro `-p` pasado al comando `mkdir` provoca que si el directorio ya existe, finalice en silencio en lugar de fallar notificando un error; consulte la página del manual sobre `mkdir` para obtener más información.

En este momento, la versión 1.4 del Java 2 SDK de Sun estará instalada en /opt/java/sun-j2sdk-1.4. El motivo para haber cambiado el nombre desde el nombre por defecto de j2sdk1.4 es porque se podría necesitar instalar al mismo tiempo más de una versión del J2SDK en el sistema. El cambio de este nombre facilita la administración de las versiones instaladas. Por ejemplo, la versión 1.3 de la implementación

de IBM del JDK podría instalarse junto a la versión de Sun en el directorio /opt/java/ibm-jdk-1.3. Este esquema de nombrado hace más fácil identificar qué software de qué fabricante está instalado en qué directorio.

Configuración de J2SDK

Extraer el J2SDK es la parte fácil; resulta algo más complicado conseguir que funcione correctamente. Esta sección comentará las técnicas para configurar J2SDK para su uso real. Hay dos objetivos principales: debería ser posible instalar múltiples versiones del JDK en el sistema al mismo tiempo sin que se produzcan conflictos, y debería ser rápido y fácil instalar y utilizar aplicaciones Java.

El directorio /opt/java

En la sección anterior, el paquete JDK se instaló en el directorio /opt/java/sun-jdk-1.4. Esta es una ligera desviación de la práctica habitual de instalar software directamente en /opt. (Por ejemplo, en el capítulo 11 el servidor HTTP Apache se instaló directamente como un subdirectorio de /opt.) El motivo para crear un subdirectorio completamente separado para el software Java es porque hay más componentes de software relacionados con Java aparte del J2SDK. El J2SDK es sólo el comienzo: una vez que está instalado, se necesitará instalar programas reales escritos en Java, y también se necesitará instalar bibliotecas Java de soporte.

En este sentido, el J2SDK (o cualquier JDK) es realmente su propio pequeño sistema operativo, con programas y bibliotecas que necesitan instalarse en él. Esto abarrotaría rápidamente el directorio /opt, por eso se ha creado el subdirectorio /opt/java. Funciona exactamente igual que /opt, excepto que sólo contiene archivos relacionados con Java. El directorio /opt/java, tal y como se usa en este libro, contiene los siguientes archivos y directorios:

- /opt/java/[directorio]: Subdirectorios aparte para cada JDK que se instale.
- /opt/java/packages: Un directorio que contiene todas las bibliotecas Java de soporte; archivos JAR y ZIP.
- /opt/java/jdk: Un enlace simbólico que apunta a la instalación JDK principal (a la instalación por defecto)

La utilización del directorio /opt/java es claramente opcional, y no se encontrará en ningún estándar ni especificación (como el FHS comentado en el capítulo 3 en la sección "Estructura del sistema de archivos"). Es una técnica que el autor del libro ha desarrollado a lo largo del tiempo para ayudar a un gran grupo de desarrolladores de Java que utilizaban estaciones de trabajo Linux en una importante compañía. El directorio /opt/java simplemente hace más conveniente gestionar las instalaciones de implementaciones de JDK de diversos fabricantes, aplicaciones y archivos de

soporte. Otros administradores pueden preferir otra solución (como colocar todos los archivos directamente en /opt); en el fondo, es una cuestión de gusto. El propósito de este libro es ofrecer ejemplos y prácticas recomendadas; debería tomar este material y comenzar a trabajar a partir de él, desarrollando sus propias soluciones particulares.

Instalación de múltiples JDK

Recuerde que uno de los objetivos de este ejemplo de instalación de Java es hacer que el uso del entorno de desarrollo Java y de sus aplicaciones sea realmente rápido y sencillo. Esto significa que los usuarios individuales no deberían tener que realizar cambios extensos en la configuración, editar muchos guiones de shell (¡ni siquiera uno!), ni cosas así. Sin embargo, esto se contrapone (como siempre sucede) a la necesidad de flexibilidad: si instala JDK de diversos fabricantes, entonces el usuario obviamente tendrá que escoger entre ellos.

El compromiso en este caso es el enlace simbólico /opt/java/jdk. Este enlace apunta a una instalación del JDK en particular que queda designada como JDK por defecto. Por ejemplo, un sistema dado podría tener instalados tres JDK distintos: el último J2SDK con calidad de producción de Sun, un JDK de IBM optimizado de cara a la velocidad, y el más reciente J2SDK con calidad beta de Sun para realizar pruebas. La mayor parte del tiempo, el desarrollador sólo querrá utilizar la última versión del JDK, por eso el enlace /opt/java/jdk debería apuntar a esa instalación. En este ejemplo, sólo hay instalado un JDK (concretamente, el J2SDK de Sun), por eso el enlace /opt/java/jdk debería apuntar a esa instalación, lo que puede lograrse mediante la ejecución de este comando:

```
% ln -s /opt/java/sun-j2sdk-1.4 /opt/java/jdk
```

Cuando se actualice un JDK (quizás cuando se reemplace la versión del J2SDK de calidad beta por el siguiente lanzamiento de producción), se puede instalar simplemente junto a los otros, y se puede cambiar el JDK por defecto actualizando el enlace simbólico /opt/java/jdk. Sin embargo, es vital recordar que esto sólo funcionará si los entornos de usuario realmente apuntan a /opt/java/jdk; obviamente, si los usuarios tienen instalaciones especificadas incrustadas directamente en sus entornos, entonces no servirá de nada modificar /opt/java/jdk. Por este motivo, todos los detalles del entorno de usuario descritos en las siguientes secciones han sido creados para hacer referencia a /opt/java/jdk en lugar de a un directorio de instalación específico.

Instalación de archivos JAR

La plataforma Java proporciona la capacidad de añadir bibliotecas de soporte para el uso de las aplicaciones. Estas bibliotecas están empaquetadas como archivos comprimidos en formato ZIP (que tienen la extensión .zip) o como archivos

JAR. Estas bibliotecas son similares en concepto a las bibliotecas dinámicas de Unix comentadas en el capítulo 7 y a los módulos de carga dinámica de Apache comentados en el capítulo 11.

Sin embargo, hay que hacer explícitamente disponibles estos archivos JAR a la máquina virtual Java. Dependiendo de la implementación del JVM, hay típicamente un par de maneras de conseguir esto: de la manera más habitual, deberá situar los archivos JAR en un directorio específico conocido por el JVM o bien añadir los nombres de los archivos JAR a una variable de entorno (llamada la variable CLASSPATH). Con cualquiera de estas dos maneras, rápidamente puede haber una explosión de archivos JAR, con versiones conflictivas. Esta confusión de archivos tiende a provocar errores y puede ser a veces muy difícil de depurar.

Este es el motivo para la existencia del directorio /opt/java/packages. Los administradores sitúan en este directorio archivos JAR y ZIP que deben estar disponibles para todo el sistema, y el entorno de usuario fijará automáticamente las variables CLASSPATH de los usuarios para incluir estos archivos. Añadir un nuevo archivo a /opt/java/packages tendrá efecto en los entornos de los usuarios la siguiente vez que entren en el sistema.

No obstante, un problema con este enfoque es que el directorio /opt/java debe pertenecer al usuario *root*; esto significa que los usuarios normales no podrán colocar sus archivos particulares es ese directorio. Tendrían que gestionar su variable CLASSPATH por sí mismos, lo que desafía el propósito de tener un directorio /opt/java/packages común. Para enfrentarse a este problema, los guiones de entorno del usuario que se explican en la siguiente sección soportan no sólo el directorio /opt/java/packages, sino también un directorio java/packages en el directorio local del usuario.

Los guiones funcionan del mismo modo para ambos directorios y son complementarios entre sí. Si desea más información, no tiene más que continuar con su lectura y avanzar hasta la siguiente sección.

Configuración del entorno de usuario

Los subdirectorios de /opt/java funcionan bien para la organización de los archivos, pero resultan inútiles a no ser que los entornos de los usuarios estén configurados correctamente. Esta sección mostrará y comentará los cambios y guiones necesarios para configurar los entornos de los usuarios para su acceso a la instalación de Java en el directorio /opt/java. En general, sólo hay tres cosas que se necesite hacer: fijar la variable CLASSPATH de los usuarios, añadir los programas Java a la variable PATH de los usuarios y fijar la variable de entorno JAVA_HOME.

Fijar CLASSPATH

CLASSPATH (variable de la ruta de clases) es un variable de entorno que indica a la máquina virtual Java dónde encontrar los archivos JAR. Java es un lenguaje de programación orientado a objetos, lo que significa que funciona con los conceptos

de clases como un modelo de sistema software. Estas clases son lo que se guarda dentro de los archivos JAR y ZIP, y por eso la ruta que contiene estas bibliotecas se conoce como CLASSPATH. CLASSPATH es simplemente una variable de entorno cuyo valor es una lista de directorios y archivos ZIP y JAR que contienen clases Java, con las entradas separadas entre sí mediante dos puntos.

En la sección anterior llamada "Instalación de archivos JAR", se creó el directorio /opt/java/packages como repositorio de archivos JAR y ZIP. Entonces, se debe añadir cada archivo que se encuentre en este directorio individualmente a la variable CLASSPATH.

Además, si se añade un nuevo archivo JAR al directorio (¡o se elimina!), CLASSPATH debería ajustarse automáticamente para que el administrador no se vea obligado a realizar el cambio manualmente. Este problema debería resultarle familiar: ¡es el problema de depósito de configuraciones, descrito de manera extensa en el capítulo 4!

El espíritu de Unix es no reinventar la rueda, por eso el modo más fácil de resolver este problema es utilizar las mismas técnicas empleadas por Red Hat tal y como se describen a lo largo del capítulo 4. De hecho, el problema se resuelve de manera muy simple; el siguiente fragmento de guión shell escrito con la sintaxis `sh` (en lugar de la sintaxis `csh`) configurará correctamente la variable CLASSPATH:

```
set CLASSPATH=.
for jar in /opt/java/packages/*; do
    set CLASSPATH=${CLASSPATH}:${jar}
done
export CLASSPATH
```

El fragmento anterior simplemente encuentra cada archivo JAR que esté en el directorio /opt/java/packages y lo añade a CLASSPATH. La variable CLASSPATH comienza como un simple punto ("."). Recuerde que una ruta con el valor "." es la abreviatura de Unix para el directorio de trabajo actual, y de este modo el fragmento incluye el directorio de trabajo actual como la primera entrada en la variable CLASSPATH. Esto resulta útil porque permite a los programas JAVA que se ejecutan desde la línea de comandos encontrar las clases que puedan estar es el directorio desde el que se ejecutaron; eso permite que se localicen las denominadas clases Java anónimas. (No se preocupe si esto no tiene sentido, no es importante saber cómo funciona Java. Sólo asegúrese de comprender cómo se está fijando el valor de la variable.) Un detalle final que hay que reseñar es que el fragmento de código añade cualquier archivo o subdirectorio que esté dentro de /opt/java/packages a la variable CLASSPATH; por eso, las bibliotecas Java que estén preparadas como directorios planos en lugar de cómo archivos JAR o ZIP también funcionarán con esta técnica.

Recuerde que fijar la variables CLASSPATH para incluir archivos del directorio /opt/java/packages sólo supone la mitad del problema: los usuarios aún necesitarán modificar ellos mismos su CLASSPATH para incluir los archivos específicos que necesiten. Sin embargo, el siguiente ejemplo similar sirve para añadir archivos desde los directorios personales de los usuarios:

```
set CLASSPATH=.
if [ -d ${HOME}/java/packages ]; then
    for jar in ${HOME}/java/packages/*; do
        set CLASSPATH=${CLASSPATH}:${jar}
    done
fi
export CLASSPATH
```

Este fragmento de código es casi idéntico al caso anterior, excepto que realiza una comprobación para asegurarse de que el usuario realmente dispone de un directorio ~/java/packages antes de intentar añadir archivos procedentes de él. (El guión puede asumir simplemente que existe /opt/java/packages, porque es un directorio global y gestionado por el administrador del sistema.) Otra cosa en la que hay que fijarse es en que la primera línea reinicia globalmente el valor de CLASSPATH; si pretende combinar estos dos guiones, ¡asegúrese de eliminar esa línea!

Fijar PATH y JAVA_HOME

La variable PATH (ruta) la utiliza la shell para encontrar los programas que puede ejecutar; la variable JAVA_HOME simplemente necesita apuntar hacia el directorio en que está instalado el J2SDK. Estas variables se fijan de manera muy simple; sin embargo, como se comentó con anterioridad, si se han fijado apuntando directamente hacia una instalación de un JDK en particular (como /opt/java/sun-jdk-1.4), entonces el guión tendrá que ser modificado cada vez que se actualice el sistema para utilizar un JDK nuevo.

Hay una solución bastante simple: basta fijar estas variables para que apunten hacia el enlace simbólico /opt/java/jdk en lugar de a una instalación específica. Aunque esto funcionará perfectamente bien, los usuarios que necesiten cambiar a un JDK diferente (uno distinto del indicado mediante /opt/java/jdk) tendrán que volver a fijar manualmente sus variables PATH. Una solución más flexible es ofrecer al usuario una oportunidad para anular la selección por defecto, mediante el procedimiento de apuntar a una instalación específica. El siguiente fragmento de guión muestra este método:

```
if "X${JAVA_HOME}" == "X"; then
    set JAVA_HOME=/opt/java/jdk
fi

set PATH=${PATH}:${JAVA_HOME}/bin
export PATH JAVA_HOME
```

Este fragmento sencillamente fija la variable JAVA_HOME al valor por defecto de /opt/java/jdk, pero sólo si la variable no se ha fijado antes, y después añade una entrada en PATH relativa a JAVA_HOME. Para cambiar a una versión específica del J2SDK (o de otro JDK), el usuario sólo tiene que fijar la variable JAVA_HOME y volver a ejecutar el guión precedente. Tomará la variable JAVA_HOME particular y configurará el entorno de manera apropiada.

Nota: Comprensión de los JDK

La instalación de ejemplo de un JDK en esta sección se basa en una suposición importante. Los ejemplos (en especial los guiones de shell) suponen que el JDK que se usará funciona del mismo modo que la referencia J2SDK de Sun. La mayoría de los JDK habituales (como la versión de IBM) se comportan del mismo modo que la de Sun; sin embargo, es posible que algunas no lo hagan así. Vigile esas diferencias en el comportamiento y esté preparado para adaptar este ejemplo al JDK específico que vaya a instalar.

Juntarlo todo

Los fragmentos de guión mostrados en este capítulo realizan el 90 por ciento de lo que se necesita hacer; el resto es simplemente integración. El listado 13.1 es una versión completa y perfectamente funcional de los fragmentos de guión. Este listado podría copiarse en un archivo e incluirse en la configuración del entorno de shell para los usuarios. (La siguiente sección describe cómo llevar a cabo esto realmente en cada una de las distribuciones de muestra.) Este archivo está de nuevo en la sintaxis shell `sh`; para encontrar un archivo equivalente con la sintaxis `csh` para los usuarios de shells `csh` y `tcsh`, consulte el CD-ROM que se incluye junto con el libro.

Listado 13.1. Guión de configuración del entorno de usuario para J2SDK.

```
#!/bin/sh

# fija JAVA_HOME al valor por defecto, a no ser que ya esté fijado
if [[ "X"${JAVA_HOME} == "X" ]]; then
    JAVA_HOME=/opt/java/jdk
fi

# añade el directorio bin de JDK a PATH (a la ruta por defecto)
PATH=${JAVA_HOME}/bin:${PATH}

# fija CLASSPATH; con el valor inicial de "." (el directorio de trabajo
actual)
CLASSPATH=.
for jar in /opt/java/packages/*; do
    if [[ "${jar}" == "/opt/java/packages/*" ]]; then
        break;
    fi
    CLASSPATH=${CLASSPATH}:${jar}
done

# añade cualquier JAR que esté en el propio directorio de paquetes java del
usuario
```

```
if [ -d ${HOME}/java/packages ]; then
    for jar in ${HOME}/java/packages/*; do
        if [[ "${jar}" == "${HOME}/java/packages/*" ]]; then
            break;
        fi
        CLASSPATH=${CLASSPATH}:${jar}
        done
fi

export PATH JAVA_HOME CLASSPATH
```

Instalación de J2SDK en diferentes distribuciones

Esta sección comentará cómo adaptar el ejemplo genérico mostrado en las secciones anteriores a distribuciones específicas. Ya que instalar el J2SDK consiste generalmente en extraer los archivos y modificar los entornos de los usuarios, adaptar la instalación para una distribución es tan simple (¡o tan complicado!) como enganchar el listado 13.1 en los guiones globales de entrada al sistema del usuario.

Instalación en Red Hat Linux

Como se comentó en el capítulo 4, Red Hat Linux hace un uso extensivo de la técnica del directorio de deposito de archivos de configuración. El directorio de depósito para la personalización del entorno shell de usuario es /etc/profile.d. Recuerde que los guiones de entrada al sistema de mayor nivel simplemente invocan a los archivos que se encuentran en /etc/profile.d (la shell `bash` invoca a los archivos que acaben en `.sh`, y la shell `tcsh` a los archivos que acaben en `.csh`.)

Nota

Véase la sección "Entorno de usuario" del capítulo 4 para obtener más detalles sobre la configuración del entorno de usuario de Red Hat Linux.

Personalizar la instalación de J2SDK en Red Hat Linux es entonces tan fácil como colocar el guión shell del listado 13.1 en el directorio /etc/profile.d; una buena elección podría ser llamar al archivo /etc/profile.d/java.sh. Para los usuarios de `tcsh`, la versión del listado 13.1 con sintaxis `csh` podría colocarse en /etc/profile.d/java.csh. (Esta versión del listado 13.1 puede encontrarse en el CD-ROM que acompaña al libro.)

Sin embargo, una advertencia importante es que hay que tener cuidado con el modo en que los usuarios anulan individualmente la ubicación del JDK por defecto.

El guión del listado 13.1 realiza comprobaciones para ver si el usuario ya ha fijado la variable JAVA_HOME y sólo la fija con su valor por defecto si el usuario no la ha fijado previamente. Sin embargo, ya que este guión se invocará a través del proceso de entrada al sistema global antes de que se invoquen a los guiones personales del usuario, el usuario no tendrá una oportunidad de fijar su propio valor para la variables JAVA_HOME. Afortunadamente, todo lo que tiene que hacer el usuario es simplemente entrar en el sistema, fijar la variable JAVA_HOME manualmente y volver a ejecutar el guión /etc/profile.d/java.sh para reconfigurar la shell (mediante un comando como `. /etc/profile.d/java.sh` para los usuarios de la shell `bash`).

Instalación en Slackware Linux

Como se comentó en la sección "Personalización del entorno de usuario" del capítulo 5, Slackware Linux tiene un mecanismo bastante simple para encargarse de los guiones de entrada al sistema. Por una parte, esto facilita el realizar la personalización manualmente, pero por otra dificulta la misma personalización de manera automática. Es decir, añadir el listado 13.1 al proceso de entrada en un sistema Slackware Linux no es tan sencillo como depositar un archivo en un directorio (como sí sucede con Red Hat Linux, como se comentó con anterioridad).

En su lugar, el administrador tendrá que modificar los guiones de entrada globales para añadir el soporte a Java en los entornos de los usuarios. La primera solución que se puede sugerir es simplemente copiar el guión del listado 13.1 y pegarlo directamente dentro del archivo /etc/bashrc (después de borrar la primera línea, `#!/bin/sh`). Esto funcionará bien, pero entonces el guión global será mucho más grande. Una solución mejor podría ser copiar el contenido del listado 13.1 dentro del directorio /etc (por ejemplo, como un archivo llamado /etc/javarc.sh) y después añadir una referencia a ese archivo desde /etc/bashrc, como muestra este ejemplo:

```
if [ -f /etc/javarc.sh ]; then
    ./etc/javarc.sh
fi
```

El inconveniente de este enfoque es que si se van a instalar muchos archivos de este tipo (además del único fichero para el J2SDK), entonces el directorio /etc estará abarrotado. Por eso, es un compromiso: un único guión de shell monolítico, o múltiples archivos que pueden añadir una gran cantidad de engorro. Todo se reduce a una cuestión de preferencias por parte del administrador.

Instalación en Debian GNU/Linux

El entorno shell de entrada de los usuarios en Debian GNU/Linux se prepara de un modo similar al de Slackware Linux. Es decir, los archivos /etc/profile, /etc/bashrc, /etc/csh.cshrc y /etc/csh.login se utilizan para configurar las diversas

shells de los usuarios. En Debian GNU/Linux estos guiones son similares, pero tienen contenidos distintos de los de sus contrapartidas en Slackware Linux. Como resultado, conectar la instalación de J2SDK descrita en este capítulo se lleva a cabo de manera muy parecida al proceso descrito en la sección anterior sobre Slackware Linux, aunque habrá algunas diferencias menores. Si ha llegado hasta esta parte del libro, no debería tener ningún problema al enfrentarse con esto.

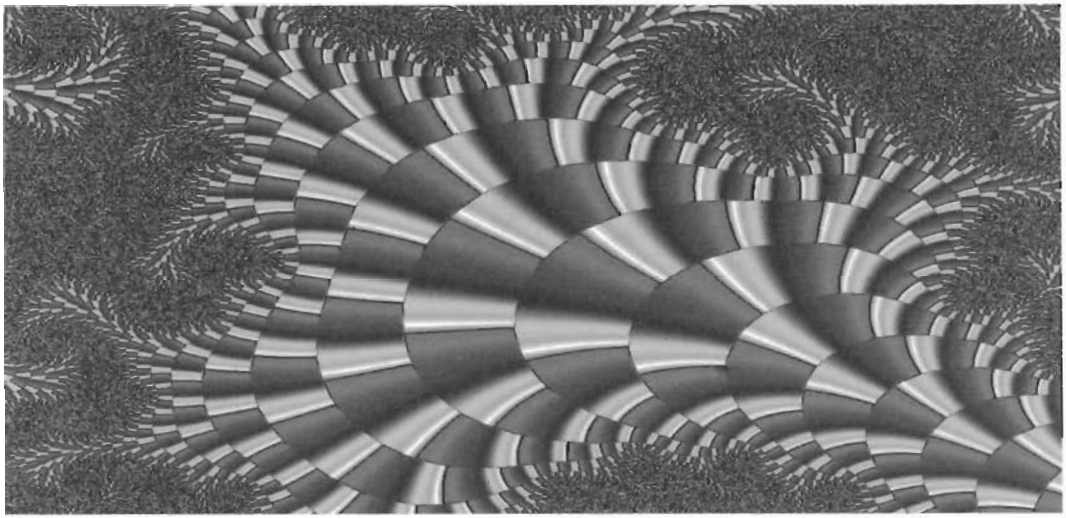
Resumen

Ahora ya ha visto cómo instalar el J2SDK de Sun Microsystems. Esto le habrá enseñado tres cosas. En primer lugar, obviamente habrá aprendido a instalar un JDK, que le permite desarrollar software Java (si eso es lo que quiere). En segundo lugar, habrá aprendido en un sentido más general cómo configurar un conjunto de software similar al J2SDK; esto se reduce esencialmente a gestionar variables de entorno y las instalaciones de software. En tercer lugar, también habrá aprendido cómo aplicar las técnicas que ya conocía para resolver un nuevo problema. Hablo específicamente sobre cómo el directorio /opt/java le ayuda a administrar sus instalaciones de software Java replicando el comportamiento del directorio /opt estándar, al que ya estaba acostumbrado.

Es de esperar que ahora esté comenzando a ver cómo todos estos pequeños trucos y técnicas que ha comentado el libro se utilizan en conjunto para ayudarle a resolver problemas reales. Ahora, armado con el conocimiento sobre una variedad de instalaciones de software distintas, debería ser capaz de aplicar estos pequeños trucos y técnicas a cualquier problema nuevo con el que se encuentre (del mismo modo en que aplicó y adaptó los principios de /opt al nuevo directorio /opt/java). Este capítulo concluye la tercera parte de este libro, que trata acerca de la instalación de software. La cuarta parte comentará varios casos de estudio de sistemas Linux prácticos y reales. Esta cuarta parte se apoyará en el material de las demás partes, y en especial en la tercera parte. Si ya ha acabado de leer esta parte completamente (desde el capítulo 7 hasta este capítulo), felicidades, ¡ya está preparado para ver algunos casos de estudio reales!

Parte IV

Casos de estudio



14
Construcción de un sistema de escritorio

Este capítulo le muestra cómo instalar y configurar un sistema válido para el uso personal. Como con todos los casos de estudio, este sistema comienza a partir de una distribución Red Hat Linux 7.3 estándar, y se instala software adicional según resulte necesario. En primer lugar se describirán someramente los objetivos y las aplicaciones de esta configuración, y después se describirá en detalle el proceso real de construcción.

Comienzo

Esta sección delinea los objetivos y motivaciones de este caso de estudio. Describe el tipo de usuario que con más probabilidad estará interesado en este entorno y define algunos términos básicos. Si lo desea puede limitarse a hojear esta sección, pero asegúrese de que al menos la lee de esta manera, ya que encontrará que la información suministrada aquí le resultará útil cuando lea el caso de estudio real.

¿A quién se dirige este ejemplo?

El objetivo de este capítulo es mostrar cómo puede cubrir un sistema Linux las necesidades de un usuario típico habitual. No hay realmente nada en este capítulo que no conozca ya, ningún truco mágico de instalación ni configuración vudú. Todo el software puede instalarse y configurarse mediante técnicas parecidas a las

comentadas en los capítulos 7 a 13; en ocasiones, el proceso será tan simple como instalar el paquete RPM apropiado, y en otras puede que tenga que instalar aplicaciones a partir del código fuente u otro formato de empaquetado.

Puede ver este capítulo como una especie de popurrí de aplicaciones útiles. Este capítulo no entrará en detalles sobre los matices de la instalación o la configuración del software, ya que ya sabe cómo llevar esto a cabo. En su lugar, se centra en indicarle la dirección correcta para seleccionar e instalar el software que necesita para cumplir sus objetivos.

El objetivo último, en realidad, es producir un escritorio repleto de prestaciones. Frecuentemente escuchará cómo los "expertos" sobre tecnología dicen que los sistemas Linux no son buenos sistemas de escritorio. Bien, después de este capítulo será capaz de tomar una decisión sobre este tema por sí mismo. El sistema descrito en este capítulo tiene todo lo que yo necesito: si sus necesidades son similares a las mías, entonces (como yo) puede que nunca vuelva a utilizar un sistema operativo que no sea Linux.

Por supuesto, quizás sus necesidades sean diferentes. Quizás lea este capítulo, experimente con algunas de las aplicaciones y decida que, al fin y al cabo, Linux no es viable en su caso como un sistema de escritorio. Ese es, en definitiva, el objetivo fundamental: educarle lo suficiente como para que pueda tomar esa decisión por sí mismo, y no tome la palabra de algunos "expertos" como si fuera la suya, sin sopesarla.

Definición de un escritorio

Cualquier problema puede resolverse con facilidad si se define el problema de manera suficientemente minuciosa. Por eso, necesita definir el concepto de un sistema de escritorio con suficiente precisión para poder crear uno. Para los propósitos de este ejemplo, un sistema de escritorio es simplemente un ordenador que se utiliza para las tareas informáticas rutinarias, de todos los días.

Las tareas en las que se centra este capítulo aparecen en la lista siguiente; si un sistema puede soportar estas tareas básicas y comunes, entonces considérelo un sistema de escritorio.

- Productividad ofimática (procesador de textos, hoja de cálculo y demás)
- Navegación web
- Soporte hardware (dispositivos USB, dispositivos de almacenamiento y demás)
- Multimedia y entretenimiento (fotografía digital, juegos y demás)
- Conectividad (acceso a su escritorio de manera remota, acceso a otros sistemas)

Estos puntos están bien, pero ¿con qué se va a comparar este sistema de escritorio? Bueno, hay otras tres clases generales de sistemas informáticos:

- Dispositivos embarcados, como asistentes personales digitales (PDA)
- Servidores
- Estaciones de trabajo

Este libro no comenta sistemas Linux embarcados, así que no tendrá que considerar esa clase. Además, un servidor es bastante simple: es una máquina dedicada a proporcionar alguna característica o funcionalidad específica (es decir, un servicio) a otras máquinas que estén en una red. Esto nos deja con la estación de trabajo, cuyo límite es un poco más borroso.

Mucha gente emplea los términos "escritorio" y "estación de trabajo" de manera intercambiable, pero resulta útil establecer una barrera entre ambos. Acabo de definir un escritorio. Una estación de trabajo suele ser un ordenador personal de alto rendimiento utilizado para una tarea específica en lugar de para un conjunto general de tareas. Por ejemplo, la máquina que utiliza un arquitecto o un ingeniero para realizar un trabajo de diseño asistido por ordenador sería una estación de trabajo; del mismo modo, el sistema que utiliza un ingeniero de software para desarrollar una aplicación sería una estación de trabajo, al igual que lo sería el ordenador que utilizaría un científico investigador para ejecutar un programa de proceso numérico intensivo.

Cada uno de estos ejemplos es de un ordenador utilizado para una tarea específica con gran demanda de rendimiento. La construcción de una estación de trabajo como estas es el tema del caso de estudio que podrá encontrar en el capítulo 15; este capítulo se centra en configuraciones que es más probable encontrar en una casa u oficina que en un departamento de ingeniería.

Contenidos de este capítulo

Ahora que he definido los objetivos y delineado lo que cubriré en este capítulo, es hora de hablar sobre cómo será el enfoque adoptado para cumplir estos objetivos. Esta sección describe el tipo de información que encontrará en este capítulo.

Recuerde que este capítulo asume que ya está familiarizado con el material de la tercera parte, que comentaba los mecanismos de instalación y configuración de software. Este capítulo le guía en las direcciones correctas, para proporcionarle alguna idea sobre cómo aplicar esos mecanismos que ya ha aprendido.

Con este fin, este capítulo se centra en el software. Se comentan varias categorías de software, y cada categoría cita paquetes de software populares y funcionales que satisfacen los objetivos de la categoría. Los propios paquetes de software se describen de forma muy breve, se comenta su aplicabilidad y se mencionan las alternativas cuando sea apropiado. También encontrará información sobre las peculiaridades de estos paquetes, como potenciales dificultades comunes de su configuración, restricciones de licencias y demás. Finalmente, escogeré un paquete para el caso de estudio y explicaré la razón de mi elección.

Una vez que haya leído la sección correspondiente a una categoría de software dada, tendrá una comprensión general de las opciones disponibles en esa categoría,

algunas de sus limitaciones y un poco de perspectiva sobre cualquier dificultad que pueda encontrar. En otras palabras, tendrá una idea del estado del arte para esa categoría. Esta comprensión le permitirá aplicar las habilidades obtenidas en las tres partes previas. Será capaz de adoptar las elecciones presentadas en este capítulo literalmente para duplicar la configuración, o podrá realizar la elección razonada de una alternativa. Para comenzar, la siguiente sección describirá los conceptos básicos del caso de estudio del sistema de escritorio. Aprenderá cuales son los objetivos y requisitos del sistema y después leerá cómo se instaló el sistema básico Red Hat Linux 7.3 como núcleo de ese sistema.

Construcción del sistema

Obviamente, el primer paso en la construcción de cualquier sistema, este caso de estudio incluido, es instalar la distribución base. Como todos los casos de estudio, este comienza a partir de una instalación estándar de Red Hat Linux 7.3. Esta sección describe algunos detalles básicos sobre cómo se configuró inicialmente el sistema.

Esta sección se centra en la instalación de la distribución Red Hat Linux estándar. No se centra en explicar el sistema al completo en detalle; este es, después de todo, el enfoque tomado para el resto del capítulo. En otras palabras, si desea duplicar este caso de estudio, puede comenzar con la instalación de Red Hat Linux una vez que haya leído esta sección, pero es una buena idea leer en primer lugar el capítulo entero, de modo que pueda obtener una imagen global de cómo duplicar el sistema.

Elección de los paquetes

Como se comentó en el capítulo 4, Red Hat Linux tiene varias clases de instalación entre las que puede escoger el usuario cuando se instale el sistema. Estas clases son realmente simples listas de paquetes RPM que juntas forman una configuración típica para un uso particular. Por ejemplo, la configuración de servidor contiene paquetes como el servidor HTTP Apache, software NFS y demás.

La clase de instalación seleccionada para este caso de estudio fue la estación de trabajo KDE. Como leerá más adelante en este mismo capítulo, para este caso de estudio se escogió KDE por delante de GNOME, aunque nada le impide realizar una elección diferente. No se personalizó más la lista de paquetes durante la fase de instalación, aunque hay partes del resto de este capítulo que comentan los añadidos a la configuración básica.

Detalles de la configuración

Durante el proceso de instalación, el instalador de Red Hat (llamado anaconda) le solicita una gran variedad de información de configuración. No se hizo nada

fuera de lo ordinario en este caso de estudio; los valores por defecto se configuraron como esperaría que se hiciera para una estación de trabajo. Por ejemplo, se configuró la red para utilizar el protocolo DHCP (*Dynamic Host Configuration Protocol*) y se crearon las particiones del disco siguiendo una línea típica, con /boot, /usr, y / home en distintas particiones.

Nota

Lea la segunda parte del libro, o consulte su referencia favorita sobre Red Hat si quiere más información sobre la configuración del sistema durante la instalación.

Después de la configuración, no se modificó nada, excepto lo relativo a los procedimientos de seguridad básicos comentados en el capítulo 4 (como inhabilitar los servicios inetd innecesarios). Hay que notar que el cortafuego incluido junto con Red Hat Linux se dejó habilitado, con el nivel de seguridad fijado como alto. Sólo se dejó expuesto el puerto para el protocolo SSH (*Secure Shell*), de modo que se pueda emplear OpenSSH para el acceso remoto (posibilidad explorada más adelante en la sección "Acceso remoto al escritorio"). Esto resulta importante de cara a la seguridad (incluso si su red dispone de un cortafuegos independiente como el que se comentará en el capítulo 16), y se hará referencia a ello ocasionalmente a lo largo del resto de este capítulo.

Como puede ver, no hay mucho que hacer con respecto a la configuración de la instalación Red Hat Linux básica empleada para este caso de estudio. Este, por supuesto, fue en primer lugar el motivo de escoger Red Hat: hace mucho o la mayor parte del trabajo en su lugar. En el resto de este capítulo, leerá sobre los motivos por los que se tomó cada decisión, y sobre cualquier advertencia o riesgo que deba tener en cuenta si reproduce este caso de estudio. La siguiente sección profundiza un poco más y comienza a delinear el software adicional que se instaló.

Uso de aplicaciones de escritorio y productividad

La primera cosa que se encontrará cuando use cualquier sistema operativo es la interfaz de usuario. En los sistemas basados en Linux, la interfaz de usuario suele ser el sistema de ventanas X (típicamente personificado como XFree86), y un gestor de escritorio o ventanas. Después del entorno de escritorio, muchos usuarios consideran que el entorno de "productividad ofimática" y herramientas similares son el siguiente software más vital. Esta sección describe el gestor de escritorio X y un conjunto de herramientas de productividad importante para el sistema de este caso de estudio.

Entorno de escritorio

El entorno de escritorio es probablemente la parte más inmediata y por ello más global de un sistema operativo con la que el usuario interactúa. Por ello, escoger el entorno de escritorio para este caso de estudio define el carácter de ese entorno.

Por otro lado, escoger un entorno de usuario típicamente no define el software que podrá o no utilizar en su sistema. Por ejemplo, escoger el sistema KDE no le impide utilizar programas escritos para funcionar con el entorno GNOME. Desde esta perspectiva, entonces, la elección de un entorno de escritorio no afecta mucho a la selección de software que podrá emplear.

En otras palabras, la elección del entorno de escritorio es en gran medida una simple cuestión de gusto. Puede tener algún impacto en la manera de trabajar, pero normalmente no impide que utilice un programa en particular. El peor caso probablemente sea el consumo excesivo de memoria. Por ejemplo, si normalmente utiliza GNOME pero ejecuta un programa de KDE, ese programa puede necesitar cargar una gran variedad de bibliotecas compartidas relacionadas con KDE que sólo serán usadas por ese programa; normalmente estas bibliotecas estarían compartidas entre muchos programas KDE, pero no serán utilizadas por los programas GNOME.

La elección: KDE

Fíjese en los siguientes detalles sobre el entorno de escritorio K (KDE):

- **Ubicación:** www.kde.org
- **Versión:** 3.0
- **Paquetes:** Muchos paquetes; incluido junto con Red Hat 7.3
- **Descripción:** Un entorno de escritorio X de código abierto, completo, con todas las características y herramientas de productividad relacionadas.

Este caso de estudio se configuró para utilizar KDE. KDE se instaló durante la instalación de la distribución de Red Hat Linux (en lugar de después de este proceso) simplemente mediante la selección del conjunto de alto nivel de paquetes de KDE. Con alguna excepción, cada uno de los demás programas mencionados en este capítulo funcionarán bien junto con KDE, incluso si están dirigidos a un entorno diferente. En pocas palabras, utilice el entorno de escritorio que desee. Si descubre que utiliza muchos programas procedentes de otro entorno (como por ejemplo si utiliza muchos programas GNOME en el caso de que sea un usuario de KDE), entonces podría considerar cambiarse a otro entorno, pero al fin y al cabo no importa demasiado. Existe una excelente documentación electrónica sobre cada uno de los entornos principales, así que realmente es bastante complicado equivocarse.

Las alternativas: GNOME, GNUStep, Sawfish, BlackBox, y FVWM

Quizás sea GNOME el competidor más popular de KDE. Tanto KDE como GNOME han sido mencionados en este libro varias veces y están bien documenta-

dos en muchas otras partes. Sin embargo, realmente hay un número significativo de otros entornos de escritorio y gestores de ventanas que son menos conocidos que estos dos "gigantes". La tabla 14.1 recoge una lista de varios de estos entornos, y ofrece los URL hacia sitios en que encontrar información sobre ellos. Algunos de estos entornos son bastante interesantes, así que puede que desee probarlos cuando vaya a tomar su propia decisión.

Tabla 14.1. Entornos de escritorio y gestores de ventanas.

Entorno	Descripción	URL
GNUStep	Implementación de código abierto de OpenStep	www.gnustep.org
Sawfish	Guionizable mediante el lenguaje LISP	sawmill.sourceforge.net
BlackBox	Un gestor de ventanas minimalista	blackbox.alug.org
FVWM	Un popular gestor de ventanas clásico	www.fvwm.org

La tabla 14.1 no es en absoluto exhaustiva. Una simple búsqueda en su motor web de búsqueda favorito revelará cantidades ingentes de gestores de ventanas X. Debería darse una vuelta por la web y echar un vistazo para ver si algo le llama la atención.

Por lo menos, si siente que esta sección le ha restado importancia al asunto del entorno de escritorio, puede que tenga razón. Este libro trata sobre la distribución, ajuste y personalización de su sistema. Hoy en día, cada distribución que merezca la pena incluye paquetes de varios gestores de ventanas y seleccionar un gestor de ventanas es tan simple como utilizar su herramienta de gestión de paquetes (como RPM o apt-get) e instalar los archivos correctos. Emplear mucho tiempo en algo tan manido parece un poco una pérdida de tiempo, en especial cuando no importa demasiado. Por eso, ¡simplemente escoja lo que quiera y siga adelante!

Productividad ofimática

Muchos usuarios son fieles seguidores de sus suites de oficina, y ese tipo de software se ha convertido en algo básico de la informática moderna, por supuesto. Hay disponibles varias de estas suites para los sistemas Linux. Esta sección describe algunas de ellas.

La elección: OpenOffice

Escogí la suite OpenOffice para este caso de estudio principalmente por su extenso soporte para la importación y exportación de archivos a formatos de otras suites ofimáticas. Esta habilidad se vuelve algo crucial cuando necesita interactuar con otra gente que puede no estar usando la misma suite que usted (¡como cuando

está escribiendo un libro sobre el ajuste y la personalización de un sistema Linux!). Sin embargo, si esto no es importante para el usuario, puede utilizar alguna de las alternativas propuestas más adelante. Aquí están los detalles sobre OpenOffice:

- **Ubicación:** www.openoffice.org
- **Versión:** 1.0
- **Paquetes:** OOo_1.0.0_LinuxIntel_install.tar.gz
- **Descripción:** Una suite de productividad ofimática bastante completa de código abierto.

Instalación de OpenOffice

OpenOffice se distribuye desde www.OpenOffice.org como un archivo *tarball* (un archivo .tar.gz). Ya que no es un RPM, no será capaz de utilizar esa herramienta para instalar y administrar el programa. Aún más, esto significa que tiene que instalar el software por sí mismo de manera manual. OpenOffice es un paquete de software bastante grande, y por eso el archivo *tarball* incluye un guión de instalación, incluso para las distribuciones binarias.

En última instancia, puede tratar a OpenOffice del mismo modo en que trató al kit de desarrollo de Java de Sun, tal y como se comentó en el capítulo 13. Es decir, ambas son distribuciones preconstruidas de grandes proyectos de software en formato .tar.gz. Esencialmente sólo tiene que extraerlos, aunque en el caso de OpenOffice tenga que ejecutar después un guión de instalación. La mayoría de las técnicas sobre las que leyó en el capítulo 13 también se aplican a OpenOffice, puesto que necesita actualizar variables PATH y otras parecidas.

Sin embargo, hay una cosa que necesitará tener en mente mientras instale OpenOffice, que es que desafortunadamente la instalación no es realmente multiusuario. De manera más específica, el usuario que ejecuta OpenOffice tiene que poseer los archivos de la instalación. Esto hace que no resulte práctico instalar una única copia (como en /usr/local u /opt) para que la compartan todos los usuarios. Cada usuario debe disponer de su propia copia, lo que puede redundar en un desperdicio de espacio de disco bastante importante. (Sin embargo, parece muy probable que en una versión futura se añada el soporte a este tipo instalaciones multiusuario.)

No obstante, una vez que ha conseguido sobreponerse a estas dificultades, OpenOffice resulta una aplicación muy capaz. Debería consultar el sitio web de OpenOffice que se encuentra en www.openoffice.org en busca de más información.

Las alternativas: Koffice, Abiword, HancomOffice, Applixware, y Wordperfect Office

OpenOffice no es el único conjunto de aplicaciones para oficina disponible. La tabla 14.2 incluye una lista de otras suites ofimáticas y de programas de productividad (aunque, por supuesto, la tabla 14.2 no representa una lista exclusiva). Algunos

de ellas estarán disponibles como paquetes RPM para su instalación (lo que hace que la configuración resulte más sencilla), mientras que otras utilizarán otros formatos de empaquetado. En realidad, no hay mucho más que decir; la instalación de estos paquetes variará desde la simple instalación de paquetes RPM a procedimientos más complicados parecidos a los descritos en los capítulos 8 a 13. También podrá obtener paquetes con código fuente para algunos de ellos y construir así los programas por sí mismo.

Tabla 14.2. Aplicaciones de productividad ofimática.

Aplicación	URL	Notas
KOffice	www.koffice.org	La suite de oficina de KDE; incluida junto con Red Hat Linux
Abiword	www.abisource.com	Otro procesador de texto de código abierto
HancomOffice	en.hancom.com/index.html	Una suite de oficina comercial
Applixware	www.vistasource.com/products/axware	Otra suite de oficina comercial
Wordperfect Office	linux.corel.com	Otra suite de oficina comercial

Finanzas personales

El software de finanzas personales ayuda a su usuario a administrar ingresos, cuentas corrientes, cuentas de inversión y demás. El software de este tipo es muy popular entre los usuarios de ordenadores, y la competición en esta área es feroz (al menos en el mundo Windows). Hoy en día, hay disponibles unos pocos paquetes de software de código abierto para las finanzas personales, aunque no haya tantos como en el mundo Windows.

La elección: GNUCash

GNUCash es el software de finanzas personales elegido para este caso de estudio. GNUCash es un programa de finanzas personales bastante sencillo y funcional, y debería resultar de muy fácil aprendizaje para cualquiera que haya usado software similar antes. De manera notable, GNUCash también soporta diversos formatos estándar para datos financieros personales, lo que le proporciona un cierto grado de interacción con las finanzas electrónicas y vía web. Red Hat Linux 7.3 incluye un paquete RPM para GNUCash, lo que hace que su instalación sea trivial. Aquí hay algunos detalles sobre GNUCash:

- **Ubicación:** www.gnucash.org
- **Versión:** 1.6.6

- **Paquetes:** gnucash-1.6.6-3.i386.rpm
- **Descripción:** Un gestor personal de finanzas.

Una cosa de la que hay que ser consciente es que GNUCash utiliza el conjunto de componentes X GTK+. Esto significa que mientras que funciona bien junto con KDE, los dos no están realmente integrados. Probablemente la mayor preocupación que surja sea que GNUCash podría consumir más memoria de la que consumiría si se estuviera utilizando GNOME (que también está basado en GTK+), ya que serían capaces de compartir memoria para los conjuntos de componentes. Aún así, no es probable que suponga un problema a no ser que tenga limitaciones de memoria.

Si quiere disponer de documentación, manuales de usuario y demás, visite la página web de GNUCash en el sitio www.gnucash.org.

Las alternativas: Kapital

GNUCash parece ser la única aplicación de finanzas personales de código abierto con calidad de producción. Sin embargo, existe otra alternativa: la aplicación Kapital producida por TheKompany. Kapital no es de código abierto, y de hecho ni siquiera es de coste nulo: tendrá que adquirir la licencia de TheKompany. Sin embargo, muchos usuarios la encuentran superior a GNUCash, así que puede merecer la pena probarla si no está satisfecho con las prestaciones de GNUCash.

Puede encontrar más información sobre Kapital en www.thekompany.com/products/kapital/.

Una vez que haya escogido un gestor de finanzas personales (si desea o necesita uno), puede continuar leyendo acerca de la elección de un navegador web.

Navegación web

Una de las aplicaciones más fundamentales en un ordenador actualmente es el navegador web. Mucha gente no ve completo un sistema operativo si no dispone de un buen navegador. Esta sección comenta los navegadores web evaluados y escogidos para el caso de estudio.

La elección: Konqueror y Mozilla

Este caso de estudio incluye dos navegadores: el navegador Konqueror que viene junto con KDE y el navegador Mozilla. Ambos navegadores son modernos, rápidos y muy capaces. Aquí hay algunos detalles sobre Konqueror.

- **Ubicación:** www.konqueror.org
- **Versión:** 3.0
- **Paquetes:** Incluido con KDE

- **Descripción:** Navegador de diversos medios integrado en KDE y basado en componentes.

Y ahora algunos detalles sobre Mozilla:

- **Ubicación:** www.mozilla.org
- **Versión:** 1.0
- **Paquetes:** ¡Varios! Consulte el sitio web
- **Descripción:** Un navegador web, editor y cliente de correo electrónico de código abierto, muy alta calidad y repleto de características.

Konqueror es el navegador incorporado incluido en KDE. Konqueror es más que un simple navegador web; también es el explorador y gestor del sistema de archivos, y puede reproducir una amplia variedad de tipos de archivos como HTML, PDF, texto y demás. Konqueror realmente es sólo un marco de trabajo que utiliza componentes KDE (uno para cada tipo de archivo) para mostrar los archivos dentro del navegador.

Por este motivo, Konqueror está muy íntimamente integrado con KDE y es fácil utilizarlo desde ese entorno de escritorio. Este fue el motivo principal para escoger Konqueror como uno de los navegadores (aunque también es bastante rápido y robusto), por eso, si va a utilizar otro entorno como GNOME, probablemente escogerá otro navegador.

El navegador Mozilla es un navegador web de código abierto con muchas características. Es una suite tradicional de navegador web, lo que significa que incluye un lector de correo electrónico, un editor de páginas HTML, y otras aplicaciones además del propio navegador. (Compare esto con Konqueror, que es realmente un simple navegador genérico y permite que otros programas KDE realicen el resto de las tareas.) Mozilla es también el componente principal en el que se basan los últimos navegadores de Netscape de la serie 6.x; los productos Netscape incluyen algunas prestaciones no presentes en Mozilla, pero por lo demás los dos son funcionalmente muy parecidos. Mozilla no está realmente integrado ni con KDE ni con GNOME, aunque utiliza el conjunto de componentes GTK+ que también utiliza GNOME. Esto significa que Mozilla funcionará bien en cualquiera de los dos entornos.

El motivo de se utilicen ambos navegadores en este caso de estudio es porque ocasionalmente algunas páginas web funcionarán mejor con uno que con otro. Un motivo para esto es que a veces los desarrolladores de sitios web, sin detenerse a pensarlo, escogen utilizar contenido no estándar para sus páginas; por ejemplo, pueden asumir que los visitantes del sitio van a utilizar un navegador específico. Tanto Konqueror como Mozilla disponen de un buen soporte para los diversos estándares web, pero su "emulación" de algunas de estas singularidades y características no estándar frecuentemente utilizadas por los desarrolladores web varía. En consecuencia, Mozilla funciona con algunas páginas para las que Konqueror no realiza un buen trabajo (o incluso se bloquea o finaliza su ejecución), y viceversa.

Las versiones recientes de ambos navegadores están lo suficientemente mejoradas como para que esto se vuelva cada vez menos común, así que podría descubrir que sólo necesita uno u otro. Además, si escoge un navegador alternativo, su situación y experiencia puede ser completamente distinta.

Las alternativas: Galeon y Opera

Hay una variedad bastante grande de navegadores web de entre la que se puede escoger. La tabla 14.3 incluye una lista de los navegadores más populares disponibles para los sistemas Linux. Si está interesado, debería echar un vistazo a estos navegadores y escoger el que se adapte mejor a su propio gusto.

Tabla 14.3. Navegadores web.

Navegador	URL	Notas
Galeon	galeon.sourceforge.net	Un navegador muy simple basado en el mismo núcleo que Mozilla
Opera	www.opera.com	Un navegador comercial, muy funcional

Extensiones del navegador

Quizás tan importante como el propio navegador, sea el soporte que ofrece el navegador a sus extensiones. Una extensión (*plug-in*) es una funcionalidad desarrollada por una tercera compañía que se distribuye típicamente en forma de biblioteca compartida (o biblioteca de enlace dinámico) y que se carga en el navegador mediante alguna interfaz de programación de la aplicación (*Aplication Programming Interface*, API). En la web son de uso habitual diversas extensiones de navegador; los usuarios que no disponen de estas extensiones descubrirán que muchas páginas les resultan inaccesibles. Esta sección delinea algunas extensiones populares disponibles para Red Hat Linux e instaladas en el sistema del caso de estudio.

Uso de extensiones con los navegadores

Una extensión sólo es útil si puede instalarse y usarse en un navegador. Esto significa que el navegador tiene que implementar una API para ofrecer a las extensiones un modo de ser ejecutadas. Cada navegador podría implementar potencialmente su propia API: sin embargo, en este caso los creadores de las extensiones tendrían que soportar una gran variedad de API.

Para evitar esto, muchos navegadores (en especial los navegadores para los sistemas Unix) han adoptado la API de extensiones de Netscape. Esta fue la API que Netscape Corporation desarrolló para su producto Netscape Navigator. Netscape Navigator fue un navegador muy popular durante muchos años y por eso

hay un gran número de extensiones disponibles que utilizan la API de extensiones de Netscape. Cuando otros navegadores implementan la API de extensiones de Netscape, pueden aprovecharse de esas extensiones.

El navegador web Konqueror de KDE implementa la API de extensiones de Netscape, y de hecho utiliza el mismo directorio de instalación que Netscape Navigator. Por eso, cualquier extensión que se instale para Netscape Navigator también podrá ser utilizada por Konqueror. Además, el navegador Mozilla es un descendiente (al menos en espíritu, sino en cuanto a implementación real) de Netscape Communicator, y por eso Mozilla también soporta estas extensiones. Igualmente lo hace muchos otros navegadores alternativos.

Las siguientes secciones comentan dos extensiones específicas que están disponibles para y funcionan con Konqueror y Mozilla, pero en general cualquier extensión que utilice la API de extensiones de Netscape debería funcionar, con diversos grados de éxito. Estas extensiones comentadas aquí son Macromedia Flash y Java.

Nota

A veces errores en el código u otras singularidades pueden provocar que las extensiones fallen o incluso no funcionen en absoluto.

La extensión Macromedia Flash

El formato Flash de Macromedia es un modo de producir juegos, animaciones, o utilidades interactivas y publicarlas en la web de modo que estén disponibles para todos los usuarios. Los usuarios acceden a estas aplicaciones a través de su navegador web, en lugar de descargarlas e instalarlas. Las aplicaciones son ejecutadas por una extensión que funciona desde los navegadores de los usuarios. Las aplicaciones Flash son típicamente bastante livianas y suelen utilizarse para animaciones cortas o juegos simples o para animar un sitio web.

Flash es extremadamente popular, pero no es completamente un estándar abierto, lo que es una preocupación para mucha gente. Aún así, tanto si le gusta como si le disgusta Flash, es una realidad de la web. Red Hat Linux incluye Flash como parte del paquete netscape-common. Si ya tiene instalado ese paquete, dispone de soporte para Flash. Si no tiene ese paquete y no se va a preocupar por instalarlo (quizás porque no utiliza el ya anticuado Netscape Communicator), entonces puede que necesite visitar el sitio de Macromedia en www.macromedia.com y descargar e instalar la extensión de Flash por sí mismo. También necesitaría hacer eso si lo que desea es actualizar la extensión a una versión posterior. La instalación es bastante fácil y en general implica la extracción de los contenidos del archivo .tar.gz procedente de Macromedia en un directorio (normalmente ~/.netscape/plugins) y después indicar a Konqueror (u otro navegador) que refresque su lista de extensiones instaladas. El archivo de distribución de Macromedia contiene la documentación completa al respecto.

La extensión Java

El capítulo 13 comentó la instalación del kit de desarrollo de Java (JDK) de Sun. Parte de la API de Java incluye una API de applets. Una *applet* (aplicacioncita) es simplemente un programa escrito en Java que está destinado a ser ejecutado desde un navegador web. Al igual que con Flash, el soporte de las applets Java necesita una extensión de navegador.

Con las versiones recientes de la plataforma Java, hay disponible una extensión que conecta la instalación de Java al navegador. Esta extensión no está incluida con las instalaciones de KDE o Mozilla (ya que tampoco lo está Java), pero se incluye junto con el JDK. La instalación de la extensión es bastante fácil, y hay disponibles instrucciones en el sitio web de Sun Microsystems: java.sun.com.

La instalación es bastante simple con Konqueror. El JDK ya contiene todo lo que necesita,así que sólo tiene que preparar su navegador. Si crea un enlace simbólico desde su directorio ~/.netscape/plugins a la biblioteca de la extensión (que es /opt/java/jdk/jre/plugin/i386/ns4/javaplugin.so) y después refresca la lista de Konqueror de las extensiones instaladas, ¡ya estará todo! Es tan fácil como ejecutar este comando:

```
$ ln -s /opt/java/jdk/jre/plugin/i386/ns4/javaplugin.so ~/.netscape/plugins
```

La única advertencia que se debe recordar cuando se instale esta extensión es utilizar la ruta correcta hacia el JDK. En el capítulo 13 se instaló el JDK de tal manera que pudiera actualizarse fácilmente. Específicamente, el JDK se instaló en un subdirectorio de /opt/java (por ejemplo, /opt/java/sun-jdk-1.3.1 para la versión 1.3.1 del JDK) para permitir que se instalasen múltiples JDK. Se creó un enlace simbólico en /opt/java/jdk que apuntase a la instalación JDK que se fijó como la instalación por defecto para el sistema.

Sin embargo, sólo se puede habilitar una extensión Java en cada momento, así que asegúrese de que va a instalar la extensión correcta. Además, asegúrese de recordar actualizar sus instalaciones de la extensión si actualiza el JDK.

Tan importante como un software vital resulta un hardware funcional (e implícitamente el software de bajo nivel que controla el hardware). La siguiente sección comenta cómo configurar y utilizar algunos dispositivos hardware comunes con Red Hat Linux.

Soporte para hardware

Los ordenadores tiene la facultad de hacer un amplio rango de cosas por el usuario, pero frecuentemente sólo será así si disponen del hardware apropiado instalado. Esta sección comenta algunos dispositivos hardware instalados de manera particular en el sistema del caso de estudio, así como el soporte disponible para ellos. Este caso de estudio se basa en Red Hat Linux 7.3, y parte de la información de esta sección es específica a esa distribución. Sin embargo, el soporte hardware

para cualquier sistema Linux realmente se reduce a lo que soporta el propio núcleo. Ya que todas las distribuciones tienen el mismo núcleo, gran parte de este material no es específico de la distribución, y sería aplicable a las demás distribuciones tratadas en este libro, e incluso a otras no comentadas. Se indicará qué cosas son específicas del núcleo y qué cosas son específicas de Red Hat Linux.

Uso de una disquetera ZIP

La disquetera ZIP de Iomega es un producto muy popular. Los discos ZIP son cartuchos de 3,5 pulgadas, aproximadamente las mismas dimensiones de los disquetes tradicionales, que pueden almacenar varios cientos de megabytes de datos. Las disqueteras ZIP están disponibles en un gran número de formatos de interfaz; algunos se acoplan externamente mediante el puerto paralelo del sistema o una interfaz SCSI o USB, mientras que otras son internas y utilizan el estándar ATAPI IDE, que es muy común.

El núcleo Linux soporta disqueteras ZIP en cada uno de estos formatos. Sin embargo, como cada interfaz de disquetera es bastante distinto, el soporte para ellos varía. Por ejemplo, si la disquetera ZIP utiliza una interfaz SCSI, entonces parece literalmente idéntica a cualquier otro dispositivo SCSI. Sin embargo, si la disquetera ZIP es una disquetera interna ATAPI, entonces se soporta como un disco IDE estándar, al igual que si fuera un disco duro IDE. Las disqueteras ZIP USB las soporta un controlador de dispositivo USB especial y se muestran como los dispositivos SCSI. Por último, sólo para complicar las cosas, las disqueteras ZIP en el puerto paralelo las soporta un módulo del núcleo controlador de dispositivo específico para disqueteras ZIP (llamado "ppa" o "imm" dependiendo de la versión particular de la disquetera ZIP), pero se muestran al sistema como los dispositivos SCSI.

Configuración de una disquetera ZIP

Hay cuatro tipos de disqueteras ZIP (puerto paralelo, SCSI, USB, e IDE) y por eso hay cuatro casos de configuración de una disquetera ZIP. Cada uno de estos métodos es bastante diferente a bajo nivel, pero al final todos consiguen el mismo resultado: un archivo de dispositivo que apunta al disco que puede utilizarse para montarlo. Las tres próximas secciones comentan cómo se manejan las disqueteras ZIP para los casos de SCSI, puerto paralelo, e IDE; las disqueteras USB se comentarán más adelante en este capítulo.

Configuración de disqueteras ZIP SCSI

Una disquetera ZIP SCSI está soportada de manera transparente como un disco SCSI por el marco de trabajo SCSI de Linux. Esto significa que la disquetera ZIP estará expuesta al sistema como un dispositivo SCSI estándar, con un nombre como /dev/sda. La parte "/dev/sd" denota un disco SCSI; la "a" indica que es el primer disco SCSI en el sistema. Si tiene una disquetera ZIP SCSI, entonces puede o no ser

el primer dispositivo SCSI, así que tendrá que cambiar "a" por otras letras. Por ejemplo, si su disquetera ZIP es el cuarto dispositivo SCSI, será /dev/sdd.

El controlador del núcleo Linux para los dispositivos SCSI es sd_mod (por "*SCSI Disk*", disco SCSI) y normalmente está incluido como un módulo del núcleo en los sistemas Red Hat Linux (y en el disco RAM [*ramdisk*] inicial para los sistemas que lo necesiten, como se comentó en el capítulo 4).

La disquetera ZIP aparecerá como un simple disco en el controlador SCSI al que esté conectado.

Configuración de disqueteras ZIP de puerto paralelo

Una disquetera ZIP de puerto paralelo contiene hardware especial que es realmente un controlador SCSI. El núcleo Linux incluye un controlador de dispositivo que puede comunicarse con este hardware a través del puerto paralelo y exponer la disquetera ZIP como un dispositivo SCSI estándar. Este toma la forma de un segundo controlador SCSI; es decir, si el sistema ya tiene una tarjeta controladora SCSI, estos controladores del núcleo aparecerán como un segundo controlador SCSI.

Hay dos versiones de este hardware puerto paralelo/SCSI, conocidas como PPA e IMM. Linux incluye controladores (llamados simplemente "ppa" y "imm", respectivamente) para ambas variantes. Los propios controladores crean un controlador SCSI independiente para el dispositivo, y la disquetera ZIP aparecerá como el único dispositivo conectado a ese controlador. Al igual que con los dispositivos SCSI normales, las disqueteras ZIP de puerto paralelo aparecerán como /dev/sda, /dev/sdd y demás.

Configuración de disqueteras ZIP IDE

Una disquetera ZIP IDE se trata como un dispositivo disquetera IDE normal. Linux incluye un controlador de dispositivo que soporta disqueteras IDE. Las disqueteras ZIP IDE quizás tengan más en común con los lectores de CD-ROM IDE que con las disqueteras IDE, pero, aún así, se tratan del mismo modo que las disqueteras de disquetes normales. Es decir, las soporta el mismo controlador de dispositivo.

El controlador de disquetera IDE se llama ide-floppy. Normalmente, Red Hat Linux tiene este controlador incorporado al núcleo (y no en forma de módulo cargable), por eso no tendría que hacer nada, y el núcleo debería detectar su disquetera ZIP durante el arranque.

El dispositivo aparecería como un dispositivo IDE normal, como /dev/hda para el caso del primer dispositivo IDE, o /dev/hdd para el cuarto. (La convención de nombrado es similar a la de los dispositivos SCSI comentada anteriormente, excepto en que se utiliza "hd" en lugar de "sd".)

Nota: Disqueteras ZIP IDE en otras distribuciones

Red Hat Linux incluye el controlador de dispositivo ide-floppy, que soporta disqueteras ZIP IDE como una característica incorporada, en lugar de cómo un

módulo cargable. Esto hace que sea fácil detectar disqueteras ZIP (y, por supuesto, otras disqueteras IDE) durante el arranque. Sin embargo, no todas las distribuciones hacen esto del mismo modo, y algunas incluyen ide-floppy como un módulo del núcleo.
En estos sistemas, puede caer en un escenario que puede resultar confuso al principio, especialmente si está utilizando el módulo de emulación SCSI ide-scsi, que se mencionará en la sección "Uso de una grabadora de CD", más adelante en este mismo capítulo. Específicamente, si se utiliza ese módulo e ide-floppy se compila como un módulo del núcleo, entonces es posible que el módulo ide-scsi se cargue antes del módulo ide-floppy.
En este caso, la disquetera ZIP podría estar gestionada por el módulo ide-scsi (en lugar de por el módulo ide-floppy) y estaría expuesta al sistema como un dispositivo SCSI en lugar de cómo un dispositivo IDE. Por ejemplo, podría ser /dev/sdb en lugar de /dev/hdd.
Si está utilizando otra distribución, o incluso simplemente Red Hat Linux con núcleo particularizado, podría encontrarse en esta situación, y debería estar al tanto de este problema. Todo lo que cambiará será el nombre de dispositivo de la disquetera ZIP.

Montaje de una disquetera ZIP

Una vez que tiene cargados los módulos del núcleo apropiados, la disquetera ZIP aparecerá como un dispositivo de disco de bloques estándar, así que puede interactuar con él como lo haría con cualquier dispositivo de este tipo. Si aún no está seguro del nombre de dispositivo bajo el que aparece su disquetera ZIP, tendrá que comprobar la configuración de su sistema. Puede comprobarla utilizando el comando `dmesg` o el archivo /var/log/messages para ver cómo ha llamado el núcleo a la disquetera, o mirando en el archivo /etc/fstab para ver cómo prepara kudzu los puntos de montaje para el dispositivo.

Hay una singularidad menor de la que hay que ser consciente cuando se utilicen disqueteras ZIP, y es la partición que se utiliza. Si compra un disco ZIP, probablemente vendrá preformateado con el sistema de archivos FAT (*File Allocation Table*). Normalmente, esperaría que esta fuera la primera partición (por ejemplo, /dev/sda1 si su disquetera ZIP es /dev/sda).

Sin embargo, esto no es lo que sucede, y por defecto los discos ZIP nuevos tienen el sistema de archivos en la cuarta partición (es decir, /dev/sda4 para el mismo ejemplo).

Puede modificar esto si vuelve a hacer la partición del disco ZIP mediante el programa fdisk, pero es algo de lo que conviene estar al tanto cuando utilice discos nuevos, o si desea reformatear un disco sin volver a crear sus particiones (como cuando vuelva a dar formato a un disco ZIP con el sistema de archivos ext2 estándar de Linux, que es definitivamente posible).

Nota: Particiones de disco ZIP y Linux

Los discos ZIP tienen de fábrica particiones que utilizan la cuarta partición como la partición real de datos (como /dev/sda4 o /dev/hdc4). Como un tema relacionado, algunos fabricantes de disqueteras ZIP tienen fijados unos parámetros de hardware que pretenden que este comportamiento resulte invisible para el sistema operativo, escondiendo la tabla de partición de los discos ZIP que se inserten en el sistema. Lamentablemente, este comportamiento descompone Linux. Si tiene una disquetera ZIP de este tipo (y funciona bajo Windows u otro sistema operativo, pero no bajo Linux), entonces probablemente haya un puente interruptor en el dispositivo que pueda cambiar para inhabilitar el comportamiento.

Por supuesto, puede que nunca tenga que hacer realmente todo esto, porque Red Hat Linux se encargará de ello en su lugar. Cuando kudzu detecta una disquetera ZIP, no sólo configura los controladores de dispositivo adecuados, sino que también crea un punto de montaje para la disquetera. Específicamente crea el directorio /mnt/zip100.0 para las disqueteras ZIP de 100 MB, y crea la entrada correspondiente en /etc/fstab. (El archivo fstab es bastante simple, y es común a casi todos los sistemas similares a Unix; debería consultar la página del manual sobre fstab si desea más detalles.)

Por eso, para montar su disquetera ZIP, probablemente no tiene que hacer nada más que simplemente ejecutar mount /mnt/zip100.0 y funcionará. Si utiliza un entorno de escritorio como KDE, también se crearán iconos para la disquetera automáticamente, en cuyo caso también podrá utilizarlos. Sin embargo, si necesitara hacer algo directamente con la disquetera o el disco, es útil saber cómo encontrarlo, lo cuál era el propósito de esta sección.

Uso de dispositivos USB

El Bus Serie Universal (*Universal Serial Bus*, USB) es un estándar para la conexión de una gran variedad de dispositivos hardware al ordenador. Cualquier número de ingeniosos artilugios utiliza USB, como reproductores MP3 que lo utilizan para gestionar la lista de reproducción, PDA que utilizan USB para sincronizar datos con el ordenador, cámaras digitales que utilizan USB para transmitir imágenes al ordenador, etc. El soporte USB en los sistemas Linux está proporcionado principalmente por el núcleo Linux, lo que significa que la distribución no tiene mucho que ver con ello. Esta sección comenta varios ejemplos de uso de USB con el sistema del caso de estudio.

En general, no tiene que hacer mucho para que el soporte USB funcione (al menos en esencia). Lo más probable es que Red Hat Linux detecte su hardware controlador de USB correctamente y lo configure en su lugar. Después de esto, todo lo que tiene que hacer es enchufar un dispositivo USB y ver si lo reconoce el núcleo

(que es como decir, si el dispositivo está soportado y hay disponible un módulo del núcleo para él).

Es importante ser consciente de que incluso una vez que se ha cargado el controlador de su dispositivo, todo lo que ha sucedido es que se ha abierto el acceso a él desde el núcleo a los programas del espacio de usuario. Otro modo de decir esto es que cada controlador de dispositivo USB realmente sólo presenta una pequeña API para interactuar con el dispositivo. Los propios programas del espacio de usuario son los responsables de saber cómo utilizar esta mini-API. La API puede tomar distintas formas, dependiendo del controlador. A veces estos son bastante estándar (como para los dispositivos de almacenamiento USB, descritos más adelante en esta sección), y a veces son bastante complejos (como en el caso de un reproductor MP3 USB, igualmente descrito más tarde).

Algunos de los dispositivos USB que usaré como ejemplo en el sistema del caso de estudio se muestran en las siguientes secciones, junto con información sobre cómo configurarlos. El objetivo de estos ejemplos es mostrar cómo el controlador USB específico permite el acceso al dispositivo USB de un modo específico para ese dispositivo, en lugar de ofrecer un tutorial sobre cómo utilizar este tipo de dispositivos. (Ya hemos llegado a un punto en que debería ser capaz de descubrir los mecanismos por sí mismo, una vez que se le den algunas indicaciones.)

Antes de adentrarse en los detalles, ayuda tener una determinada comprensión sobre cómo funciona de manera general la arquitectura USB en Linux. Se incluye una amplia documentación sobre la arquitectura USB de Linux con el código fuente del núcleo, así que esta será simplemente una breve introducción. En general el soporte USB en Linux existe en cuatro niveles, que se describen a continuación.

Marco de trabajo principal

El primer nivel es el marco de trabajo principal. Este es el módulo del núcleo que implementa las llamadas al sistema USB generales dentro del núcleo, al igual que una capa de abstracción con la que interactúan las aplicaciones. Es decir, las aplicaciones no trabajan directamente con el hardware de la controladora USB, sino que en su lugar funcionan con el marco de trabajo principal ofrecido por el núcleo. Este nivel puede verse como análogo a las API de red general de alto nivel, como TCP/IP, que funciona del mismo modo sin importar qué hardware se está usando.

Controlador de la controladora USB

El segundo nivel es el controlador de la controladora USB. Este toma forma de un controlador de dispositivo escrito para una pieza de hardware específica. Refiriéndose a la analogía del sistema de red, la controladora USB sería un equivalente al controlador de la tarjeta Ethernet. Sin embargo, los fabricantes de USB acordaron dos estándares para el hardware de la controladora USB: la interfaz de controladora universal (*Universal Host Controller Interface*, UHCI) y la interfaz de controladora abierta (*Open Host Controller Interface*, OHCI).

Los fabricantes de controladoras USB implementan una u otra de estas especificaciones hardware, lo que significa que no hay una variedad tan amplia de

controladores USB como la hay de controladoras Ethernet, porque los dispositivos de muchos fabricantes utilizan los mismos controladores. De hecho, sólo existen tres controladores comunes para las controladoras USB: el controlador usb-ohci para controladoras que emplean el estándar OHCI y los controladores usb-uhci y uhci para las tarjetas UHCI. (Los dos controladores para UHCI ofrecen un soporte ligeramente distinto para diferente hardware; si un controlador no funciona bien con su configuración, puede desear probar el otro.)

Controladores específicos de dispositivo

El tercer nivel de soporte USB es el de los controladores específicos de dispositivo. Estos son controladores que realmente funcionan con un dispositivo USB específico, como una cámara digital o un reproductor MP3. Puede pensar en los dos primeros niveles como si simplemente ofrecieran conectividad USB, en el sentido de que solamente permite al núcleo y al sistema "ver" dispositivos USB. Sin embargo, cada dispositivo es distinto, por eso incluso aunque ya "vea" el dispositivo, aún tiene que "hablar su idioma" para poder utilizarlo. Un determinado número de controladores de este nivel implementan el soporte para diversos dispositivos USB específicos a nivel del núcleo.

Conexión en caliente del núcleo

El último nivel de soporte USB es el mecanismo "conexión en caliente" (*hotplug*) del núcleo, que se utiliza para dispositivos que se enchufan a un sistema en funcionamiento. Cuando el núcleo detecta un evento de este tipo, invoca a un programa (específicamente, /sbin/hotplug) que es el responsable de la identificación del dispositivo y la carga del controlador apropiado para él. Se identifican los dispositivos comparando su cadena de identificación (que puede obtenerse del propio dispositivo) con una lista de identificadores soportados que se corresponderán con módulos del núcleo; entonces se cargará el módulo correspondiente. Los archivos relacionados (que contienen estas correspondencias y los guiones relacionados con el proceso) se encuentran en el directorio /etc/hotplug. El mecanismo de conexión en caliente funciona en más casos además de en el de USB (también funciona con buses hardware PCI y otros similares).

Uso de una cámara digital

La fotografía digital se está volviendo cada vez más popular a medida que las cámaras digitales se van abaratando. Esta sección comentará cómo conectar este tipo de hardware a un sistema Red Hat Linux.

La sección "Trabajo con cámaras digitales" que está más adelante en este capítulo comenta el software de aplicación que necesitará para completar el círculo. Muchas de estas cámaras digitales utilizan el puerto USB (en lugar del mucho más lento puerto serie RS-232 o el puerto paralelo) para transferir imágenes desde la cámara al ordenador para almacenarlas. Algunas de estas cámaras tienen disponibles controladores USB para Linux.

El modelo de cámara utilizado en este caso de estudio es una Kodak DC-280, aunque el modelo no es en sí mismo relevante; la misma información general se aplica a cualquier cámara digital USB soportada. Una vez que la cámara está conectada al puerto USB del sistema, el controlador usb-core la detectará, y el núcleo activará el mecanismo de conexión en caliente que cargará el controlador específico para la cámara (en este caso, dc2xx).

Una vez que esté cargado el controlador, estará disponible un nuevo conjunto de llamadas de sistema para las aplicaciones del espacio de usuario. (Los detalles de esta API sólo interesan a los desarrolladores, por eso no entraré en ellos aquí.) Aún así, el trabajo no está bastante completo. Aún necesita un programa que haga uso de estas llamadas al sistema.

Probablemente el programa más popular de este tipo sea el programa gphoto que se comentará más adelante. Una vez que gphoto esté funcionando, será capaz de utilizar su cámara digital del mismo modo en que lo haría con cualquier otro sistema operativo.

Uso de un lector de tarjetas flash

Muchos dispositivos, como las cámaras digitales, PDA, reproductores MP3, y muchos más, utilizan tarjetas flash comunes y poco caras para sus labores de almacenamiento. En general estas tarjetas utilizan el estándar *Compact Flash* (CF), o el estándar *SmartMedia* (SM), o algún otro formato. En el caso de las tarjetas CF o SM, también puede adquirir lectores para las tarjetas, disponibles en muchas partes y relativamente baratos.

Típicamente estos lectores de tarjetas flash son simplemente un pequeño puerto de plástico en el que se insertan las tarjetas y que se conecta al ordenador mediante USB.

Muchos de estos lectores se adhieren a un estándar de especificación hardware para dispositivos de almacenamiento en masa USB. De hecho, este estándar soporta muchas más cosas que sólo lectores de tarjetas flash; también incluye bonitos aparatejos como algunos dispositivos de almacenamiento del tamaño de un llavero, disqueteras ZIP USB, grabadoras de CD USB, y otros. El núcleo Linux incluye un controlador para dispositivos de almacenamiento en masa USB llamado usb-storage. Si dispone de un lector (u otro dispositivo) que obedezca al estándar de almacenamiento en masa USB, el mecanismo de conexión en caliente puede detectarlo y cargar el controlador usb-storage automáticamente.

Al contrario de los controladores de la cámara digital descritos antes, el controlador usb-storage no precisa ningún software especial, porque no utiliza ninguna llamada al sistema particularizada. En su lugar, el controlador usb-storage simplemente presenta cualquier dispositivo que detecte como discos SCSI estándar. Por ejemplo, si ya tiene conectados dos discos SCSI a su sistema y entonces conecta un lector de tarjetas flash, aparecerá como el tercer disco SCSI: /dev/sdc. Entonces puede montar, formatear, o incluso crear las particiones de estos dispositivos, exactamente del mismo modo en que lo haría con otro dispositivo de almacenamiento SCSI.

Uso de un reproductor MP3

Otro dispositivo popular es el reproductor MP3 portátil, que permite a un usuario almacenar y reproducir una gran cantidad de música en el formato estándar de alta calidad MPEG-2 Layer 3 (MP3). Estos dispositivos frecuentemente se conectan al ordenador mediante USB.

Algunos de los dispositivos puede utilizar realmente el mecanismo usb-storage que se acaba de comentar para permitirle copiar archivos directamente en y desde ellos. Sin embargo, muchos utilizan algún otro esquema (es decir, una API propia sobre USB). Para algunos de estos dispositivos para los que hay disponible información técnica adecuada procedente del fabricante (o a los que puede aplicárseles ingeniería inversa de un modo independiente), hay disponibles controladores USB. Por ejemplo, muchos dispositivos RIO producidos por SonicBlue están soportados por el controlador USB rio500. Al igual que con las cámaras digitales, conectar un Rio-500 (por ejemplo) al sistema provocará que el núcleo cargue el controlador correcto mediante la llamada al sistema de conexión en caliente. Llegado este punto, estará disponible todo un conjunto de llamadas al sistema, ahora para gestionar los contenidos del Rio-500.

Una vez más, como con las cámaras digitales, necesitará un programa especialmente escrito para utilizar estas llamadas al sistema. Uno de estos programas es el proyecto Rio 500 que se encuentra hospedado en rio500.sourceforge.net. Este es en realidad el sitio web del módulo controlador USB del núcleo de Linux que proporciona la conectividad con el reproductor Rio, pero también es el URL que le ofrece acceso a programas de usuario que puede utilizar para administrar los contenidos del dispositivo. Las herramientas le permiten almacenar archivos MP3 en el Rio, borrarlos, crear listas de reproducción y demás. También hay disponibles herramientas con interfaz gráfica.

Uso de otros dispositivos USB

Linux dispone de controladores para muchos otros dispositivos USB, y la lista sigue creciendo. Hay soporte para joysticks y controladores de juego USB, algunos adaptadores de red USB, cámaras web USB y demás. Este libro obviamente no puede comentar todos estos dispositivos, pero ahora que ha visto algunos ejemplos del uso de dispositivos USB con Linux, debería ser capaz de descubrir cómo añadir otros.

En general, simplemente recuerde el procedimiento básico: cuando conecta un dispositivo, el controlador de la controladora USB lo detectará, y el núcleo ejecutará el programa hotplug para tratar de encontrar y cargar un controlador para él. Después de eso, necesitará un programa que sepa cómo utilizar el controlador (como con las cámaras digitales y los reproductores MP3), o tendrá que interactuar con el dispositivo (como con los dispositivos SCSI mostrados por el controlador usb-storage).

El soporte de Linux para USB tiene sus defectos: a veces los controladores no detectan los dispositivos (en especial cuando los dispositivos son más nuevos que

el controlador), y en ocasiones el soporte para los dispositivos es irregular o lento, pero de forma global funciona bastante bien. En algunos casos (en especial con dispositivos de entrada USB como teclados, ratones y joysticks) puede tener que realizar algo más de trabajo para conseguir que funcionen correctamente, pero ahora ya no debería resultar difícil. Simplemente recuerde cómo funciona el sistema, y sabrá por donde comenzar. USB para Linux no es tan complicado.

Uso de una grabadora de CD

Utilizar CD grabables se ha convertido en un medio muy asequible de almacenar datos, ya sea una copia de seguridad de su sistema operativo, una copia de datos importantes, un álbum de fotos de su última reunión familiar, o una mezcla de pistas de su CD de música favorito. Red Hat Linux hace bastante llevadero el uso de estos dispositivos.

Comprensión de la emulación SCSI

Red Hat Linux intenta tomar un enfoque unificado. En general, Red Hat Linux intenta tratar los dispositivos como dispositivos SCSI tanto como sea posible. Linux incluye un módulo del núcleo conocido como ide-scsi que emula el protocolo SCSI en ayuda de los dispositivos IDE. Esto permite que los dispositivos IDE aparezcan expuestos como dispositivos IDE, y Red Hat Linux puede utilizar este módulo.

Cuando el módulo ide-scsi se carga (mediante un comando como `modprobe` o `insmod`), el controlador analiza los buses IDE, encuentra todos los discos IDE, y después comienza a tratarlos como dispositivos SCSI, convirtiendo llamadas SCSI en llamadas IDE. Por eso, si una disquetera ZIP IDE está presente en el sistema, una vez que se cargue el módulo ide-scsi estará disponible como un disco SCSI, indistinguible de otros dispositivos SCSI.

Esta facultad de tratar cualquier disco como un disco SCSI resulta extremadamente útil, y es realmente muy ingeniosa, si se para a pensarlo. Se usa de manera constante en Red Hat Linux, como ha visto antes y continuará bien durante el resto de este capítulo. Muchas de las secciones que vienen a continuación hacen referencia a este concepto de emulación SCSI.

Red Hat Linux y SCSI

Cuando arranque Red Hat Linux, cargará el módulo ide-scsi si es necesario (es decir, si hay dispositivos que lo necesiten, como grabadoras de CD o disqueteras ZIP). En general, si conecta una disquetera ZIP al sistema, Red Hat Linux la detectará mediante el programa kudzu, que se ejecuta desde /etc/rc.d/rc.sysinit durante el arranque del sistema, como se comentó en el capítulo 4. Debido al empleo que hace Red Hat del módulo ide-scsi, kudzu sólo tiene que preocuparse normalmente por la detección de dispositivos SCSI, en lugar de tanto por los dispositivos SCSI como los IDE. Una vez que se ha detectado el disco, kudzu preparará el dispositivo para su uso.

Configuración de su grabadora

Si su grabadora de CD es un dispositivo SCSI, ya está todo listo y no necesita hacer nada más. Sin embargo, si es una grabadora IDE, entonces es necesario el módulo ide-scsi de manera que el dispositivo IDE quede expuesto como un dispositivo SCSI ante el software de grabación de CD que se va a utilizar. Por fortuna, Red Hat Linux 7.3 detecta las grabadoras de CD IDE, y configura automáticamente el módulo ide-scsi en su lugar para ese dispositivo.

Puede realizar esto pasando un parámetro al núcleo desde el cargador de arranque (que suele ser GRUB). El argumento que se necesita pasar es `[dispositivo IDE]=ide-scsi`; por ejemplo, si su grabadora de CD suele ser el tercer dispositivo IDE, aparecería como /dev/hdc. En ese caso, necesitaría pasar al núcleo el argumento `hdc=ide-scsi`. Red Hat Linux hace esto en lugar del usuario cuando se instala.

Esto significa que probablemente no tenga que hacer nada para conseguir que su grabadora de CD funcione si ya estaba presente cuando instaló el sistema. Sin embargo, si la añade después de que se haya instalado el sistema, tendrá que añadir la línea a la configuración del cargador de arranque usted mismo. (Realmente, el programa kudzu comentado en el capítulo 4 también puede hacer esto en su lugar, cuando reinicia el sistema. Aún así, una vez más, resulta útil saber lo que está pasando.)

Nota

Véase la sección "Uso el cargador de arranque GRUB" del capítulo 4 para obtener más información y referencias sobre el cargador de arranque GRUB.

Una vez que el sistema está listo para mostrar la grabadora de CD como un dispositivo SCSI, sólo necesita utilizar el programa apropiado para grabar un CD. En este capítulo, dentro de la sección "Exploración de las opciones multimedia y de entretenimiento", se comentan dos programas bastante populares, cdrecord y xcdroast.

Uso de tarjetas gráficas 3D

Los ordenadores modernos suelen disponer de algún tipo de aceleración 3D hardware en la tarjeta de vídeo. Esta aceleración la utilizan juegos y algunas clases de software técnico y de ingeniería (como las herramientas CAD y de imágenes médicas). Ya que se trata de hardware, estas tarjetas tienen que estar soportadas bajo Linux por un controlador de dispositivo.

Como leyó en el capítulo 3, la amplia mayoría de los sistemas Linux utiliza XFree86 en conjunto con algún tipo de gestor de escritorio o ventanas como interfaz gráfica. Como resultado, XFree86 es el encargado de soportar hardware 3D en los

sistemas Linux. Esto significa generalmente que XFree86 tiene que tener un controlador para la tarjeta de vídeo de modo que pueda funcionar dentro del servidor X. Además, muchas aplicaciones que utilizan 3D lo hacen a través de la API de programación OpenGL, y por eso, hablando prácticamente, el soporte para 3D bajo Linux también necesita una implementación de OpenGL.

Puede argumentarse sin miedo a equivocarse que el soporte 3D en los sistemas Linux es muy débil, aunque esta situación está mejorando rápidamente, especialmente con las versiones recientes de XFree86. Las principales dificultades son que muchos fabricantes de tarjetas 3D no disponen de controladores y que no existe una implementación de aceleración hardware mediante OpenGL para ese hardware.

Algunos fabricantes hacen pública la información de programación adecuada sobre su hardware, y por eso el propio XFree86 soporta la aceleración 3D para estas tarjetas. En su lugar, otros fabricantes proporcionan controladores descargables para XFree86 que ofrecen aceleración hardware. En este caso, los controladores frecuentemente no son ni software libre ni software de código abierto, incluso aunque su código fuente pueda estar disponible. Frecuentemente tendrá que instalar el controlador en un directorio y modificar la configuración de XFree86 para que cargue el controlador, que se distribuye como una biblioteca de enlace dinámico.

Nota

Vea las secciones "Los orígenes de Linux" y "GNU y Linux: el sistema operativo de los campeones" del capítulo 1, y la sección "Código abierto frente a código libre" del capítulo 2 si quiere más detalles sobre las diferencias entre software libre y software de código abierto.

Sin embargo, incluso si está habilitada la aceleración hardware, el software aún necesita ser capaz de acceder a esas prestaciones de aceleración. Se usa ampliamente la especificación OpenGL, pero no es técnicamente un estándar. Un paquete de código abierto conocido como Mesa tiene la intención de ser compatible con OpenGL (lo que significa que implementa la misma API), pero incluso esta biblioteca sólo es capaz de ofrecer aceleración hardware para chips cuya API pueda entender.

En consecuencia, muchas veces los fabricantes que hacen públicos sus propios controladores para XFree86 también incluyen una implementación de OpenGL, de manera que ese software pueda aprovecharse realmente de la aceleración. La cara positiva de este enfoque es que así consigue un buen soporte de la aceleración hardware de parte del fabricante, pero la cara negativa es que tendrá que instalar estos controladores manualmente, ya que no están incluidos junto con la distribución.

La tarjeta de vídeo en este caso de estudio es una NVidia TNT2 M64. NVidia es un fabricante del último tipo, es decir, ofrece sus propios controladores para XFree86 para proporcionar aceleración hardware bajo Linux. Lamentablemente, los controladores de NVidia obviamente sólo funcionarán con sus tarjetas.

En resumidas cuentas, si la aceleración 3D de su tarjeta de vídeo no está soportada por parte de XFree86, el fabricante de la tarjeta tendrá que proporcionarle ese soporte. Cada fabricante puede hacer las cosas de distintos modos, así que no hay una regla general válida. Lo único que puede hacer es visitar el sitio web del fabricante de su tarjeta para ver qué soporte ofrece, y después seguir las instrucciones que dé el fabricante para la instalación de los controladores. Afortunadamente, esta no es una tarea demasiado complicada.

Hasta ahora ha leído acerca del software de productividad y el software básico de escritorio, al igual que sobre el hardware utilizado en el sistema del caso de estudio. A continuación leerá acerca del software adicional, menos esencial (pero no menos útil) utilizado en el sistema del caso de estudio, comenzando por el software multimedia y de entretenimiento

Exploración de las opciones multimedia y de entretenimiento

Mucha gente utiliza sus ordenadores para crear y gestionar diversos tipos de contenido (como imágenes procedentes de una cámara digital, dibujos, películas caseras, etc.), al igual que para su entretenimiento (como los juegos de ordenador y el vídeo digital). Esta sección describirá como se utiliza el sistema del caso de estudio para estos tipos de actividades y detallará el software necesario para ello.

Trabajo con cámaras digitales

Si posee una cámara digital, entonces es bastante probable que esté interesado en este concepto tan extraño de transferir las fotografías almacenadas en ella al sistema de almacenamiento de su ordenador. Si esto le suena, esta sección le resultará interesante, ya que describe como extrae las imágenes de una cámara digital mediante el programa gphoto.

Las cámaras digitales incluyen invariablemente software para administrar las fotografías contenidas en la cámara. Típicamente este software le permite ver versiones reducidas de las fotografías, copiar algunas o todas al disco de su ordenador, borrar fotografías, y cosas así. Sin embargo, pocos (si es que hay alguno) fabricantes de cámaras disponen de versiones para Linux de su software. ¡Genial, porque puede utilizar el programa de código abierto gphoto en su lugar!

La versión actual de gphoto es la 2.0; sin embargo, la versión 2.0 es muy distinta de la versiones anteriores (que eran una especie de versión alfa, en las series 0.4x y 0.5x). La principal diferencia entre las dos versiones es que las versiones más antiguas ofrecían un programa gráfico muy completo y un entorno muy parecido al software producido por los fabricantes para otros sistemas operativos. La nueva versión 2.0 no tiene una interfaz gráfica y en su lugar se centra en la facultad de gestionar cámaras desde de la línea de comandos, al igual que en ofrecer un conjun-

to de bibliotecas que otros programas pueden utilizar para construir interfaces particulares.

Mientras tanto, KDE 3.0 (que está incluido junto con Red Hat Linux 7.3) incluye un programa llamado Kamera, que hace uso de las bibliotecas de gphoto para ofrecer un soporte integrado para las cámaras digitales desde el navegador Konqueror (Konqueror es el navegador estándar de KDE para la web, los sistemas de archivos, y demás). Este es otro motivo (aunque menor) para la selección de KDE para este caso de estudio. Las instrucciones para utilizar esta funcionalidad están incluidas en la documentación sobre KDE, que está disponible fácilmente desde el sistema, pero en general es tan simple como utilizar las herramientas de KDE para configurar las bibliotecas de gphoto, y después introducir el URL `gphoto:/` en Konqueror.

Manipulación de fotografías con GIMP

Una vez que ha descargado las fotografías digitales desde la cámara al ordenador, querrá manipularlas (es decir, hacer cosas como recortarlas, redimensionarlas, modificar el color y el brillo, y cosas así). Hay una amplia variedad de software disponible que es capaz de hacer esto, y la mayoría de las cámaras digitales incluyen al menos un pequeño programa para editar fotografías. Una vez más, este software rara vez funciona con los sistemas Linux. Por fortuna se creó GIMP (*GNU Image Manipulation Program*, programa de manipulación de imágenes de GNU) para hacer frente a este problema.

GIMP no está tan limitado como su nombre podría sugerir. De hecho, la mayoría de los usuarios parecen estar de acuerdo en que GIMP es al menos tan funcional como sus competidores comerciales, y muchos piensan que GIMP es superior en algunos aspectos. Es bastante sencillo iniciarse en el uso de GIMP, pero es una pieza de software muy complicada (como normalmente lo son los programas verdaderamente potentes).

Con Red Hat Linux 7.3 se incluye un paquete RPM de GIMP, y probablemente esté instalado por defecto en su sistema, a no ser que lo haya eliminado de su selección de paquetes explícitamente. Ejecutar GIMP no debería ser un problema, ¡pero dominarlo es otro asunto! Puede iniciarse en los misterios de GIMP mediante una visita al manual de usuario en el sitio de GIMP, www.gimp.org.

Creación de discos compactos

La sección "Uso de una grabadora de CD" comentó cómo configurar el hardware de su grabadora de CD para utilizarla con Linux (en pocas palabras, el núcleo la trata cómo un dispositivo SCSI mediante la emulación del módulo del núcleo ide-scsi). Sin embargo, eso sólo consigue que el hardware funcione; ahora necesita acceder a ese hardware.

Como con las cámaras digitales, las grabadoras de CD normalmente vienen con software que hace uso del controlador, pero típicamente vuelve a no existir una

versión Linux de este software. Por fortuna, existe un cierto número de programas de código abierto que le permiten utilizar su grabadora de CD. Los programas utilizados en el sistema de este caso de estudio son mkisofs, cdrecord y xcdroast.

Los dos tipos más comunes de CD son los CD de música y los discos de datos que utilizan el formato estándar de sistema de archivos CD ISO-9660 (quizás con alguna extensión).

El programa mkisofs crea sistemas de archivos ISO-9660, y el programa cdrecord graba los CD en CD reales. El programa xcdroast es un envoltorio para estas otras herramientas, y proporciona una interfaz gráfica parecida a las de los programas que encontraría en los sistemas Windows.

Uso del programa mkisofs

Si va a crear un CD de datos, el primer paso es generar una imagen del sistema de archivos ISO-9660 para él. Este es un archivo normal que se encuentra en su disco duro que contiene el propio sistema de archivos que se grabará en un CD. El programa mkisofs, que se incluye con Red Hat Linux 7.3, se utiliza para generar esta imagen, y es un programa bastante sencillo. Debería estar instalado en la mayoría de las configuraciones de escritorio, pero si no lo está, puede instalarlo mediante los métodos habituales.

Utilizar mkisofs normalmente es bastante simple, aunque tiene una gran cantidad de opciones. En su forma más simple, sólo tiene que ejecutarlo con el nombre de un directorio que mkisofs se encargará de colocar en la imagen ISO-9660. No se sienta apabullado por la gran cantidad de opciones en línea de comandos, ya que no usará la mayoría de ellas. Las opciones más habituales son `-J` y `-R` que, respectivamente, indican a mkisofs que cree la imagen ISO-9660 con las extensiones al estándar "Joliet" y "Rock Ridge"; y el argumento `-o <nombre_de_archivo>`, que indica a mkisofs que genere la imagen ISO-9660 en el archivo indicado. La gente suele dar a los archivos de imágenes ISO-9660 la extensión .iso, pero este no es en absoluto un estándar formal.

Cuando lo ejecute, mkisofs copiará todos los archivos que se encuentren en el directorio especificado (al igual que en sus subdirectorios) en el archivo imagen que especificó. Una vez que haya acabado, puede grabar la imagen a un CD, o comprobarla pasando el argumento `-o loop` a `mount`, que provocará que `mount` monte un archivo como si fuera un volumen mediante el módulo del núcleo loopback. Por ejemplo, los siguientes comandos crearán una imagen ISO-9660 en /tmp/foo.iso a partir de los contenidos del directorio /home/foo y lo montarán (para comprobarlo) en el directorio /mnt/foo:

```
$ mkisofs -R -J -o /tmp/foo.iso /home/foo
$ mount -o loop -t iso9660 -r /tmp/foo.iso /mnt/foo
```

El argumento `-o loop` que se pasa a `mount` necesita que el controlador del dispositivo de bloques del bucle de retorno (*loopback*), que es un módulo del núcleo llamado "loop", esté disponible para el núcleo; este es el caso con el núcleo por defecto de Red Hat Linux 7.3.

Uso del programa cdrecord

El programa cdrecord se utiliza para grabar datos en los CD reales, y es capaz de crear CD de música (que podría reproducir en la mayoría de los reproductores de CD estándar recientes) y CD de datos (como a partir de un archivo imagen ISO-9660 creado mediante mkisofs). Como mkisofs, cdrecord debería estar incluido en la mayoría de las instalaciones de escritorio de Red Hat Linux.

Utilizar cdrecord implica proporcionarle un archivo o lista de archivos que debe grabar en un CD, especificar el dispositivo grabadora de CD que se utilizará e indicar el formato del CD (como de datos [*data*] o audio [*audio*]). Una vez más, cdrecord tiene muchas opciones, pero en general no necesitará la mayoría de ellas. Por ejemplo, para grabar la imagen creada en el ejemplo anterior en un CD, podría utilizar un comando como este:

```
$ cdrecord dev=1,1,0 -data /tmp/foo.iso
```

El único asunto complicado con este comando es el argumento `dev=`. Este argumento indica a cdrecord qué dispositivo del bus SCSI debe utilizar para crear el CD. Obviamente, este no es el formato al que está acostumbrado (como, por ejemplo, /dev/sda), pero por fortuna cdrecord puede comprobar todos los dispositivos SCSI en su lugar. Simplemente ejecute el comando `cdrecord -scanbus`, y encontrará todos los dispositivos SCSI disponibles y mostrará sus identificadores de un modo legible por los usuarios. Debería ser capaz de identificar fácilmente su grabadora de CD de entre la lista y utilizar su descriptor en el argumento `dev=` que se le pase a cdrecord.

Uso del programa xcdroast

Como probablemente haya notado, cdrecord y mkisofs pueden ser programas bastante complicados. A estas alturas debería ser capaz de descubrir cómo funcionan sin demasiados problemas, por supuesto, pero, aún así, puede preferir una interfaz de usuario gráfica. Si eso es lo que quiere, entonces deseará probar xcdroast.

El programa xcdroast es un envoltorio gráfico para mkisofs y cdrecord, y le permite crear y grabar CD de audio y de datos recogiendo las opciones que necesita pasar a esos programas y ejecutándolos en su lugar. Una vez más, xcdroast viene incluido con Red Hat Linux 7.3. Si prefiere un entorno gráfico, entonces definitivamente debería probar este programa; realmente puede hacer que el proceso de la creación de CD sea bastante llevadero.

Reproducción de archivos de audio digital

Muchos usuarios disfrutan escuchando música mientras trabajan, y uno de los formatos más populares es el estándar MP3. Un estándar parecido es el formato Ogg Vorbis; la compañía Xiphophorous tiene un proyecto llamado "Ogg" que produce (entre otras cosas) un formato de audio digital comprimido llamado "Vorbis". El estándar Ogg Vorbis es un proyecto completamente abierto, mientras que deter-

minados aspectos del estándar MP3 están copados por las patentes de varias empresas. (Para obtener más información, visite el sitio web de Xiphophorous en www.xiph.org.) Esta sección comentará programas que pueden reproducir archivos en formato Ogg Vorbis y MP3.

El programa mpg321

Red Hat Linux incluye el programa mpg321 para reproducir archivos MP3. (El paquete RPM es mpg321.) Puede reproducir cualquier archivo MP3 simplemente ejecutando el comando `mpg321 <nombre_de_archivo>`. También puede especificar una lista de archivos y, además, hay disponible una serie de opciones en línea de comandos para crear un orden aleatorio para la reproducción canciones y cosas así.

Comandos como `mpg321 -z *.mp3` funcionan muy bien para reproducir archivos MP3, pero el paquete mpg321 no incluye ningún programa para codificar archivos MP3. Si desea codificar archivos MP3, tendrá que utilizar otro programa. Lamentablemente, el proceso de codificación de archivos MP3 es uno de los aspectos de MP3 sobre los que algunas compañías ostentan patentes, así que puede ser difícil encontrar codificadores MP3 libres y gratuitos. El proyecto LAME es un codificador MP3, pero no se incluye un paquete para él con Red Hat Linux 7.3. No obstante, aún puede instalarlo (empleando las técnicas mencionadas en los capítulos 8 a 13), y puede encontrar toda la información relacionada con este proyecto en www.mp3dev.com.org/mp3/.

El programa ogg123

Recuerde que el proyecto Ogg produce una especificación de codificador/decodificador (codec) audio conocida como Vorbis. El codec Ogg Vorbis es bastante similar al de MP3, pero está basado en distintos algoritmos y tecnologías, y por eso no está bloqueado por las patentes que obstruyen el uso de MP3.

Hay tres paquetes RPM relevantes para Vorbis: libvorbis, libvorbis-devel, y simplemente vorbis. Los dos primeros contienen las bibliotecas compartidas que permiten que otros programas (como XMMS, que se comentará más adelante) codifiquen y decodifiquen archivos Vorbis. El paquete vorbis contiene herramientas en línea de comandos de reproducción y codificación.

El comando necesario para reproducir archivos Ogg Vorbis es `ogg123`. Este programa es muy parecido a mpg321 en la sintaxis de su línea de comandos, como se pretendía, ya que muchos usuarios ya estaban familiarizados con mpg321 (y el programa en el que se inspiró el propio mpg321, mpg123, que no se comenta en este libro). El comando `ogg123 -z *.ogg` reproduce todos los archivos Vorbis que se encuentren en el directorio actual según un orden aleatorio, y es similar al comando mpg321.

Al contrario que mpg321, el paquete Ogg Vorbis también incluye un programa para codificar archivos Vorbis a partir del formato estándar .WAV para audio digital sin compresión. El programa se llama oggenc, y, como un programa en línea de comandos típico, su funcionamiento ya debería resultar muy familiar.

Los paquetes de las bibliotecas (libvorbis y libvorbis-devel) son usados por otras aplicaciones para integrar el soporte para el codec Ogg Vorbis. Por ejemplo, el programa XMMS (que se comentará en la siguiente sección) utiliza estas bibliotecas para permitirle la reproducción de archivos Vorbis al igual que la de los archivos MP3. (Estos RPM también son utilizados por el propio paquete vorbis.)

Si le gusta codificar sus CD de música en un formato más portátil, entonces puede que desee investigar Ogg Vorbis como una alternativa a MP3. Las ventajas de Vorbis son que al menos es de tan alta calidad como MP3 (y mejor según la opinión de algunos), y que es un estándar completamente abierto, al igual que de código abierto. Si esto le parece importante, debería inspeccionar el Proyecto Ogg y sus herramientas.

El sistema multimedia X

El sistema multimedia X (*X Multimedia System*, XMMS para abreviar) es una interfaz gráfica utilizada para reproducir una gran variedad de archivos. En apariencia es parecido al popular reproductor de música WinAMP disponible para los sistemas Windows, y soporta muchos formatos de archivo. De manera notable, soporta ambos formatos comentados con anterioridad, MP3 y Ogg Vorbis. XMMS también ofrece algunas características atractivas para visualizar la música al mismo tiempo que se reproduce.

Si prefiere utilizar un programa gráfico para gestionar listas de reproducción de archivos de música, o si quiere mezclar archivos MP3 y Vorbis en una única lista de reproducción, entonces debería considerar XMMS.

Hay disponibles otros programas gráficos que llevan a cabo la misma tarea en los sistemas Linux, pero XMMS tiene la virtud de estar incluido en Red Hat Linux. Para este caso de estudio, XMMS resulta más que adecuado y es de hecho bastante aconsejable su instalación.

Reproducción de archivos de vídeo digital

Al igual que muchos usuarios disfrutan escuchando archivos de audio digital, muchos otros (o incluso los mismos) se entretienen viendo archivos de vídeo digital. En la actualidad hay mucha actividad en esta área, varias opciones de reproductores de código abierto y versiones comerciales entre las que escoger. En primer lugar, esta sección comentará los reproductores reales utilizados por el sistema de este caso de estudio, y después indicará algunas otras opciones.

El reproductor multimedia avifile

Algunos estándares de codificación de vídeo son estándares abiertos, como MPEG (*Motion Picture Expert Group*), mientras que otros están férreamente controlados por sus propietarios, quienes dictaminan quién puede y quién no puede utilizarlos. Lamentablemente, parece que la mayoría de los codec forman parte de la segunda categoría, y como no son un estándar abierto, como poco es difícil escribir software

para poder reproducir sus esquemas de codificación. Por fortuna, aquí es donde entra en acción el programa avifile.

El proyecto avifile produce un programa reproductor multimedia que lee realmente archivos binarios de bibliotecas de enlace dinámico (DLL) de los sistemas operativos Windows de Microsoft y accede a las implementaciones de los codec contenidas en las bibliotecas. Este es un enfoque interesante, porque significa que avifile puede reproducir archivos de vídeo codificados con un cierto formato sin tener realmente que implementar de manera directa el codec. Sin embargo, este no es un enfoque infalible, y obviamente sólo funciona para codecs cuyas DLL estén disponibles y cuyas licencias permitan que se utilicen de esta manera.

A pesar de sus limitaciones, avifile es bastante funcional, y por eso se utiliza en el sistema del caso de estudio. No está incluido en Red Hat Linux (aunque está incluido en otras distribuciones como Debian GNU/Linux y Mandrake Linux), pero su instalación no es difícil. La única advertencia que hay que tener en cuenta es que cuando construya el programa tiene que especificar el nombre de un directorio en el que se instalarán los archivos DLL de Windows, así que es importante tomar esta decisión de antemano.

Una desventaja menor de avifile es que puede ser algo difícil compilarlo. Además, no reproduce archivos MPEG, lo que significa que será necesario un programa independiente para ese propósito. Sin embargo, avifile es funcional, si puede sobreponerse a las dificultades que supone utilizarlo en un sistema Red Hat.

El programa smpeg

Una desventaja menor de avifile es que no puede reproducir archivos MPEG. Por eso, se necesita un programa separado para estos archivos. Por fortuna, Red Hat Linux incluye un reproductor capaz de reproducir archivos MPEG, en forma del paquete smpeg, que instala un programa llamado plaympeg.

El paquete smpeg es realmente una parte del proyecto SDK, que es una biblioteca multimedia mencionada con algo más de detalle en la siguiente sección. No tendrá ningún problema en absoluto al usar el programa plaympeg una vez que haya instalado el paquete.

Otros programas

Los dos programas que se acaban de mencionar son, obviamente, dos programas distintos. Esto significa que tiene que prestar atención al tipo de archivo de vídeo que en su momento intente reproducir, y escoger el programa correcto para ello. Además, muchos reproductores multimedia actuales pueden conectarse a flujos de datos de vídeo de servidores y reproducirlos en directo. Esto puede ser difícil o a veces imposible tanto con avifile como con smpeg. Por esta razón, los usuarios que vean vídeos habitualmente pueden considerar que estos programas resultan insatisfactorios.

Hay otras soluciones disponibles, pero en general no son gratuitas. Por ejemplo, CodeWeavers, Inc. tiene un producto conocido como CrossOver Plugin, que es una extensión para los navegadores Mozilla y Netscape 6.x. Esta extensión es una espe-

cie de meta-extensión en el sentido de que permite que extensiones para los navegadores de Windows funcionen junto con Mozilla y Netscape en los sistemas Linux. Esto incluye, entre otras cosas, el reproductor multimedia de Windows (Windows Media Player) de Microsoft. Este producto no es gratuito, pero su coste puede merecer la pena para algunos usuarios. Más información sobre el producto puede encontrarse en www.codeweavers.com/products/crossover/.

Hay disponibles muchas otras aplicaciones. KDE, por ejemplo, dispone de un reproductor multimedia integrado, pero en general no resulta tan capaz como avifile. De manera similar, el programa Xine tiene una interfaz de usuario muy atractiva y puede reproducir tanto archivos MPEG como AVI, pero no puede reproducir algunos de los codecs más nuevos. Puede desear emplear un poco de tiempo libre con su motor de búsqueda favorito en busca de otras opciones que puedan existir. Hay una gran cantidad de actividad en esta área, y podría descubrir algo que se adapte a lo que necesita.

Juegos

Uno de los indicadores finales de que un sistema operativo ha entrado en la corriente principal de los sistemas de escritorio es el momento en que hay disponibles juegos para él de una manera amplia. Linux puede utilizarse como un sistema de escritorio (como demuestra este capítulo), pero probablemente sea demasiado pronto para decir que ha alcanzado la corriente principal a ese respecto. Como resultado, hay juegos comerciales para Linux, pero, honestamente, no tantos.

Las estrategias de algunas empresas de juegos incluyen a Linux y han producido juegos lucrativos para esta plataforma. Merece la pena mencionar los juegos de id Software de acción en primera persona (*First Person Shooter*), como la serie de DOOM, Quake y demás. Estos juegos funcionan bien en Linux, e incluso disponen de soporte para la aceleración hardware 3D. Tampoco se encuentra sola id Software; muchas más compañías han adaptado sus juegos a Linux. Lamentablemente, estos juegos pueden resultar a veces muy complicados de instalar, por decir poco. Por ejemplo, el soporte del hardware gráfico 3D se comentó previamente en este capítulo; este es frecuentemente uno más de los muchos obstáculos que hay que superar antes de lograr que la mayoría de los juegos funcionen sobre los sistemas Linux. En pocas palabras, los juegos comerciales sobre Linux no han llegado aún al punto en que la instalación sea llevadera (por supuesto, ya que ha leído los capítulos 8 a 13, debería resultarle fácil, pero ¡no suele ser tan sencillo como instalar simplemente un RPM!).

No obstante, eso es cierto de manera general para los juegos comerciales que siguen la corriente principal del mercado. Si no se trata de un jugador ávido interesado en los juegos más actuales y fastuosos, realmente puede quedar sorprendido ante el número de juegos de escritorio disponibles para Linux. Puede encontrar una gran cantidad de juegos de cartas, de ingenio, juegos simples de acción, y muchos más para KDE y GNOME. Además, hay diversos esfuerzos orientados a producir motores multimedia de código abierto comercialmente viables para Linux. De ma-

nera notable, el Simple DirectMedia Layer (SDL), que se puede encontrar en www.libsdl.org, es un conjunto de bibliotecas para el desarrollo de aplicaciones multimedia intensivas, como reproductores multimedia y juegos, en una amplia variedad de sistemas operativos; Red Hat Linux incluye paquetes RPM para estas bibliotecas. Se están desarrollando muchos juegos libres e incluso algunos juegos comerciales sobre esta API.

La conclusión es que si es un jugador empedernido, podría quedar decepcionado por el estado de los juegos sobre Linux. Aún así, si en realidad lo único que busca son algunos entretenimientos ocasionales, probablemente con lo que hay tenga más que suficiente. Si está interesado en aprender más, un sitio fantástico es el foro de juegos de Linux que se encuentra en www.linuxgames.com. Para este caso de estudio, sucede que mi dosis de juegos se ve suplida con algunos de los viejos juegos clásicos como NetTrek, así que todos estos asuntos no son importantes.

En la siguiente sección, avanzaremos desde los juegos hasta el trabajo. Leerá sobre algunos trucos interesantes y técnicas de gran utilidad de las que puede aprovecharse, ahora que ha preparado su sistema Linux.

Implementación de la conectividad y el acceso remoto

Como punto final, este libro comentará varios modos en los que el sistema del caso de estudio ofrece una conectividad y accesibilidad mejoradas. Puede no necesitar ninguna parte de esta información, así que siéntase libre de hojearla sin más. Sin embargo, incluso aunque la utilice, puede desear leer completamente esta sección, ya que proporciona algunas ideas interesantes que pueden hacerle pensar.

Acceso remoto al escritorio

El sistema del caso de estudio está conectado a Internet en todo momento mediante un módem de cable. Esto proporciona la facultad de acceder al sistema de manera remota mediante SSH. No es una característica utilizada muy habitualmente, pero de vez en cuando resulta muy práctica, así que resulta algo que merece la pena considerar si dispone de una conexión permanente de gran velocidad similar (resultaría muy arduo utilizar un mecanismo similar sobre una conexión telefónica).

La configuración de OpenSSH para el sistema de escritorio no es más que la instalación comentada en el capítulo 8. Todo el material de ese capítulo se aplica a este caso de estudio; sin embargo, la instalación del caso de estudio se basa en los paquetes RPM estándar de OpenSSH proporcionados por Red Hat, de manera que no se ha construido el sistema a partir del código fuente. Excepto este detalle, puede leer el capítulo 8 para obtener más información.

La facultad de ofrecer un acceso seguro y cifrado al sistema puede resultar muy conveniente. Por ejemplo, si está viajando, puede entrar en el sistema para comprobar su correo (ya sea empleando un programa en línea de comandos, o mediante la facultad de SSH de reenviar conexiones al sistema de ventanas X automáticamente si utiliza un cliente gráfico). Quizás resulte más importante poder ejecutar OpenSSH para acceder remotamente a los archivos del sistema mediante las utilidades scp y sftp. Esto puede ser increíblemente útil si olvidó en casa un archivo que necesita en el trabajo, o mientras está de viaje. Simplemente recuerde mantener el sistema actualizado, para evitar exponerse a vulnerabilidades de seguridad innecesarias.

Provisión de otros servicios

Otra característica, quizás más novedosa, proporcionada por el escritorio es un servidor web proxy personal. Muchas empresas utilizan un cortafuego para proteger la red corporativa, lo que redunda en que los usuarios necesitan un servidor proxy (un intermediario). Frecuentemente estos servidores proxy emplean diversas técnicas de filtrado para restringir el acceso del usuario a determinado material de la web. A veces se hace para proteger el buen nombre de la empresa. Por ejemplo, ¡podría generarse una mala impresión sobre la empresa si se observara que sus empleados visitan sitios web cuestionables durante su periodo de trabajo! Por supuesto, a veces el servidor proxy corporativo es simplemente demasiado lento, o incluso permanece inactivo.

Nota

Antes de utilizar **cualquiera** de las técnicas tratadas en esta sección, ¡compruebe con los administradores de su red que está permitida! No todas las empresas pueden permitir estos tipos de conexiones.

Es posible ejecutar un servidor proxy como tiniproxy (que se encuentra disponible en tiniproxy.sourceforge.net) en su sistema doméstico de escritorio conectado directamente a Internet, como con el sistema del caso de estudio. Conectándose a continuación a su sistema doméstico desde su sistema de escritorio empleando OpenSSH y configurando el reenvío por agente SSH puede acceder a su servidor proxy doméstico desde su ordenador en el trabajo. (En general, simplemente indica a su navegador web que utilice "localhost" como el servidor proxy, y habilita el reenvío por agente SSH para reenviar esas peticiones a través de la conexión SSH para que sean respondidas por su servidor proxy personal que se encuentra fuera del cortafuego.)

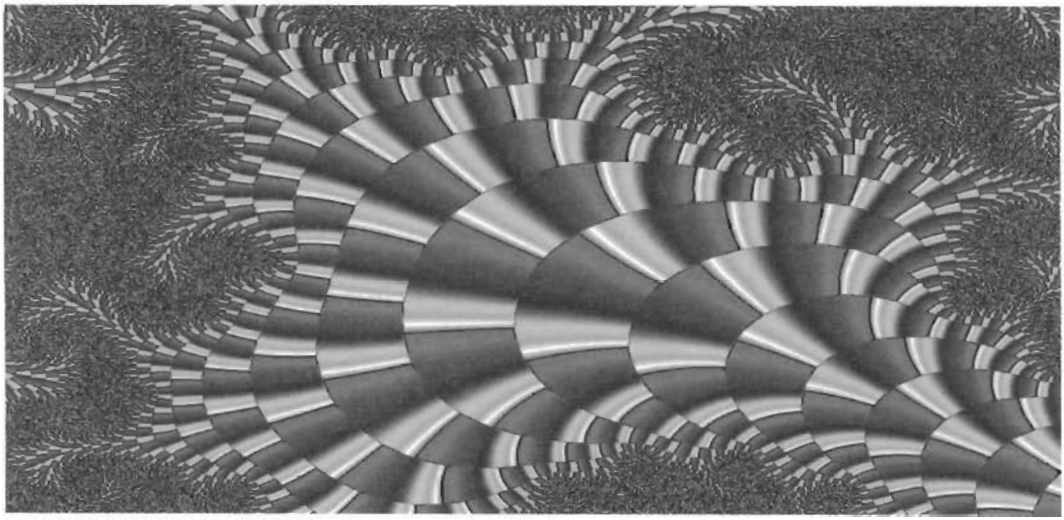
Esto significa que todas sus peticiones web parecerán haberse originado en su propio ordenador de escritorio personal, al otro lado del cortafuego de la empresa. Ya que su servidor proxy lo administra usted, no está restringido en cuanto a los sitios que puede visitar. Esta puede ser una gran característica para usted en su

trabajo si el proxy corporativo bloquea el acceso a sitios a los que necesita acceder o funciona mal o está atascado.

Sin embargo, ¡no debería proceder sin más y preparar esta situación! Muchas empresas no aprobarán este comportamiento, y podría estar violando una política de empresa si hace esto. Esta configuración es muy fácil de montar, pero antes de hacerlo es importante que compruebe con los administradores de red si tiene su aprobación. Su empleador puede preferir o explícitamente pedir que no haga esto. Esta técnica se muestra aquí simplemente para mostrar algunas de las cosas tan creativas que puede hacer con un sistema Linux una vez que lo ha dominado. ¡No nos eche la culpa si utiliza esta técnica (o una similar) y sucede algo desafortunado!

Resumen

Ahora ya ha leído un caso de estudio de un sistema de escritorio que funciona con Red Hat Linux. Las partes primera a tercera le presentaron una distribución y trataron tres de ellas en detalle, además de proporcionarle las técnicas básicas necesarias para instalar software y personalizar su sistema. Este caso de estudio señaló algunas de las cosas fantásticas que puede hacer con su sistema, y mostró las formas en que se puede ajustar y personalizar una instalación estándar de Red Hat Linux para cubrir sus necesidades. Esperemos que gracias a este capítulo haya llegado a la conclusión de que Linux es un sistema operativo muy viable como sistema de escritorio.



15
Un entorno de desarrollo de software corporativo

Este capítulo presenta el segundo caso de estudio de esta cuarta parte, que trata de una estación de trabajo de desarrollo. En la vida real, soy un científico informático e ingeniero de software, y este caso de estudio se basa en el sistema que monté para cubrir estas necesidades. Una vez que haya leído este capítulo, sabrá cómo duplicar esa configuración. Y lo que es igualmente importante, habrá adquirido las bases para ajustarla para poder particularizarla para su propia red y necesidades. Si también ha leído el caso de estudio anterior sobre un sistema de escritorio, entonces tras la lectura de este capítulo se dará cuenta de cómo el simple ajuste de unos pocos detalles puede producir un sistema muy distinto.

Comienzo

El primer paso para la construcción de cualquier sistema es comprender los detalles. Esta sección describe los objetivos y requisitos que guiaron el desarrollo de la configuración de la estación de trabajo de desarrollo comentada en este caso de estudio. Debería leer esta sección y después compararla con sus propias necesidades y objetivos.

¿A quién se dirige este ejemplo?

Este ejemplo está orientado hacia los desarrolladores de software, pero una gran parte de él es aplicable de manera más general. El capítulo 14 detalló una configura-

ción que resultaba útil para el uso simple y cotidiano de escritorio; esa configuración podría funcionar bien en algunos entornos corporativos, pero normalmente este tipo de entornos es mucho más complejo que su sistema de escritorio doméstico medio. Por eso, incluso si todo lo que necesita en el trabajo es una configuración de escritorio, podría necesitar mucha de la información mostrada en este capítulo, en especial el material de las secciones "Herramientas de administración y red", "Interoperación corporativa", y "Cuestiones de seguridad". Por supuesto, si realmente es un desarrollador software, necesitará los contenidos de este capítulo.

¿Para qué se utiliza este sistema?

El sistema de este caso de estudio se utiliza para el desarrollo de una gran variedad de propósitos en diversos lenguajes. Por ejemplo, esta misma configuración con cambios sorprendentemente menores se ha utilizado para desarrollar aplicaciones en Java 2 Enterprise Edition (J2EE), al igual que software de control de tiempo real para equipamientos de distribución eléctrica. No obstante, este capítulo, se centra sólo en el primer caso, ya que es realmente el caso más complejo en cuanto a términos de configuración del sistema se refiere. Muchos de los elementos en este capítulo preparan el entorno para el desarrollo de software.

Sin embargo, la misma cantidad, o más, del contenido de este capítulo se centra en la configuración del sistema para que funcione correctamente con una red corporativa. Este tipo de redes suele tener un mecanismo de autenticación y de distribución de archivos muy importante (como NIS+ y NFS, que se comentarán en este capítulo), y por eso, obviamente, la estación de trabajo tiene que poder vivir en paz con estos servicios. Además, las realidades de los negocios modernos dictan prestaciones adicionales que el sistema necesita soportar, como la facultad de trabajar con documentos de oficina o utilizar el software estándar de la empresa, que puede que sólo exista para otros sistemas operativos, y cosas así.

Estos objetivos duales (y a veces enfrentados) son lo que encontrará en este caso de estudio. Lo que no encontrará son detalles extensos sobre cómo instalar mucho del software de esta sección; este capítulo asume que ya ha leído los capítulos 7 a 13 y hace referencia a ellos de manera extensa. Además, este caso de estudio es en gran parte una versión más amplia del caso de estudio del escritorio, así que podría resultar de utilidad leer también el capítulo 14.

Como el capítulo 14, este capítulo comienza a partir de los componentes básicos. En la siguiente sección, leerá sobre cómo comenzar a partir de un sistema Red Hat Linux 7.3 estándar y utilizarlo para reconstruir este caso de estudio.

Construcción del sistema

Esta sección describe cómo construir los componentes básicos de la estación de trabajo de desarrollo. Como en todos los casos de estudio, este ejemplo comienza a

partir de un sistema Red Hat Linux 7.3 estándar y particulariza la lista de paquetes instalados y la configuración del sistema estándar para cumplir los objetivos establecidos.

Comienzo con el escritorio

Una estación de trabajo de desarrollo es en su mayor parte un caso más amplio de un sistema de escritorio. Como probablemente ya haya leído acerca de un sistema de este tipo en el capítulo 14, no hay necesidad de repetirlo todo aquí. Por eso, por obvio que parezca, el primer paso en la instalación de un sistema escritorio corporativo es realizar la instalación de un escritorio.

Por supuesto, no toda la información del capítulo 14 es relevante. Por ejemplo, probablemente no tenga mucha necesidad de gestionar un reproductor MP3 portátil desde su estación de trabajo de oficina (aunque puede que aún necesite el soporte USB). Por eso, en realidad no necesita todo el capítulo 14; sólo necesita los conceptos básicos.

De manera específica, está interesado en componentes como los siguientes:

- KDE como entorno de desarrollo
- OpenOffice para la productividad
- Konqueror y Mozilla para la navegación web
- Conectividad y acceso remoto
- Soporte para hardware (como una disquetera ZIP o una grabadora de CD, cuando no ése reproductor MP3)

En otras palabras, comience con el sistema de escritorio del capítulo 14 e instale las características apropiadas para su situación. Este mismo caso de estudio emplea la lista anterior pero, por supuesto, su propia lista puede ser distinta.

Elección de los paquetes

El resto de este capítulo es una lista de paquetes de software para instalar y, cuando sea necesario, instrucciones sobre cómo configurarlos. En consecuencia, podría ser una buena idea hojear el resto de estas secciones antes de instalar el sistema básico para hacerse una idea de los paquetes que necesitará seleccionar durante el proceso. Sin embargo, en su mayor parte, puede conseguir la mayoría de lo que necesita mediante la selección de las opciones de paquetes de nivel superior que Red Hat le ofrece durante la instalación.

Seleccionar unas cuantas categorías de alto nivel le proporcionará el 85 por ciento de lo que necesita (parte del software comentado en este capítulo no se incluye junto con Red Hat Linux, como por ejemplo OpenOffice). Para el resto, puede instalarlo individualmente desde el CD o mediante el comando `up2date`

(que se comentó en el capítulo 4) y después configurarlo tal y como necesite, empleando la información de este capítulo como una guía. Otra alternativa consiste en seleccionar simplemente "Instalar todo"; sin embargo, recuerde que esto instalará grandes cantidades de software innecesario, lo que puede tener graves repercusiones sobre la seguridad del sistema.

Detalles de la configuración

No es necesaria una gran cantidad de modificaciones en la configuración por defecto de Red Hat Linux. Por supuesto, una gran parte del resto de este capítulo implica configurar los paquetes individuales que se instalen, pero no tiene que hacer gran cosa con el sistema estándar.

Quizás lo único que deba considerar es sí desea dejar habilitado el cortafuego por defecto. Sin embargo, esto depende de su entorno y la política que emplee; consulte la sección "Cuestiones de seguridad" que se encuentra más adelante en este mismo capítulo.

En la siguiente sección, leerá acerca de un software importante que permite que su nueva estación de trabajo se conecte a una red. Leerá sobre la información que necesita conseguir para poder preparar el sistema, y después aprenderá qué debe configurar y cómo hacerlo.

Herramientas de administración y red

Esta sección comenta varias herramientas y sistemas habituales para administrar diversas partes del entorno de un ordenador. Estas herramientas suelen emplearlas las redes corporativas debido al modo en que facilitan la administración. Algunas de estas herramientas le sonarán del escritorio del caso de estudio del capítulo 14, mientras que otras pueden resultarle nuevas. Como siempre, ya que este es un caso de estudio de un sistema real, su entorno puede no ser idéntico a este otro.

Compresión de la red local

El primer paso en la configuración de su nueva estación de trabajo consiste en entender exactamente qué va a hacer con ella. Después de todo, no puede configurar algo si no sabe que tiene que hacerlo. Esta sección describe las realidades de la red en la que reside este caso de estudio. Es un entorno corporativo bastante típico, así que puede que descubra que su propio entorno es muy parecido; sólo tenga cuidado con las posibles diferencias sutiles. La red empleada en este caso de estudio funciona así:

- Las estaciones de trabajo obtienen sus direcciones IP mediante DHCP (protocolo de configuración dinámica de máquinas).

- La autenticación del sistema recae en los servicios de información de red estándar (*Network Information Services*, NIS+). El nombre de dominio NIS es en este caso "universal".
- Los directorios locales de los usuarios y los archivos relacionados con proyectos comunes se comparten entre las estaciones de trabajo (y los servidores) mediante el estándar NFS.
- El grupo de servicios de gestión de tecnología de la información (*Information Technology Management Services*, ITMS) de la organización rastrea el estado del sistema mediante el software Asset Insight de Tangram Enterprise Solutions.
- El grupo de soporte de red requiere acceso como *root* para todos los sistemas similares a Unix de la red mediante una clave OpenSSH.

No resultará sorprendente que cada uno de estos asuntos sea el tema del resto de esta sección. Cada una de las subsecciones siguientes proporciona información sobre cómo configurar estos aspectos del sistema, pero no sobre cómo instalarlos; supongo que a estas alturas puede hacer esto por sí solo, tras la lectura de los capítulos 7 a 13.

Uso de DHCP en el sistema

El protocolo de configuración dinámica de máquinas (*Dynamic Host Configuration Protocol*, DHCP) se utiliza para escritorios, estaciones de trabajo y, en ocasiones, para servidores, para conectarse a Internet. Ya habrá visto aparecer a DHCP en algunas ocasiones; por ejemplo, el caso de estudio del sistema de escritorio del capítulo 14 también utiliza DHCP. Como sin duda habrá intuido a partir de esto, DHCP es un sistema muy útil. (De hecho, incluso el caso de estudio de un cortafuego de red del próximo capítulo utiliza DHCP. No sólo eso, ¡incluye un servidor DHCP!) Configurar DHCP es tan sencillo como utilizar el programa netconfig de Red Hat y activar DHCP. Si prefiere realizar esta tarea manualmente, puede editar el archivo /etc/sysconfig/network-scripts/ifcfg-eth0 (reemplazando "eth0" por el dispositivo Ethernet real que utiliza su sistema, si es necesario). Este archivo simplemente fija tres variables: DEVICE, ONBOOT y BOOTPROTO. Los valores deberían ser "eth0" (o el identificador de su dispositivo Ethernet real), "yes" y "dhcp" respectivamente. Esto ya debería resultarle familiar.

La primera variable fija el dispositivo Ethernet y debería corresponderse con el nombre del archivo (por ejemplo, no use `DEVICE=eth1` dentro del archivo ifcfg-eth0). La segunda variable determina si la tarjeta debe activarse cuando arranque el sistema o sólo cuando se utilice manualmente el comando `ifup` (como al ejecutar `ifup eth0`). La variable BOOTPROTO indica que protocolo se utilizará, que en este caso (y probablemente en la mayoría de los casos) es DHCP. Todo lo que hace realmente el programa netconfig de Red Hat es crear el archivo /etc/sysconfig/network-scripts/ifcfg-eth0 (aunque también modifica algún otro archivo).

Instalación del servicio portmap

El servicio de asignación de puertos (conocido como portmap) es una forma bastante antigua de una técnica de programación conocida como llamada a procedimiento remoto (*remote procedure call*, RPC). RPC es el término empleado cuando un programa que se ejecuta en una máquina accede a una función que reside en otra máquina, como si el "procedimiento remoto" fuera en realidad parte del programa local. El servicio portmap empleado como un mecanismo RPC es algo que raramente se utiliza hoy en día, pero hay algunos programas bastante populares que aún lo necesitan. De manera notable, los servicios NIS+ y NFS comentados en este capítulo necesitan portmap para poder funcionar. Por ello, tendrá que instalar portmap para utilizar esos servicios. Red Hat Linux 7.3 incluye una instalación de portmap en el paquete RPM portmap. Después de instalarlo, tendrá que habilitarlo mediante el programa chkconfig; vea el capítulo 4 para obtener más detalles.

Merece la pena hacer una pequeña aclaración relacionada con la seguridad. El servicio portmap tiene una larga historia de debilidades de seguridad. Por esta razón, es bastante razonable suponer que existen vulnerabilidades de seguridad adicionales que podrían ser explotadas en el futuro. Para impedir esta exposición, no debería ejecutar en absoluto el servicio portmap (ni ningún servicio) a no ser que realmente lo necesite. En otras palabras, no instale portmap a no ser que realmente necesite el soporte NIS y NFS comentado en este capítulo.

Uso de servicios de información de red (NIS)

El primer paso para utilizar un sistema es entrar en él. Esta sección describe cómo está configurado el sistema del caso de estudio para utilizar los servicios de información de red (NIS, o NIS+) para llevar a cabo la autenticación de usuario.

Normalmente, los sistemas Unix almacenan la información de usuario como el directorio local, la shell por defecto, y la contraseña en el archivo /etc/passwd. Sin embargo, si un usuario necesita entrar en varios sistemas, esa información tiene que replicarse en el archivo /etc/passwd de cada sistema. Disponer de múltiples copias de la misma información en varios sistemas supone un dolor de cabeza administrativo. NIS es la aspirina, la solución a este problema.

NIS es un sistema bastante simple, conceptualmente. En lo esencial, NIS es simplemente un modo de distribuir archivos a través de una red. Esto es principalmente útil para los archivos de configuración, como /etc/passwd. En otras palabras, NIS simplemente comparte archivos entre múltiples sistemas, de modo que se puede mantener únicamente la copia central, en lugar de mantener muchas copias en muchos sistemas. Un archivo de estos es el archivo passwd, que se utiliza en lugar de un archivo passwd local.

Obviamente, hay un solapamiento entre algunos de estos archivos. Por ejemplo, si la cuenta de un usuario está presente en ambas partes, ¿cuál tiene prioridad: el archivo /etc/passwd local del sistema o la versión que se encuentra en NIS? El archivo /etc/nsswitch.conf define esta prioridad. Por ejemplo, el archivo /etc/

nsswitch.conf por defecto de Red Hat Linux tiene esta línea, que define el orden para las contraseñas:

```
passwd: files nisplus nis
```

Las tres palabras tras "passwd:" hacen referencia a distintas fuentes para ese archivo. Que el identificador "files" se encuentre en primer lugar significa que el sistema debería consultar el archivo /etc/passwd local antes que las copias almacenadas en NIS+ ("nisplus") o que las del NIS más antiguo ("nis"). Los valores por defecto debería funcionar bien en la mayoría de los casos, pero puede modificar el orden para un determinado archivo si lo necesita (por ejemplo, podría cambiar el orden de manera que la información NIS de una cuenta sea más prioritaria que /etc/passwd). Si mira en el archivo /etc/nsswitch.conf, verá que hay otros subsistemas que también pueden cargar información desde NIS. Debería consultar la página del manual sobre nsswitch.conf para obtener más información.

Para que funcione este esquema, las bibliotecas del sistema tienen que haberse compilado para utilizar NIS como parte del proceso de autenticación. (Esto también es cierto para cualquier otro sistema que utilice archivos proporcionados por NIS, como aquellos mostrados en el archivo /etc/nsswitch.conf.) Esto es por lo que resulta difícil instalar el soporte para NIS después de que se haya instalado el sistema, del mismo modo que sería difícil acomodar del mismo modo PAM al sistema, como se comentó en el capítulo 9. En realidad, configurar su estación de trabajo para que utilice NIS es bastante simple con Red Hat Linux, ya que incluye de antemano todos los archivos que necesita. Para comenzar, necesita instalar dos paquetes: yp-tools e ypbind. (Anteriormente NIS era conocido como "páginas amarillas" [*yellow pages*] y cambió su nombre debido a una cuestión de marcas comerciales, por eso algunos de sus archivos aún utilizan la abreviatura "yp".)

Una vez que haya instalado estos paquetes, debe configurar NIS para que sea capaz de encontrar los servidores. Por lo general, esta tarea se lleva a cabo mediante dos archivos de configuración: /etc/sysconfig/network y /etc/yp.conf. El archivo /etc/sysconfig/network ya debería existir en su sistema; el archivo /etc/yp.conf se instalará desde el paquete ypbind.

Modificación del archivo /etc/sysconfig/network

Como recordará del capítulo 4, el archivo /etc/sysconfig/network contiene los valores para variables importantes que utilizan los guiones de configuración de la red. Por ejemplo, la variable HOSTNAME fija el nombre de máquina en la red del sistema. La variable relevante para NIS es NISDOMAIN, que determina el nombre del dominio NIS. El dominio NIS es simplemente un nombre lógico para una configuración particular de NIS. Una red dada puede tener en funcionamiento muchos servidores NIS, cada uno de ellos atendiendo a un conjunto diferente de usuarios. Estas configuraciones se denominan dominios, y un cliente tiene que quedar ligado a un dominio específico de la red. Este es el dominio que se especifica mediante la variable NISDOMAIN. Su administrador de red tendrá que indicarle el nombre del dominio NIS que debería utilizar. En el sistema del caso de estudio, el nombre de

este dominio es "universal", y por eso la línea correspondiente del archivo /etc/sysconfig/network es simplemente `NISDOMAIN=universal`.

Modificación del archivo /etc/yp.conf

Típicamente este archivo tiene una sola línea, que indica al software cliente NIS donde encontrar el servidor NIS. Hay tres posibles maneras de crear esta línea, y todas ellas están documentadas en comentarios dentro del archivo /etc/yp.conf. Una manera es simplemente especificar un nombre de la máquina NIS que se utilizará como servidor para un dominio dado. Otra manera es hacer que el cliente transmita una petición a toda la red para encontrar un servidor NIS para el dominio. La tercera opción es incrustar la dirección de un servidor NIS, sin importar el dominio. Por ello necesitará que el administrador de su red le proporcione dos piezas de información: el dominio NIS (del que ya dispone desde la sección anterior) y posiblemente el nombre de la máquina del propio servidor NIS. Eso, en realidad, lleva a una sutileza bastante extraña de la configuración de NIS.

El asunto es que el formato que utiliza en el archivo /etc/yp.conf puede variar de acuerdo con el modo en que está configurada su red. Por ejemplo, el sistema del caso de estudio tiene esta línea:

```
domain universal server nis01
```

Sin embargo, el sistema del caso de estudio es en realidad una de las diferentes configuraciones idénticas de la red. O, mejor dicho, casi idénticas. Estos otros sistemas se diferencian solamente en sus archivos /etc/yp.conf. Muchos de ellos disponen de esta línea, en lugar de la línea anterior:

```
domain universal broadcast
```

El motivo de esta diferencia parece deberse a las configuraciones de encaminamiento de la red. Estos sistemas se encuentran en diversas partes del edificio y por eso forman parte de diversas subredes de la LAN. Esto significa que el servidor NIS será accesible a algunas estaciones de trabajo mediante el proceso de difusión, pero no estará disponible para otras. Cuando se añade un nuevo sistema, frecuentemente habrá que crear el archivo /etc/yp.conf mediante el procedimiento de prueba y error. Puede que no encuentre este problema en su red, pero igualmente podría encontrarlo. Simplemente asegúrese de probar todas las posibilidades cuando intente configurar su sistema.

Después de que haya configurado el sistema NIS para que pueda comunicarse con el servidor NIS, ya estará todo listo. El sistema NIS cargará normalmente de manera automática los archivos de configuración que estén disponibles (como el archivo de información de usuario passwd) en el servidor NIS.

Uso de servicios del sistema de archivos de red (NFS)

NIS proporciona un mecanismo centralizado de autenticación. El equivalente de esto es el servicio NFS, que proporciona un almacenamiento centralizado de datos.

Típicamente, los directorios locales de los usuarios se guardan en servidores NFS, los cuales montan otros ordenadores cuando los usuarios entran en el sistema. Por eso, los datos de los usuarios los "siguen" por la red. En algunas ocasiones, se crean directorios compartidos específicos para cada proyecto para permitir a los usuarios trabajar sobre archivos comunes de una manera más fácil. Esta sección describe cómo está configurado NFS para este caso de estudio.

En general, NFS está bastante ligado a NIS. Es decir, probablemente el modo más conveniente de distribuir la información de configuración NFS sea mediante NIS. (Hay un motivo para que el acronimo NIS venga de *Network Information Services* y no se haya creado un acrónimo sólo a partir de *User Information Services*.) La funcionalidad principal de NFS en los sistemas Linux se encuentra en los módulos del núcleo que implementan NFS. Hay dos de estos módulos: uno es un servidor NFS y el otro es el cliente. El módulo servidor no se utiliza en el caso de estudio (ya que no se trata de un servidor NFS).

El módulo del núcleo del cliente es muy sencillo: es simplemente un controlador del sistema de archivos. Es decir, implementa un sistema de archivos empleando el marco de trabajo estándar del sistema de archivos virtual (*Virtual Filesystem*, VFS) de Linux, precisamente del mismo modo que en que se implementan los sistemas de archivo estándar ext3 de Linux e is9660 de CD-ROM. Puede montar y desmontar sistemas de archivos NFS del mismo modo en que lo haría con cualquier otro sistema de archivos, empleando "nfs" como el tipo para la opción `-t` que se pasa a `mount`.

En otras palabras, el cliente NIS autentica a un usuario empleando información extraída de un servidor NIS, y también recupera la información sobre la cuenta del usuario, como la ubicación del directorio local. Después de esto, el núcleo simplemente monta el directorio NFS del usuario como un sistema de archivos normal.

Nota

La implementación de NFS que emplea Linux es notablemente remilgada, y se han encontrado extraños efectos con respecto a las implementaciones de NFS de algunos fabricantes (a veces debido a errores en los productos de otros fabricantes más que en Linux). Si encuentra problemas extraños que parezcan estar relacionados con NFS, compruebe la documentación sobre errores de su distribución, ya que puede que no sea la única víctima. La distribución Red Hat Linux se ha visto afectada por problemas de este tipo en versiones tan actuales como la versión 7.3.

Preparación de los servicios de apoyo

El soporte central de NFS está confinado en el núcleo. Sin embargo, hay un par de servicios de apoyo que son necesarios para utilizar NFS en la práctica. El primero es el servicio portmap, que ya he comentado. No obstante, hay dos servicios

adicionales que son específicos de NFS (y que utilizan portmap): el demonio de bloqueo (*locking*) NFS y el inspector de servicio de red (*Network Service Monitor*, NSM) NFS.

El demonio de bloqueo de archivos NFS es un servicio RPC que emplea el servicio portmap. Se utiliza para manejar los bloqueos de archivos distribuidos mediante NFS. El bloqueo de archivos impide que dos usuarios accedan a un archivo al mismo tiempo, ya que sólo un usuario puede "bloquear" un archivo en cada momento. El demonio de bloqueo NFS implementa este comportamiento para los clientes NFS. (Tal y como sucede, el problema del bloqueo de archivos distribuidos no es un problema de fácil resolución, y este demonio tiene sus limitaciones. Aún así, funciona razonablemente bien la mayor parte del tiempo, al menos para los sistemas de escritorio.) El programa realmente relevante es /sbin/rpc.lockd, que forma parte del paquete RPM nfs-utils.

El NSM NFS está implementado por un programa que se llama /sbin/rpc.statd. Este programa, al igual que el demonio de bloqueo, es también un programa RPC que funciona junto con el servicio portmap. Este programa se utiliza para vigilar el estado de los servidores NFS, prestando especial atención a su reinicio. Lo utiliza el demonio de bloqueo para mantener todo en sincronía con los servidores. Este programa también se incluye en el paquete nfs-utils.

Estos dos demonios se encuentran generalmente emparejados: es decir, con toda probabilidad nunca necesite uno independientemente del otro. Por eso, el paquete nfs-utils gestiona ambos desde un único servicio SysV llamado nfslock. Para instalarlo, simplemente instale el paquete nfs-utils y utilice después el comando `chkconfig` (comentado en el capítulo 4) para habilitar el servicio nfslock recién instalado para su nivel de ejecución (normalmente los niveles 3 o 5). Ese único servicio gestionará ambos demonios de apoyo NFS.

Compresión del montador automático

La auténtica cuestión radica en cómo se montan los directorios. Después de todo, el módulo del núcleo puede ser un controlador de sistema de archivos, pero en algún punto tiene que haber un programa que monte realmente el directorio local del usuario. ¿Qué programa se encarga de ello?

La respuesta es el montador automático, conocido como servicio autofs. Este es un programa demonio que monta directorios NFS (u otros) bajo petición en el momento en que realmente se acceda a ellos. Esto resulta tan conveniente como eficiente. Es conveniente porque los usuarios (y los administradores) no tiene que montar manualmente directorios locales, discos compartidos de proyectos, y cosas así. Es eficiente porque el montaje de estos directorios bajo petición significa que no tienen que montarse desde el principio, cuando el sistema arranque, desperdiciando recursos cuando no se están utilizando.

En el caso de este sistema de muestra, el servicio autofs comienza a actuar a partir de la información recogida mediante NIS. (Vea el ejemplo de la entrada "automount" en el archivo nsswitch.conf.) Cuando un usuario entra en el sistema, por ejemplo, se hará referencia al directorio local del usuario (como cuando la shell

intente entrar en el directorio local del usuario tras su entrada en el sistema). El montador automático detectará que se ha hecho referencia al directorio y lo cargará automáticamente.

Hay un servicio relevante más del que debería estar al tanto: el servicio netfs. Este es un simple guión de inicio SysV que Red Hat Linux proporciona para habilitar todos los sistemas de archivos en red. (Esto incluye no sólo a NFS, sino también a otros protocolos, como SMB y NCP.) Este servicio montará cualquier sistema de archivos NFS configurados, pero no provocará que se monten los sistemas de archivos que permitan su montaje automático. (En este caso de estudio, todos los sistemas de archivos NFS se montan automáticamente, así que el servicio netfs no hace gran cosa, pero es importante conocerlo por si acaso su situación es distinta, lo que resulta bastante posible.)

La instalación de NFS y NIS es con frecuencia muy fácil, en especial con las versiones actuales de Red Hat Linux. Sin embargo, no es infalible al cien por cien, y a veces algo sale mal, en especial con los casos más sofisticados o con la administración menos centralizada. Además, determinadas peculiaridades en la implementación Linux de NFS que pueden provocar rompederos de cabeza, así que asegúrese siempre de utilizar la versión correcta. (Los núcleos Linux de la serie 2.4 incluyen soporte para las versiones 2 y 3 de NFS; asegúrese de emplear la versión correcta para su red mediante los comandos `lsmod` y `modprobe`.)

La conclusión final de todo esto es que configurar NFS suele ser muy sencillo una vez que ha configurado NIS adecuadamente. En esencia, simplemente necesita instalar los servicios netfs y autofs y configurarlos para que se inicien cuando arranque el sistema mediante el comando `chkconfig` descrito en el capítulo 4. Sin embargo, NFS y NIS son sistemas bastante complicados, y este libro no dispone de espacio suficiente para comentarlos completamente. Esta sección simplemente ha dibujado una configuración particular: la utilizada por el sistema de este caso de estudio. Hay otras muchas maneras de configurar estos servicios. Consulte con el administrador de su red en busca de información y ayuda.

Nota: Particularización del montador automático

La configuración de NIS y NFS de este capítulo está optimizada para la red particular en la que reside el sistema del caso de estudio. En este caso, las configuraciones de NFS y autofs se almacenan mediante NIS. Sin embargo, su entorno puede utilizar una técnica más simple, y puede tener que configurar manualmente NFS modificando los archivos /etc/auto.master y /etc/fstab.

El archivo /etc/auto.master es la configuración real para autofs y puede utilizarse para montar automáticamente otros dispositivos (como disqueteras normales y disqueteras ZIP) mediante el montador automático. (En esencia, en el sistema del caso de estudio lo que hay almacenado en NIS es una versión del archivo /etc/auto.master.) El archivo /etc/fstab puede usarse para configurar para su montaje directorios NFS específicos. Si desea más información, consulte la página del manual sobre autofs, los archivos prácticos (*how-to*)

sobre el montaje automático (o *Automount*) y la documentación sobre NIS/NIS+ del sitio web del Linux Documentation Project (www.linuxdoc.org) y de Red Hat (www.redhat.com).

Administración de activos

Un problema que tienen los administradores con respecto a las grandes redes es simplemente el seguir la trayectoria de los sistemas. Con frecuencia los usuarios actualizarán, eliminarán o añadirán sistemas a una red, a veces sin el conocimiento de la organización de soporte. El grupo de soporte, por supuesto, no puede soportar algo que desconoce, y por eso esto puede representar un problema.

Para ayudar a resolver esta situación, muchos administradores de red utilizan el así llamado software de administración de activos (*asset management software*). Estos son programas instalados en estaciones de trabajo y a veces en servidores que proporcionan simplemente un modo de que los administradores pueden comprobar qué máquinas hay en una red y cuáles son las especificaciones de esas máquinas.

Estos programas son típicamente software comercial, así que no se suele disponer del código fuente. Además, este tipo de programas normalmente se escribieron en primer lugar para otros sistemas operativos, y sólo recientemente han comenzado a ser adaptados a Linux. En consecuencia, ocasionalmente adolecen de ciertas peculiaridades molestas, como restricciones en cuanto a dónde pueden instalarse.

Si el grupo de soporte de su red emplea un software similar, puede o no tener muchas indicaciones sobre dónde debe instalarse. Sin embargo, si dispone de indicaciones, debería recordar que se trata únicamente de una pieza de software como cualquier otra. Idealmente, se le proporcionará un RPM para su instalación, pero si no resulta posible, un buen directorio de instalación sería /opt o /user/local.

El sistema del caso de estudio utiliza una herramienta llamada Asset Insight (producida por Tangram Enterprises Solutions, cuyo sitio web se encuentra en www.tangram.com) como software de seguimiento de activos. La versión para Linux de este software es un proceso bastante liviano que rastrea determinados aspectos del sistema, como las especificaciones del sistema.

Provisión de acceso remoto como *root*

Los administradores de una red son los responsables de mantener esa red, al igual que las máquinas de esa red, en correcto funcionamiento. Obviamente no pueden cumplir con esta responsabilidad si no tienen la capacidad de administrar los sistemas individuales que forman parte de la red. Por ello, el sistema del caso de estudio está configurado para ofrecer acceso remoto como usuario *root* al grupo de soporte de red mediante el servidor SSH.

El primer paso, obviamente, es instalar el servidor OpenSSH. Esto ya se ha comentado en diversas ocasiones en este libro; en el capítulo 8 se ofrece un comen-

tario completo. El sistema del caso de estudio utiliza los paquetes OpenSSH proporcionados por Red Hat Linux 7.3.

El siguiente paso es añadir la clave adecuada al archivo /root/.ssh/authorized_keys. El grupo de soporte de red dispone de un par de claves SSH. La clave pública de este par se instala en todos los sistemas Unix de la red para permitir el acceso seguro a aquellos sistemas a quienes puedan acceder a la clave privada correspondiente. La clave privada está controlada por el grupo de soporte de red.

Por eso, ofrecer acceso remoto como superusuario es tan sencillo como copiar la clave pública proporcionada por el grupo de soporte en el archivo .ssh/authorized_keys en el directorio local del usuario *root*; específicamente, el archivo será /root/.ssh/authorized_keys, ya que /root es el directorio local del usuario *root*.

Una vez que la clave esté instalada, es una buena idea verificar la clave con el grupo de soporte, empleando alguna técnica "fuera de banda" (es decir, sin emplear la red), como una simple llamada telefónica. De este modo se asegurará de que no va a instalar una clave comprometedora enviada por un atacante. Después de que haya instalado y verificado la clave, el grupo de soporte de red dispondrá de un acceso administrativo seguro a su estación de trabajo.

Ahora que ha instalado algunas herramientas básicas para permitir que su estación de trabajo viva pacíficamente en la red, es hora de comenzar a instalar el software que realmente necesita para realizar su trabajo. La siguiente sección describe cómo instalar el entorno de desarrollo empleado en este caso de estudio.

Entorno de desarrollo

Una vez que nos hemos quitado de en medio las tareas básicas de preparar el sistema para su funcionamiento en la red, es tiempo de dedicarse a la tarea de preparar el software para el que se pretende realmente utilizar esta máquina. Ese es el tema de esta sección.

Instalación de las herramientas centrales de desarrollo

Como ya comenté al comienzo de este capítulo, el sistema de este caso de estudio esta principalmente orientado hacia su uso como plataforma para el desarrollo de aplicaciones sobre Java 2 Enterprise Edition (J2EE). Sin embargo, resultará muy útil disponer de algunas herramientas adicionales que no están estrictamente relacionadas con Java.

La tabla 15.1 muestra la lista de diversos programas habituales, sus propósitos y el paquete RPM en el que se incluyen. En general, estas herramientas resultarán extremadamente familiares a los desarrolladores software (cuyos trabajos giran en torno a ellas) y familiares a los administradores de sistemas (que frecuentemente

tendrán que emplearlas para instalar software, tal y como se comenta en los capítulos 8 a 13).

Tabla 15.1. Herramientas de desarrollo software centrales.

Herramienta	Paquete RPM	Descripción
gcc	gcc-2.96-110.i386.rpm	Conjunto de compiladores GNU; incluye compiladores para lenguajes habituales como C, C++, Fortran, y otros.
libstdc++	libstdc++-2.96-110.i386.rpm	Bibliotecas centrales para los programas escritos en C++ y el compilador.
make	make-3.79.1-8.i386.rpm	Una herramienta de automatización del proceso de compilación.
flex	flex-2.5.4a-23.i386.rpm	La versión del proyecto GNU del analizador léxico lex.
bison	bison-1.35-1.i386.rpm	La versión del proyecto GNU del generador de compiladores yacc.
strace	strace-4.4-4.i386.rpm	Un programa para depurar llamadas de sistema.
ltrace	ltrace-0.3.10-7.i386.rpm	Un programa para depurar llamadas a bibliotecas compartidas.

Configuración del entorno Java

Instalar el entorno Java es obviamente la parte más importante de la construcción de una estación de trabajo de desarrollo en Java. Esta sección describe los aspectos básicos del entorno en el sistema del caso de estudio. Las siguientes secciones describen software adicional ya instalado sobre la base del entorno básico.

El entorno básico es precisamente el detallado en el capítulo 13. A modo de rápido repaso, ese capítulo comenta cómo crear un entorno de tal modo que se puedan instalar muchos JDK junto (posiblemente incluso de distintos fabricantes). Es sencillo conmutar entre estas instalaciones independientes, y el entorno también incluye un mecanismo para fijar automáticamente variables de entorno (en particular la variable CLASSPATH) en las shell de los usuarios. Este entorno es el núcleo del caso de estudio de este capítulo, así que debería estar absolutamente familiarizado con los contenidos del capítulo 13.

Instalación de herramientas adicionales

Gran parte de la potencia del lenguaje de programación Java proviene de la amplia variedad de herramientas y bibliotecas que hay disponibles. Los

desarrolladores utilizan estas bibliotecas para incrementar su productividad y compartir tecnologías y código. Esta sección comenta algunas de estas herramientas y describe cómo añadirlas al marco de trabajo del entorno Java básico preparado en el capítulo 13.

Automatización de la construcción con ant

Previamente, en la sección "Instalación de las herramientas centrales de desarrollo" de este capítulo, se mencionó la herramienta make como una de las herramientas centrales de desarrollo. Es cierto, make es un programa ya venerable, extremadamente útil. Sin embargo, por una gran cantidad de razones, no funciona perfectamente con los programas Java. Una alternativa popular dentro de la comunidad Java es la herramienta ant, disponible en jakarta.apache.org/ant/index.html.

La herramienta ant es muy parecida a make, a alto nivel. La diferencia más inmediata es que el formato de sus archivos de construcción está en XML (*eXtensible Markup Language*), como oposición al uso que hace make de un formato de archivo a medida. Una vez que se ahonda más, se encuentran ventajas adicionales en ant que lo hacen más útil para las aplicaciones Java; de manera notable, resulta extremadamente programable, lo que significa que los desarrolladores pueden extenderlo e incorporarlo en sus propias aplicaciones Java.

Por supuesto, este es un libro sobre sistemas Linux, no sobre herramientas de programación Java, así que no puedo entrar en más detalle de lo que ya lo he hecho. Si necesita ant, necesitará saber cómo se instala. Afortunadamente, es bastante sencillo. Debería comenzar consiguiendo una distribución binaria de ant desde del sitio web jakarta.apache.org/ant/index.html. (Puede descargar y construir su propia instalación a partir del código fuente, pero con aplicaciones Java como ant esto es algo que carece de sentido, ya que los archivos binarios Java están diseñados para funcionar sobre todas las máquinas.) Rápidamente descubrirá que la distribución de ant es un archivo *tarball* (por ejemplo, un archivo .tar.gz). Para instalarlo, simplemente necesita extraer el archivo mediante un comando como `tar zxf jakarta-ant-1.4.1.tar.gz`. Este archivo guarda los archivos de su propio subdirectorio, por eso cuando extraiga el archivo, sus contenidos se colocarán en un subdirectorio jakarta-ant-1.4.1 de su directorio de trabajo actual. Todo esto resulta fácil y conocido; el verdadero reto es decidir dónde instalarlo.

En este caso de estudio, todas las rutas llevan a /opt/java. Recordará del capítulo 3 que el directorio /opt está destinado a instalaciones particulares o específicas del sistema de paquetes completos de software. En este caso, va a utilizar /opt/java para contener todo el software relacionado con Java. Un objetivo es mantener el software similar en la misma ubicación, para facilitar y hacer más lógica su administración al igual que su uso. Otro objetivo es permitir la facultad de actualizar paquetes de una manera irregular, sin tener que eliminar versiones antiguas. (Por diversos motivos, frecuentemente los desarrolladores Java necesitan utilizar múltiples versiones del JDK a la vez.) Todo esto, por supuesto, sería imposible si los contenidos de estos paquetes de software estuvieran mezclados en el directorio /usr/local. Si desea más detalles sobre este tema, consulte los capítulos 3 a 7.

En consecuencia, en el sistema del caso de estudio, la herramienta ant está instalada en /opt/java/ant. Esta es la elección lógica, ya que el JDK está instalado en /opt/java/jdk. (Concretamente, /opt/java/jdk es un enlace simbólico al JDK "por defecto" de entre los diversos JDK potenciales que están instalados en un momento dado; véase el capítulo 13 para obtener más detalles.) Ya que la propia herramienta ant incluye guiones de shell y otros archivos que serán ejecutados, se necesita el directorio /opt/java/ant/bin en las variables de entorno PATH de los usuarios. Esto significa que el guión java.sh comentado en el capítulo 13 necesita que se le añadan unas líneas para este caso de estudio. La versión incrementada de este archivo se incluye como listado 15.1 en una muestra de código más adelante en este capítulo, dentro de la sección "Actualización de los guiones de entorno". Realmente esto es todo lo que atañe a la instalación de ant. Utilizar esta herramienta, por supuesto, no es tan fácil, pero ¡este no es un libro sobre programación Java! Aún así, esperemos que el proceso de instalación ya le resulte bastante sencillo.

Desarrollo de software Java con JBuilder

El entorno Java básico tal y como se describe en el capítulo 13 es bastante funcional. Los kits de desarrollo de Java instalados incluyen todas las herramientas que son técnicamente necesarias para crear e instalar software. Sin embargo, muchos desarrolladores prefirieren utilizar herramientas mejoradas conocidas como entornos de desarrollo integrado (*integrated development environments*, IDE), que proporcionan diversas ayudas a la productividad. Una herramienta popular utilizada en el sistema de este caso de estudio es JBuilder de Borland. Esta sección comenta cómo instalar este software, que puede encontrarse en www.borland.com/jbuilder/.

JBuilder es software comercial, así que no hay una distribución de código fuente. Además, JBuilder incluye un programa que instala el software, en lugar de un paquete RPM o *tarball* (en otras palabras, JBuilder es una aplicación software comercial típica). Normalmente, JBuilder se instala desde un CD o un programa que se descarga. JBuilder proporciona detalles precisos sobre la mecánica de este proceso, que es muy parecido al proceso de instalación de OpenOffice que se comentó en el capítulo 14.

Al igual que lo que se comentó sobre ant con anterioridad en este capítulo, la verdadera pregunta desde un punto de vista administrativo es dónde debería instalarse JBuilder. Por fortuna, responder es tan fácil como con ant: /opt/java/jbuilder. No obstante, hay un detalle adicional: puede ser una buena idea instalar JBuilder en un directorio cuyo nombre incluya la versión, como /opt/java/jbuilder6 para la versión 6. Este detalle se debe a que herramientas como JBuilder, al igual que los JDK, puede que en ocasiones necesiten tener instaladas múltiples versiones al mismo tiempo. (Podría hacer lo mismo también con ant: sin embargo, en el sistema del caso de estudio no se comprobó que fuera necesario.)

Una vez que está instalado JBuilder, tiene que configurarse. Al igual que ant, JBuilder incluye guiones para arrancar la aplicación que necesitan estar incluidos en las variables de entorno PATH de los usuarios. El listado que se encuentra en la sección "Actualización de los guiones de entorno" más adelante en este mismo

capítulo contiene una versión actualizada del archivo java.sh que incluye soporte para ant y también para JBuilder.

Instalación de bibliotecas Java adicionales

Uno de los ingredientes básicos de los desarrolladores Java son los archivos JAR estándar, que contienen bibliotecas de código que los desarrolladores pueden reutilizar en sus aplicaciones. Algunos de estos archivos JAR contienen software de código abierto, mientras que otros contienen software comercial, pero en cualquier caso funcionan del mismo modo. Se necesita añadir los archivos JAR a la variable de entorno CLASSPATH para que las aplicaciones Java que los necesiten puedan encontrarlos. El entorno Java construido en el capítulo 13 ya soporta un mecanismo para hacer uso automáticamente de estos archivos JAR; esencialmente, cualquier archivo JAR que se encuentre en /opt/java/packages será añadido automáticamente a la variable CLASSPATH mediante los guiones de entrada al sistema.

Generalmente esto resulta bastante adecuado. No obstante, merece otra mención dentro del contexto de la instalación de archivos JAR adicionales. Si consigue un archivo JAR desde la web (o quizás construya uno por sí mismo), todo lo que necesita hacer es colocarlo en el directorio /opt/java/packages y se incluirá automáticamente en los entornos de los usuarios. En general, esto es todo lo que se necesita para instalar bibliotecas JAR, y por supuesto, este era el propósito para construir el guión.

Actualización de los guiones de entorno

El listado 13.1 del capítulo 13 es un guión que configura los entornos de entrada al sistema de los usuarios para aprovechar la instalación Java en el sistema. Como comenté con anterioridad en esta sección, este guión necesita ser actualizado para soportar el software adicional.

El listado 15.1 es una versión actualizada de /etc/profile.d/java.sh. (Al igual que con el listado 13.1, traducir el guión a otros lenguajes de shell debería ser muy fácil, así que no se comentará aquí.) La única diferencia significativa es el añadido de dos bloques hacia el final del listado que añaden ant y JBuilder a las variables PATH de los usuarios.

Listado 15.1. Archivo /etc/profile.d/java.sh actualizado.

```
#!/bin/sh

# fija JAVA_HOME al valor por defecto, a no ser que ya esté fijado
if [[ "X"${JAVA_HOME} == "X" ]]; then
    JAVA_HOME=/opt/java/jdk
fi

# añade el directorio bin de JDK a PATH (a la ruta por defecto)
PATH=${JAVA_HOME}/bin:${PATH}

# fija CLASSPATH; con el valor inicial de "." (el directorio de trabajo
actual)
```

```
CLASSPATH=.
for jar in /opt/java/packages/*; do
    if [[ "${jar}" == "/opt/java/packages/*" ]]; then
        break;
    fi
    CLASSPATH=${CLASSPATH}:${jar}
done

# añade cualquier JAR que esté en el propio directorio de paquetes java del
usuario
if [ -d ${HOME}/java/packages ]; then
    for jar in ${HOME}/java/packages/*; do
        if [[ "${jar}" == "${HOME}/java/packages/*" ]]; then
            break;
        fi
        CLASSPATH=${CLASSPATH}:${jar}
        done
fi

# añade JBuilder y ant a la variable PATH si es que están instalados
if [ -d /opt/java/jbuilder ]; then
    PATH=${PATH}:/opt/java/jbuilder/bin
fi
if [ -d /opt/java/ant ]; then
    PATH=${PATH}:/opt/java/ant/bin
fi

export PATH JAVA_HOME CLASSPATH
```

Advertencias y mejoras

La información que acaba de leer funciona bastante bien en la mayoría de las circunstancias. Sin embargo, hay ocasiones en que podría dar pie a algunos problemas. Esta sección señala algunas pegas que podría encontrar.

Uso de muchos archivos JAR

El guión del listado 15.1 añade archivos JAR a las variables de entorno CLASSPATH de los usuarios automáticamente. Debido al modo en que funciona el mecanismo de rutas de clases (*classpath*) de Java, hay que especificar estos archivos JAR con sus nombres de rutas completos, incluso aunque se encuentren en el mismo directorio. Por ejemplo, la siguiente variable CLASSPATH fija dos archivos JAR, foo.jar y bar.jar:

```
export CLASSPATH=/opt/java/packages/foo.jar:/opt/java/packages/bar.jar
```

El problema de este esquema es que si se colocan muchos archivos JAR en /opt/java/packages (o en el directorio personal del usuario ~/java/packages), el valor de la variable CLASSPATH puede ser muy largo. Si se vuelve muy largo, sobrepasará el límite máximo que la shell (como /bin/bash) permite para las variables de entorno. Si sucede esto, puede que necesite emplear una técnica diferente.

La máquina virtual Java (*Java Virtual Machine*, JVM) de Sun también soporta un mecanismo conocido como el sistema de paquetes opcionales o extensiones estándar.

En esencia, este mecanismo implica especificar uno o más directorios que contengan archivos JAR mediante una opción en línea de comandos (que es la opción `-Djava.ext.dirs`) que se le pasa al programa Java. Aquí hay un ejemplo de este mecanismo:

```
$ java -Djava.ext.dirs=/opt/ava/packages:${HOME}/java/packages
<nombre_de_clase>
```

Lamentablemente, este mecanismo sólo puede utilizarse desde la línea de comandos. Esto significa que no hay un modo sencillo de configurar los entornos de entrada al sistema de los usuarios para que empleen automáticamente el mecanismo de extensión de Java. Sin embargo, puede escribir un guión ejecutable que realice la tarea. Este guión debería colocarse en un directorio que estuviera en la variable PATH de los usuarios antes que el directorio /opt/java/jdk/bin, de modo que el guión se encuentre en primer lugar (de otro modo, el programa `java` estándar del JDK se ejecutaría en primer lugar). Una posibilidad podría ser colocar el guión en el archivo /usr/local/java. El listado 15.2 contiene un ejemplo de guión /usr/local/java. El guión simplemente fija el argumento `-Djava.ext.dirs` y después ejecuta el programa `java` normal del JDK.

Listado 15.2. Guión opcional java de sustitución.

```
#!/bin/sh

# instale este guión como /usr/local/java

# fija JAVA_HOME al valor por defecto, a no ser que ya esté fijado
if [[ "X"${JAVA_HOME} == "X" ]]; then
    JAVA_HOME=/opt/java/jdk
fi

# ${JAVA_HOME}/jre/lib del JDK es el mínimo valor que necesita -
Djava.ext.dirs
EXTDIR=${JAVA_HOME}/jre/lib@

# Añade /opt/java/packages si existe
if [ -d /opt/java/packages ]; then
    EXTDIR=/opt/java/packages@
fi

# Añade ~/java/packages si existe
if [ -d ${HOME}/java/packages ]; then
    EXTDIR=${EXTDIR}${HOME}/java/packages
fi

# Esto hace que "@" se convierta en " ", y después " " se convierta en ":"
# Se hace para eliminar cualquier ":" espurio al final del valor
EXTDIR=`echo ${EXTDIR} | sed 's/@/ /g'`
EXTDIR=`echo ${EXTDIR} | sed 's/ /:/g'`

# finalmente, ejecutar el verdadero bin/java
exec ${JAVA_HOME}/bin/java -Djava.ext.dirs=${EXTDIR} $*
```

Gestión de propiedad y permisos

Una cosa que podría descubrir mientras utiliza este caso de estudio es que a veces puede ser muy complicado de administrar. Los programadores JAVA son a veces "de alto mantenimiento", lo que significa que frecuentemente necesitan instalar o actualizar nuevos JDK, añadir o eliminar paquetes, instalar herramientas adicionales y cosas así. En otras palabras, los desarrolladores frecuentemente necesitan añadir y eliminar cosas de /opt/java de manera individual. Este tipo de peticiones constantes de instalaciones y desinstalaciones puede convertirse en una sobrecarga de trabajo bastante importante para el administrador (o para el propio desarrollador, si administra el mismo su sistema). Si los desarrolladores son autosuficientes, puede que desee permitirles gestionar su propia instalación Java.

Hay en general dos maneras de hacer esto. Obviamente, los usuarios podrían replicar simplemente la configuración de /opt/java por sí mismos, sólo que en su propio directorio local. Si tiene múltiples usuarios, sin embargo, esto supone un desperdicio de espacio ya que cada desarrollador tendrá su propia copia de los archivos JAR y las instalaciones de software relevantes.

Otra alternativa es asignar la propiedad de grupo del directorio /opt/java a un grupo en el que se incluyan todos sus usuarios Java y después hacer que el grupo pueda escribir en el directorio. Si sigue este camino, también tendrá que fijar el "bit adhesivo" para el grupo (mediante el comando `chmod g+s`, comentado en el capítulo 12). La parte negativa de este enfoque, sin embargo, es que los usuarios tendrán ahora privilegios de escritura sobre al menos un directorio del directorio /opt, lo que puede resultar desagradable para algunos administradores.

Ambas soluciones a estos problemas tienen sus pros y sus contras. Tendrá que escoger el compromiso más apropiado para su situación. Puede pensar que la instalación básica ya resulta adecuada tal y como es, o podría descubrir que necesita utilizar una de estas técnicas opcionales. ¡Depende de usted!

Uso del servidor HTTP Apache

Este caso de estudio está orientado a su utilización para el desarrollo de aplicaciones J2EE. Típicamente, esto significa construir aplicaciones web. Con frecuencia, los desarrolladores que realizan este tipo de trabajo utilizan el servidor HTTP Apache como su servidor web. De hecho, este caso de estudio es uno de estos sistemas. Esta sección describe la configuración de Apache. En realidad, no hay gran cosa que describir. La configuración empleada en el sistema del caso de estudio es idéntica a la descrita en el capítulo 11 (ese capítulo comenta cómo instalar Apache a partir de su código fuente). El único añadido a la configuración de ese capítulo es que se instaló la extensión del motor de servlets Tomcat mediante el comando `apxs`.

Ejecución de aplicaciones J2EE con Tomcat

Desarrollar aplicaciones J2EE requiere el software de servidor J2EE apropiado. Sin embargo, Apache es únicamente un servidor HTTP, así que se necesita software

adicional para desarrollar y ejecutar aplicaciones J2EE. Típicamente, estos servidores se enlazarán con el servidor HTTP Apache a través de extensiones de Apache.

El sistema de este caso de estudio se usa principalmente para desarrollar aplicaciones basadas en Servlet y JSP (*Java Server Pages*). Otro tipo habitual de aplicación es el basado en EJB (*Enterprise JavaBeans*); sin embargo, esta configuración en particular no se utilizó para ese propósito.

El motor de servlets (pequeños programas que proporcionan un servicio) escogido fue Tomcat, que es producido por el proyecto Jakarta y está disponible en jakarta.apache.org/tomcat/index.html. (Si necesita un servidor EJB, puede desear comprobar la implementación de referencia de J2EE de Sun, que está disponible en java.sun.com/j2ee/download.html.)

Tomcat incluye realmente su propio servidor HTTP incorporado. Este servidor HTTP resulta perfectamente adecuado, puede que ni siquiera necesite Apache. Sin embargo, el caso de estudio utiliza a pesar de todo Apache, para reproducir mejor el entorno de producción en que el que se desplegarán las aplicaciones.

La instalación de Tomcat es bastante sencilla. Hay dos componentes: un servidor independiente escrito en Java, y una extensión para Apache que conecta el servidor HTTP con el servidor Java independiente. Debería estar bien preparado para hacerse cargo de ambas instalaciones. Instalar la extensión de Apache es algo que se realiza mediante la herramienta apxs estándar incluida con Apache y comentada en el capítulo 11. (Véase el capítulo 11 para obtener más detalles).

La instalación del programa servidor independiente también es bastante sencilla. En esencia, se reduce a extraer la distribución dentro del directorio que escoja y después modificar un archivo de configuración para el servidor. (Esto hace que la parte del servidor de Tomcat sea similar a la configuración de un servidor mediante una estructura de archivos plana parecida a la del servidor OpenSSH y comentada en el capítulo 8.) Siguiendo con el estándar empleado a lo largo de este caso de estudio, Tomcat se instaló en el directorio /opt/java/jakarta-tomcat.

Instalar Tomcat es fácil, una vez que sabe cómo se instala un servidor parecido como, por ejmeplo, OpenSSH. Los detalles son distintos (especialmente en lo que se refiere al archivo de configuración), pero una vez que comprende el modelo general, el archivo de configuración es sencillo. La documentación de Tomcat debería cubrir el resto de la información que necesita conocer.

Hasta aquí, ha leído acerca de cómo conseguir que su estación de trabajo esté conectada a la red y cómo configurar un entorno de desarrollo de software útil. Sin embargo, hay otras realidades adicionales relacionadas con trabajar en el mundo de la empresa. En la siguiente sección, leerá sobre algunas técnicas y piezas de software útiles que le ayudarán a navegar por los procelosos mares de los negocios.

Interoperación corporativa

Hay muchos aspectos en el mundo empresarial moderno, y lamentablemente muchos de ellos implican a los productos Windows de Microsoft. Por ejemplo,

muchas empresas crean y distribuyen sus documentos en los formatos que emplea Microsoft Office, y muchos negocios dependen de software que sólo existe para la plataforma Windows de Microsoft. Opine lo que opine de estas realidades, aún así son realidades y una estación de trabajo que pretenda tener éxito debe ser capaz de manejarlas. Esta sección comenta dos herramientas muy útiles para trabajar satisfactoriamente en un entorno corporativo: OpenOffice y VMware.

Uso de OpenOffice para documentos de empresa

La suite OpenOffice de herramientas de productividad ofimática es un sistema extremadamente capaz. Puede leer, modificar y escribir archivos en formatos de otras suites. Por ejemplo, puede producir documentos de texto en el formato que emplea Microsoft Word.

En algunas ocasiones no se produce una traducción cien por cien efectiva, y algunos pequeños detalles se producirán incorrectamente. De manera notable, frecuentemente las fuentes no realizan su transición correctamente (Linux y Windows utilizan distintos valores de puntos por pulgada [*dots per inch*, dpi] en sus pantallas, y por eso a veces los tamaños de las fuentes se procesan mal, en especial en presentaciones de dispositivas en las que el tamaño de la fuente forma parte de la pantalla.)

Sin embargo, OpenOffice sigue siendo una herramienta increíblemente útil, y es más que capaz de visualizar archivos de Microsoft Office. De hecho, si toda su organización quisiera adoptar OpenOffice como estándar es probable que no echase nada en falta. OpenOffice está instalado en el sistema del caso de estudio. OpenOffice también está instalado en el sistema doméstico de escritorio comentado en el capítulo 14, y este caso de estudio emplea las mismas técnicas. (Véase el capítulo 14 para obtener más detalles.)

Aún así, hay una diferencia, que es dónde se instala OpenOffice. Esta estación de trabajo existe en el contexto de una red corporativa, y los directorios locales de los usuarios se encuentran realmente en los servidores NFS centrales y se montan a través de la red.

Esto permite una gran flexibilidad, pero como contrapartida tiene un impacto importante sobre el rendimiento, ya que las operaciones de lectura y escritura a través de una red son mucho más lentas que las mismas operaciones realizadas sobre un disco duro. Para los datos no importa realmente, pero puede producir retrasos notables para grandes programas como OpenOffice.

Para evitar esto, se crea un directorio en el disco del sistema local y se asigna al usuario principal de la estación de trabajo. En el caso de estudio, este directorio es /opt/elopez. Entonces, el usuario puede instalar datos locales y software (como OpenOffice) en este directorio, y aprovecharse del rendimiento de los discos locales. Este requisito sólo existe en sistemas en los que los directorios locales de los usuarios (que es donde normalmente instalaría una cosa de este tipo) no se encuentren en un disco local. Aparte de este cambio menor, OpenOffice se instala precisamente como en el capítulo 14.

Funcionamiento de VMware

VMware, Inc., comercializa un producto conocido como VMware. VMware es un emulador de hardware o hardware virtual. Es decir, VMware crea una "maquina virtual" (*virtual machine*, VM) mediante software que funciona realmente en el sistema operativo de la máquina. Este ordenador virtual resulta completamente indistinguible de una máquina normal, independiente. Esto significa que puede instalar un sistema operativo completo dentro de esta máquina virtual. Por supuesto, antes de que pueda instalar y utilizar VMware, tendrá que adquirir una licencia para ello y obtener el software; puede realizar ambas tareas en el sitio web de VMware (www.vmware.com).

VMware dispone de una versión que funciona con los sistemas operativos Linux. Al instalar VMware y posteriormente una versión de Microsoft Windows dentro de la máquina virtual, puede ejecutar Windows literalmente en su ordenador al mismo tiempo que Linux. Esto permite que arranque Windows (u otro sistema operativo) cuando necesite utilizar alguna aplicación software que necesite su empresa y sólo exista para otro sistema operativo distinto a Linux. (Incluso podría emplear esta solución en lugar de OpenOffice, simplemente utilizando Microsoft Office desde Windows. Sin embargo, emplear OpenOffice es una solución más ligera. En el fondo, no deja de ser básicamente una cuestión de preferencias personales.)

VMware está empaquetado en varios formatos, incluido RPM. Este es probablemente el modo más fácil de instalar el software. El paquete RPM también incluye un guión de configuración (llamado vmware-config.pl) que configura las propiedades globales de la instalación, como habilitar la red para la máquina virtual, exponer el sistema de archivos del sistema operativo huésped a la máquina virtual y cosas así. Después de eso, cada máquina virtual independiente tiene su propia configuración, que se genera mediante una ventana de diálogo dentro del entorno gráfico de VMware. (No obstante, este archivo de configuración es un archivo de texto y puede editarse manualmente si así se desea.)

Aún así, VMware no es todo diversión y alegría. Encontrará una gran variedad de obstáculos en su camino. Por ejemplo, VMware necesita varios módulos del núcleo; VMware soporta estos módulos para un cierto conjunto de versiones del núcleo. Si está utilizando un núcleo particularizado o un núcleo actualizado más nuevo que los soportados por VMware, tendrá que reconstruir los módulos del núcleo de VMware, lo que podría no funcionar. Además, VMware tiene unos requisitos de hardware bastante significativos, y algunos componentes hardware gráficos de bajo nivel (como el hardware de gráficos incorporado en algunas placas base) pueden resultar inadecuados. Para problemas como estos, tendrá que consultar el sitio web de VMware para encontrar ayuda. Esto es en realidad todo lo que tiene que ver con el uso de VMware. Por supuesto, ya que tendrá un sistema operativo independiente y real ejecutándose "dentro" de VMware, tendrá que ser capaz de proteger y administrar igualmente ese sistema. (Por ejemplo, si utiliza Windows dentro de VMware, entonces esa instalación será vulnerable a los virus de Windows y a otras cuestiones relativas a la seguridad.)

Llegado a este punto, ha conseguido una estación de trabajo completamente funcional. Sin embargo, sería una lástima dejar que todo este duro trabajo (por no mencionar los datos potencialmente valiosos) pudiera perderse debido a un ataque hostil a su sistema. En la próxima sección aprenderá cómo asegurarse de que su nueva estación de trabajo sea razonablemente segura.

Cuestiones de seguridad

Ningún sistema informático es más importante que los datos que procesa y almacena, y por eso es tan importante proteger esos datos de un robo o de cualquier otro daño. Esta sección describe algunas cuestiones relacionadas con la seguridad en el sistema de este caso de estudio. Según vaya leyendo esta sección, es importante que tenga en mente su propia situación, porque a no ser que su entorno sea exactamente como este, tendrá que adaptar los detalles a su propia situación.

Compresión de la seguridad de la red local

Proteger una estación de trabajo que es parte de una red compleja como la del sistema del caso de estudio es bastante distinto de proteger un sistema de escritorio doméstico individual (como en el caso de estudio del capítulo 14). Después de todo, cuando el sistema confía tanto en las configuraciones y corrección de otros sistemas y otro software, se vuelve muy vulnerable a las debilidades que pueda haber en esos otros programas y sistemas.

Este caso de estudio utiliza NFS para almacenar los datos de los usuarios y NIS para autenticar a los usuarios y permitir su entrada en el sistema. Cada una de estas herramientas es extremadamente valiosa, pero lo comprometido es que pueden reducir la seguridad del sistema.

En el caso de NIS, la estación de trabajo es esencialmente una esclava del servidor NIS. Esto significa que si un atacante compromete los servidores de dominio NIS, ese atacante tendrá acceso a todos los servidores y estaciones de trabajo que estén configurados para autenticarse mediante NIS. En la jerga de la industria, este es un "punto único de fallo", y se debería temer en gran medida. Sin embargo, es un mal necesario.

NFS también puede ser un punto único de fallo; si el servidor de archivos en el que se almacenan los datos de los usuarios se ve comprometido, entonces todos estos datos estarán disponibles para el atacante. Sin embargo, incluso si el servidor NFS es completamente seguro, los clientes NFS aún pueden volverse vulnerables. Por ejemplo, ser un cliente NFS (como la estación de trabajo de este caso de estudio) implica la necesidad de los servicios portmap y nfs.statd. Estos servicios tienen un largo historial de vulnerabilidades de seguridad, y por eso ejecutarlos expone a las propias estaciones de trabajo a un ataque, posiblemente incluso en casos en los que el propio servidor sea seguro.

Lamentablemente, no hay gran cosa que las estaciones de trabajo puedan hacer al respecto. El valor de NIS y NFS (es decir, el valor de la administración y el almacenaje de datos centralizados) supera los riesgos de seguridad. Además, la mayoría de las redes de este tipo están protegidas por un cortafuego. El cortafuego impedirá ataques fortuitos procedentes de Internet, pero no protegerá frente a un intruso determinado que consiga acceso mediante otro medio, como a través de una red inalámbrica o un simple engaño. (Para obtener más información sobre este tema, puede que desee leer el siguiente capítulo, el capítulo 16, que muestra el caso de estudio de un cortafuego.)

En otras palabras, en el entorno de red del caso de estudio, las estaciones de trabajo sólo están tan seguras como los servidores y el software que hacen funcionar. Desafortunadamente, no hay mucho más que un administrador de una única estación de trabajo pueda hacer. Si su propio entorno utiliza software diferente (como Kerberos en lugar de NIS y AFS en lugar de NFS), entonces los problemas de seguridad serán también diferentes. Este no es un libro sobre la seguridad, por eso no hay espacio más que para decir esto: conozca su red, e incluso si no puede protegerla completamente, sepa qué puede y qué no puede proteger.

Protección de la estación de trabajo

Dadas las restricciones impuestas por el entorno de red, la estación de trabajo individual de este caso de estudio sólo puede protegerse hasta un cierto punto. Sin embargo, aún compensa tomar algunas medidas rudimentarias, que delinea esta sección. El objetivo principal de proteger su estación de trabajo es simplemente este: asegurarse de que su estación no sea la que permita que un atacante pueda comprometer la red. Dos tareas básicas servirán en gran medida para mantener su estación de trabajo segura: mantener actualizado su software y no inhabilitar el cortafuego.

El principal modo de ataque contra sistemas es explotar los defectos conocidos del software. La única defensa contra esto es asegurarse de que el software que ejecuta no es la misma versión que la que tratan de explotar los atacantes. Esto significa que mantener seguro su sistema es un proceso continuo, y que tiene que mantenerse al día con respecto a las actualizaciones que se hacen públicas para su distribución para asegurarse de actualizar su software con una versión libre de errores tan pronto como estos se descubran y corrijan.

Un modo de colaborar a reducir la vulnerabilidad de su sistema ante ataques (incluso si se han descubierto fallos) es mantener en funcionamiento el cortafuego local del sistema. (Red Hat Linux 7.3, por ejemplo, incluye una configuración que aplica un "cortafuego personal" al sistema.) Sin embargo, dependiendo de su entorno, puede que no sea práctico un cortafuego local (por ejemplo, puede bloquear el acceso a servicios necesarios, como portmap). Siempre puede ajustar la configuración del cortafuego (modificando el archivo /etc/sysconfig/ipchains), pero puede que no merezca la pena el esfuerzo en algunos casos. De hecho, la estación de trabajo real del caso de estudio no tiene activado el cortafuego estándar de Red Hat.

Al fin y al cabo, esto es todo lo que puede hacer cuando no está a cargo de toda la red (por supuesto, si está a cargo de toda la red, ¡dispone de una gran cantidad de opciones adicionales!). Puede encontrarse en la situación de tener que delegar la responsabilidad de la seguridad sobre el administrador de red, como es el caso del sistema del caso de estudio. ¡Ya está! Ya puede desarrollar software, exactamente del mismo modo en que yo lo hago. Por supuesto, esto no va a resultar especialmente útil si tiene necesidades diferentes. La siguiente sección señala algunas maneras en que podría ajustar este caso de estudio para adaptarlos a sus necesidades específicas. Estas ideas le proporcionarán las semillas que necesita para cultivar su propia estación de trabajo personalizada.

Configuraciones alternativas

Este capítulo ha descrito una estación de trabajo real utilizada para desarrollar un determinado tipo de aplicación web. Sus propias necesidades pueden ser distintas; esta sección comenta brevemente otro software que puede ser útil.

Uso de un servidor de aplicaciones alternativo

El sistema del caso de estudio se utiliza para desarrollar aplicaciones J2EE que emplean los sistemas JSP y de servlets. El software específico utilizado para esto es el motor de servlets Tomcat, tal y como se comentó con anterioridad en este capítulo. Sin embargo, el mundo de los servidores de aplicaciones Java es muy competitivo (por decir algo) y puede descubrir la necesidad de un servidor diferente.

La tabla 15.2 incluye una lista de diversos servidores populares de código abierto para el desarrollo de aplicaciones J2EE. Por supuesto, también hay disponibles un cierto número de soluciones comercializadas. Sin embargo, estas propuestas comerciales suelen ser extremadamente pesadas y puede ser exagerado emplearlas para el desarrollo.

Tabla 15.2. Servidores de aplicaciones para J2EE.

Servidor	URL	Notas
JBoss	www.jboss.org	Un contenedor de EJB; incorpora Tomcat para soportar JSP y servlets.
Orion	www.orion.org	Un servidor J2EE completo.
J2EE RI	java.sun.com/j2ee	La implementación completa J2EE de referencia; sin coste, pero no es de código abierto.

Instalar estos servidores de aplicaciones es ligeramente diferente en cada caso, pero no hay nada para lo que no esté preparado, en especial tras la lectura de los

capítulos 7 a 13. Cuando instale estos, probablemente debería swguir instalándolos como subdirectorios del directorio principal /opt/java para mantener en el mismo lugar todo su software relacionado con Java.

Uso de otros lenguajes web

Puede que necesite desarrollar aplicaciones web, pero no en el lenguaje de programación Java. Por ejemplo, puede necesitar utilizar los lenguajes Perl o PHP. Si lleva a cabo un poco de investigación, descubrirá que utilizando Apache como núcleo, puede construir aplicaciones web en una gran variedad de lenguajes.

No existe ningún modo de describir un proceso universal para instalar estos lenguajes, ya que son muy distintos. Por ejemplo, los programas Perl emplean la interfaz de pasarela común (*Common Gateway Interface*, CGI) y frecuentemente se ejecutan desde un subproceso separado del servidor Apache principal, o desde dentro del propio proceso Apache. Compare esto con el enfoque que utiliza Tomcat, que es ejecutar un proceso de servidor completamente aislado que se comunica con el servidor web Apache a través de la red mediante un módulo Apache de enlace dinámico. Sin embargo, generalmente todos los lenguajes comenzarán a partir de un módulo de Apache. Este módulo se instalará mediante el programa apxs incluido junto con Apache, al igual que Tomcat. Lo que suceda después dependerá el sistema que vaya a instalar, pero una vez más, no debería encontrar nada realmente inesperado. El desarrollo de aplicaciones en estos lenguajes, por supuesto, ¡es otro tema completamente diferente!

Desarrollo de software tradicional

Puede necesitar una estación de trabajo de desarrollo, pero no para aplicaciones web. Es decir, puede que simplemente esté desarrollando aplicaciones de escritorio tradicionales. En este caso, necesitará algo, pero no todo, del material de este capítulo. También resulta más que probable que necesite instalar software adicional relacionado con sus necesidades. Cada desarrollador tiene sus propias herramientas favoritas y sus entornos preferidos para el desarrollo de software. No hay ningún modo de que este libro pueda siquiera comenzar a tratar todos ellos, pero por fortuna, si necesita algo parecido probablemente ya sabrá lo que necesita. (Por ejemplo, los desarrolladores KDE pueden, o no, preferir utilizar el programa KDevelop para crear aplicaciones KDE.) Lamentablemente, no hay gran cosa que este libro pueda decir al respecto. Aún así, una vez más debería estar bien equipado para hacerse cargo de cualquier eventualidad.

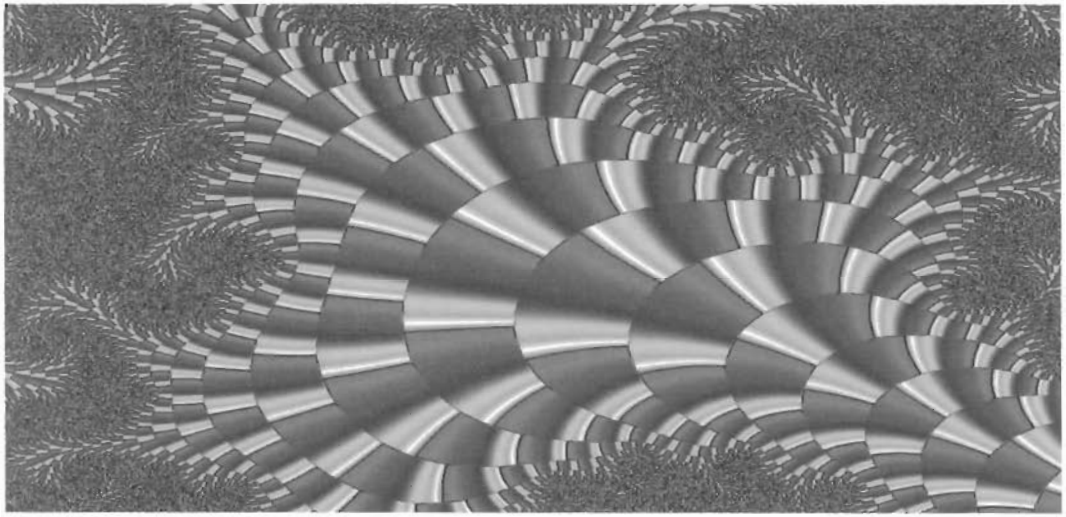
Resumen

Este capítulo comentó una estación de trabajo de desarrollo. El sistema forma parte de una gran red corporativa, y por eso debe configurarse para comportarse

adecuadamente dentro de esa red y de acuerdo con las políticas de los administradores de red. Una vez que el sistema básico de la estación está configurado, puede configurarse el entorno de desarrollo. Este entorno varía de acuerdo con las necesidades, pero para este caso de estudio implicó todo un conjunto de software Java y de herramientas de desarrollo relacionadas.

Si ha leído este capítulo al completo, habrá adquirido dos habilidades distintas. La primera es la facultad de configurar un sistema en una red corporativa. Este proceso no está relacionado de ningún modo con una estación de desarrollo de software y puede aplicarse a cualquier sistema (servidor o estación de trabajo) que necesite conectarse a la red. La segunda área, por supuesto, es la configuración de un entorno Java.

Ahora que ha completado el segundo caso de estudio, debería ser capaz de comparar dos instalaciones diferentes: el sistema de escritorio doméstico del capítulo 14 y la estación de trabajo corporativa de este capítulo. Las diferencias y perspectivas que ha logrado comprender deberían serle de ayuda para personalizar su sistema para cualquier entorno que pueda encontrarse. El siguiente capítulo es un caso de estudio de un cortafuego de red, lo que le proporcionará una tercera perspectiva, muy distinta, sobre la personalización de un sistema Linux.



16
Construcción de un cortafuego de red

Este capítulo presenta una aproximación para construir un cortafuego de red básico pero funcional. Al igual que con todos los casos de estudio mostrados hasta ahora, el objetivo de este capítulo no es conseguir que sea un experto en la construcción de cortafuegos. En su lugar, este capítulo sólo explora superficialmente la seguridad de red; no profundiza en las reglas de los cortafuegos, ya que son sutiles ¡y pueden acabar rápidamente con la funcionalidad de la red! El propósito de este capítulo es mostrar una particularización completa de un sistema Linux.

En otras palabras, el enfoque de este capítulo no es explicar los matices y sutilezas de las reglas de un cortafuego, sino mostrar cómo se ajusta la distribución Linux subyacente para dar soporte a un cortafuego. (Si quiere conocer detalles concretos sobre los cortafuegos basados en Linux, vea el excelente libro de Ziegler "Linux Firewalls", mencionado en la bibliografía al final de este capítulo.) Dicho esto, el contenido de este capítulo debería proporcionar en cualquier caso una base sólida para la construcción de su propio cortafuego básico basado en Red Hat Linux. Tras la lectura de este capítulo, habrá visto un tercer caso de estudio y aprendido sobre diversos programas de software y técnicas de configuración que le ayudarán a preparar un cortafuego o cualquier otro tipo de servidor.

Una cosa importante digna de mención es que en este capítulo se asume un conocimiento bastante amplio sobre las redes TCP/IP. No es de esperar que sea un experto en todas las cosas como la Ethernet, pero al menos debería conocer los conceptos básicos; debería saber qué es un *host* (una máquina), qué es un puerto, cómo funciona la tabla de rutas y demás. Si no tiene conocimientos sobre este material, habrá partes de este capítulo que podrían resultarle inaccesibles. (Si quie-

re una buena introducción sobre los asuntos relacionados con las redes, consulte el Linux Networking HOWTO, citado en la bibliografía al final de este capítulo.)

Comienzo

Esta sección presenta el concepto de un cortafuego, comenta quién podría estar interesado en una configuración de este tipo, y dibuja a grandes rasgos los pasos generales que se siguen para construir uno. Si ya está familiarizado con los cortafuegos en general, puede que sólo quiera hojear esta sección.

¿Qué es un cortafuego?

Puede pensar en un cortafuego (*firewall*) como en una capa de aislamiento entre dos redes. El término "cortafuego" procede de hace varios cientos de años, refiriéndose a las paredes de albañilería resistentes al fuego erigidas entre las estructuras hechas principalmente de madera en las viejas ciudades europeas, como Londres, para impedir que los fuegos locales se extendieran y destruyeran grandes zonas de la ciudad. El término se adoptó más tarde en otros campos, como en la industria del automóvil (donde se refiere a la placa de metal que protege del calor del motor al conductor y a los pasajeros que se encuentran en la cabina) y, por supuesto, en la industria informática.

Así pues, en términos informáticos, un cortafuego se parece a un muro que se levanta entre dos redes y que aísla a ambas de los males que aquejan a la otra. Habitualmente, se utilizan cortafuegos para proteger redes privadas (como las de grandes empresas o su propia red doméstica) del gran desorden salvaje y confuso que es Internet. En general la razón principal para esto es prevenir que atacantes hostiles de la red, procedentes de Internet, comprometan sistemas privados.

Un cortafuego es simplemente un concepto; en la realidad puede tomar diversas formas. Tradicionalmente, un cortafuego es un ordenador real que se sitúa en la red en el punto en que la red interna se conecta con Internet, y todo el tráfico entre las dos redes circula a través del servidor cortafuego. El servidor realiza una gran variedad de tareas (que aprenderá más tarde) para proteger las redes.

Sin embargo, el último par de años ha visto cómo aparecían los denominados cortafuegos personales, que son programas de software que se ejecutan al nivel del sistema operativo en máquinas de escritorio, de oficina y otras configuraciones individuales. Estos programas llevan a cabo los mismos tipos de protección que los servidores cortafuego reales, pero en apoyo de la estación de trabajo individual en lugar de como ayuda a la red completa.

Otro tipo reciente de cortafuego es el encaminador de banda ancha (*broadband router*) que utilizan habitualmente usuarios domésticos y de pequeñas empresas para enlazar múltiples ordenadores a la misma conexión de ancha banda (como DSL o cable); este tipo de encaminadores frecuentemente incluyen características

de cortafuego, de manera que funcionan como el primer tipo de cortafuego mencionado además de cumplir todas sus labores como encaminador y otros trabajos.

En cada uno de estos casos, el concepto de un cortafuego es el mismo: es simplemente una capa de aislamiento.

Prácticas seguras

Los cortafuegos son una parte importante de la seguridad de red moderna. No obstante, si un atacante hostil consigue acceso a la red protegida o a un ordenador que se encuentre tras el cortafuego (¡que puede ser a veces más fácil de lo que podría pensar!), entonces todavía puede comprometer los sistemas que ahora estarían desprotegidos. Por eso, es importante ver los cortafuegos como herramientas, y no tratarlos como una solución de "caja negra" en la que nunca tenga que volver a pensarse una vez que esté instalada.

La mayor parte de los expertos en seguridad sugieren una política de "seguridad en profundidad", lo que significa que nunca debería confiar en una única técnica de seguridad, sino, en su lugar, crear capas mediante el uso de múltiples soluciones. Por ejemplo, no es suficiente que un banco simplemente cierre sus puertas, porque si alguien consigue pasar tras las puertas cerradas, tendrá a su disposición todo el lugar. Los bancos confían en los cierres, sistemas de alarmas, cámaras de vigilancia, guardianes armados, y muchas cosas más.

También debería seguir esta política. Específicamente, no prepare a ciegas un cortafuego como el comentado en este capítulo y suponga que será seguro. Esté siempre atento a las configuraciones de su sistema, y haga caso a la seguridad. Por ejemplo, en los capítulos 4 a 6 se comentaron Red Hat Linux, Slackware Linux y Debian GNU/Linux, y este material incluyó pistas para mantener protegidas estas distribuciones; incluso aunque utilice un cortafuego en su red, siga estas instrucciones. De este modo, aunque su cortafuego se vea comprometido (o si accidentalmente lo configura mal e inhabilita sus protecciones, que es bastante fácil de hacer), sus sistemas no estarán completamente indefensos ante el mundo.

Esto tampoco es sólo un ejercicio académico o teórico. Algunas estadísticas muestran que en algunas circunstancias, el "tiempo de vida" de un sistema Linux no protegido en Internet es de 15 minutos hasta que el sistema se vea comprometido por un atacante.

A ese ritmo, si conecta casualmente un sistema Linux a Internet, la pregunta no es si entrarán en su sistema, sino cuándo. Yo conozco personalmente varias personas cuyos sistemas Linux han sido reventados y explotados, y no es algo agradable cuando sucede. Como poco, le costará tiempo y tensión emocional, y puede costarle algo más si sus datos se pierden o son robados.

Por eso, tómese la seguridad como algo serio. Incluso si dispone de un cortafuego, proteja sus sistemas internos. Cierre todos aquellos servicios inetd innecesarios y deje su cortafuego local activado. Si está ejecutando otros sistemas operativos, descubra cómo proteger también estos sistemas. Puede suponer un poco de trabajo duro y algún tiempo de mantenimiento, pero no le gustará la alternativa.

¿A quién se dirige este ejemplo?

Este capítulo comenta cómo crear un servidor cortafuego y una pasarela a partir de un sistema Red Hat Linux. El objetivo es crear un cortafuego razonablemente seguro y funcional para una pequeña red, en lugar de un sistema superseguro, permanentemente vigilado y de alta disponibilidad. En otras palabras, este capítulo le mostrará cómo construir un cortafuego válido para una pequeña red de oficina doméstica (*Small Office/Home Office*, SOHO), en lugar de un cortafuego completo, que sería apropiado para una red grande y complicada con muchos usuarios.

Con anterioridad en este capítulo, en la sección "¿Qué es un cortafuego?", mencioné el cortafuego "encaminador de banda ancha" que puede adquirir ya preparado. Este capítulo le muestra cómo construir un sistema similar a esas criaturas. Típicamente, estos encaminadores son pequeños sistemas e incluyen una gran variedad de características además del cortafuego, como un conmutador Ethernet incorporado y quizás un punto de acceso inalámbrico (*Wireless Access Point*, WAP) que le permita utilizar la Ethernet inalámbrica 802.11b. Estos sistemas son bastante baratos, puede encontrarlos por alrededor de 150 • en el momento de la escritura de este libro si busca con atención.

Quizás estará preguntándose por qué debería molestarse en leer este capítulo si puede realizar la misma tarea con un equipo ya construido y bastante barato. (De hecho, puede que algunos de estos dispositivos se basen realmente en Linux, ¡así que el parecido es más grande de lo que podría pensar!) Bien, hay dos respuestas a esa pregunta.

La primera es que tendrá un mayor control sobre su cortafuego si lo construye por sí mismo. Puede fijar las reglas que quiera, y configurar el sistema siempre que lo desee, si construye su propio cortafuego. También puede añadir cualquier característica exótica a su cortafuego, como configurar una red privada virtual (*virtual private network*, VPN) hacia otro sitio, añadir un módem para aceptar conexiones de marcado a su red privada mientras se encuentra en movimiento, y demás. Algunas de las unidades cortafuego ya listas soportan algunas de estas características; otras no lo hacen. Aún así, si construye una por sí mismo, garantizará que pueda soportar cualquier característica que necesite, sin importar lo poco convencional que sea. Por ejemplo, si hay un juego de red específico que necesita soportar, esto podría ser más fácil con su cortafuego "casero" que con un sistema "enlatado" sin el mismo nivel de configurabilidad.

La segunda razón es la reutilización de inversiones existentes. Quizás ya dispone de una red Ethernet inalámbrica y tiene todos los conmutadores y el resto de equipo que necesita. Todo lo que necesita es un cortafuego, pero no quiere comprar uno de esos sistemas preparados porque ya dispone de la mayoría de sus componentes. También puede tener un viejo ordenador basado en la tecnología 486 almacenando polvo y le gustaría pasarle un paño y utilizarlo como cortafuego. En un caso como este, este capítulo le resultará de interés.

Por supuesto, la razón principal para leer este capítulo es sólo la curiosidad. Si tiene una actitud del tipo "hazlo tú mismo", puede preferir construir su propio

cortafuego sólo por diversión. O, incluso si se dispone a comprar una de las unidades ya preparadas, puede desear saber cómo funciona realmente (incluso aunque no esté basado en Linux, hará gran parte de las mismas cosas). Si continúa leyendo, aprenderá sobre muchas técnicas que pueden serle útiles en el futuro; por ejemplo, muchas son aplicables para preparar cualquier servidor, no sólo un cortafuego.

Sustitución del cortafuego por defecto

Como se comentó en el capítulo 4, Red Hat Linux incluye un cortafuego básico local instalado en el sistema. Se instala por defecto a menos que se inhabilite explícitamente. (Este cortafuego es en realidad un ejemplo del "cortafuego personal" mencionado anteriormente en la sección "¿Qué es un cortafuego?"; las versiones recientes del sistema Windows de Microsoft disponen de una funcionalidad similar.)

Merece la pena resaltar que el cortafuego por defecto de Red Hat no resulta adecuado para los propósitos de este caso de estudio, que es por lo que más adelante en la sección "Creación de los guiones de inicio" inhabilitará ese cortafuego integrado. El motivo para esto es que el cortafuego de Red Hat sólo es válido para proteger una única máquina, mientras que está interesado en proteger una red (pequeña) al completo. Eso significa que necesita que su cortafuego actúe como encaminador al igual que como cortafuego, de manera que su cortafuego necesita ser más sofisticado que el de Red Hat. Tenga esto presente mientras sigue leyendo. Ahora que tiene una idea de lo que es un cortafuego y de qué tipo de cortafuego va a aprender a construir, puede entrar en materia. En primer lugar, verá una perspectiva general de lo que se necesita hacer; después leerá acerca de cómo hacerlo.

Elementos de un cortafuego Linux

Esta sección detalla los siguientes objetivos primarios de un cortafuego, y la siguiente sección describe cómo personalizar un sistema Red Hat Linux para que cumpla estos objetivos y funcione como cortafuego:

- Minimizar la exposición
- Bloquear a los malos
- Enmascaramiento
- Servicio de red

Minimizar la exposición

La primera prioridad de un cortafuego debe ser protegerse a sí mismo. Después de todo, si su cortafuego se ve comprometido, su red queda expuesta al completo.

Por ello, un buen cortafuego siempre tendrá una exposición mínima al ataque. Estar "expuesto" a un ataque significa que la máquina ejecuta software que tiene una vulnerabilidad que puede ser explotada para comprometer al sistema. Típicamente, un error muy común del software conocido como desbordamiento del buffer (*buffer overflow*) permite a un atacante sobrecargar el programa con una entrada creada cuidadosamente que provoca que el programa ofrezca al atacante una shell de línea de comandos. Si el programa estaba funcionando como usuario *root*, entonces esta shell de comandos se estará ejecutando con privilegios de superusuario, ¡lo que significa que el atacante dispone de todos los permisos sobre el sistema! (Existen otros tipos de ataque, como los ataques de denegación de servicio [*denial-of-service*], para los que hay poca defensa sin la cooperación de su ISP, e incluso errores más simples que permiten que los atacantes consigan un acceso que se supone que no deben tener.) Si un programa con una vulnerabilidad de desbordamiento de buffer es un programa de red, entonces también puede ser vulnerable a una debilidad remota. Si el programa es un programa de servidor como un servidor FTP o un servidor web, entonces el error de desbordamiento de buffer puede ser explotado por un usuario desde un sistema remoto, haciendo que la debilidad sea una debilidad remota. Las debilidades remotas son el santo grial de los reventadores de sistemas, ya que permiten a los atacantes comprometer un sistema sin ni siquiera tener que verlo.

Todos los programas tienen errores, y por eso cualquier programa servidor puede tener una debilidad remota de desbordamiento de buffer. Incluso si no existe una para un programa dado, eso no significa que no exista ninguna, simplemente quiere decir que ninguna es conocida (aún). En otras palabras, cada vez que haya un programa ejecutándose en un sistema, estará exponiendo el sistema a cualquier debilidad remota que pueda existir en ese programa. Es un compromiso entre riesgo y recompensa: se tolera el riesgo de la ejecución de un programa debido a que su utilidad sobrepasa el riesgo.

Por eso, cualquier sistema servidor debería ejecutar tan pocos servicios como sea posible, ya que los servicios que no se usen añaden riesgo pero ningún valor. Esto es especialmente verdad en un sistema cortafuego, ya que el impacto potencial de que el sistema se vea comprometido es muy grande; hace que el riesgo sea menos aceptable. Por ello, un cortafuego debería tener en ejecución la menor cantidad de servicios posible. El cortafuego tal y como se muestra en este libro no ejecuta servicios que sean accesibles para la red externa, sino que sólo ejecuta unos pocos que resultan necesarios para la red interna. Si analiza esta configuración desde el exterior con un programa de estudio de puertos (como el programa nmap que se comentará más adelante en este capítulo), entonces no encontrará nada, y de hecho ¡parecerá que el sistema ni siquiera está activo!

Llevando esta idea a su conclusión lógica, debería desinstalar cualquier software que no utilice. De este modo no ocupa espacio en el disco y garantiza que no se ejecutará. Además, cuanto menor sea la huella dejada por la instalación, mejor, porque herramientas como Tripwire (mencionada posteriormente en este capítulo) resultan más fáciles de gestionar con instalaciones más pequeñas. Para este caso de

estudio, debería indicar al sistema que instale sólo los paquetes que realmente necesita y que ignore los que no; se dan más detalles sobre esto más tarde, en la sección "Reducción de la lista de paquetes".

Bloquear a los malos

Si el primer trabajo del cortafuego es protegerse, entonces su segundo trabajo (¡y objetivo final!) es proteger la red. Esto significa esencialmente impedir que usuarios externos accedan a la red interna, al nivel de paquetes IP (*Internet Protocol*). Es decir, el cortafuego simplemente tira los paquetes no autorizados destinados a la red interna.

Esto se lleva a cabo típicamente mediante la configuración del cortafuego con un conjunto de reglas que se consultan para cada paquete entrante. Cada regla consiste en un patrón que se utiliza para identificar un paquete y una acción que se debe seguir cuando un paquete se corresponde con el patrón. Normalmente, el patrón puede contener cosas como las direcciones IP de origen o destino, el puerto TCP (*Transmission Control Protocol*) o UDP (*User Datagram Protocol*) del paquete, el estado de la conexión TCP/IP, y cosas así; casi cualquier aspecto de las cabeceras TCP/IP puede emplearse en el patrón. Las acciones típicamente incluyen aceptar el paquete, rechazarlo (enviando un paquete de rechazo apropiado de vuelta al remitente del paquete), o ignorarlo completamente. Una configuración de cortafuego es simplemente un conjunto de reglas que implementan la política definida por los administradores de red (en este caso, ¡por usted!).

Claramente, es en estos conjuntos de reglas donde se lleva a cabo el trabajo real de un cortafuego. Si existe un error en el conjunto de reglas, entonces la red podría quedar expuesta. Además, a veces los patrones y acciones puede ser bastante sutiles, haciendo que sea complicado trabajar con ellos. Sin embargo, hay una regla básica general que utiliza la mayoría de los expertos de seguridad: en primer lugar rechace todo, después acepte sólo lo que desee explícitamente.

En realidad, se trata sólo del sentido común: si su casa tiene doscientas puertas pero sólo utiliza dos, entonces realmente quiere cerrar todas las puertas y sólo abrir las que necesita cuando lo necesita. En el caso de su red, los puertos, direcciones IP, y cosas así son las puertas, así que debería crear una regla de cortafuego que rechazase todos los accesos a la red, y después permitir sólo los accesos específicos a los puertos y direcciones IP específicos que necesite. El guión de configuración del listado 16.1 que se ofrece más adelante en la sección "Creación de los guiones de inicio" proporciona más detalles. Esto se aclarará más tarde cuando lea acerca del guión que configura el conjunto de reglas de cortafuego real, comentado en la sección "Presentación de un guión iptables".

Enmascaramiento

Lamentablemente, no se trata del carnaval de Venecia; no hay ningún baile de máscaras de escayola aquí. Enmascaramiento es un término que se usa a veces en la

comunidad Linux para hacer referencia a un caso especial de traducción de direcciones de red (*Network Address Translation*, NAT). NAT se utiliza para lograr que los cortafuegos resulten, bueno, utilizables.

El problema de los cortafuegos es que suelen funcionar en ambas direcciones: impiden el acceso de los usuarios internos a la red externa con la misma efectividad con la que impiden el acceso externo a la red interna. Esto significa que los usuarios internos legítimos tienen muy complicado el acceder a recursos de Internet, como a sitios web, sitios FTP, servidores remotos mediante el protocolo SSH, y cosas así. NAT y el caso especial de enmascaramiento son un modo de restablecer la mayor parte de la facultad de los usuarios para acceder a la red, mientras que se mantienen las características de seguridad proporcionadas por un cortafuego. Debería considerar los cortafuegos y NAT como conceptos separados, incluso aunque en Linux estén implementados por la misma funcionalidad (conocida como el código de filtrado de red [*netfilter*] o iptables).

El enmascaramiento funciona en el cortafuego, y sencillamente toma todos los paquetes de las máquinas internas y los "trocea" (*mangle*, que es realmente el término técnico) para que parezcan haberse originado en el propio cortafuego. Es decir, el cortafuego enmascara al creador del paquete. El paquete se envía después hacia su destino, y cualquier respuesta a ese paquete se "recompone" antes de ser devuelta al ordenador origen del paquete. Desde la perspectiva de la máquina de destino en Internet, los paquetes enmascarados parecen haberse originado en la máquina cortafuego, mientras que en realidad se originaron en una de las máquinas que se encuentran tras el cortafuego.

NAT es más sofisticado que el simple enmascaramiento. En esencia, NAT es un sistema de procesado de paquetes muy completo. El enmascaramiento trocea una parte específica del paquete para un propósito específico: destroza la dirección IP de origen para evitar que se revele la existencia del PC original. Sin embargo, NAT soporta una variedad más amplia de patrones en que se puede basar y de reglas que puede seguir.

Con NAT puede hacer cosas como redirigir conexiones al puerto 80 (el puerto HTTP) hacia otro servidor web interno, incluso aunque el propio cortafuego no ejecute un servidor de este tipo. (Es como lo opuesto al enmascaramiento; esta forma particular de NAT se conoce como NAT de destino [*Destination NAT*, DNAT].) Hacer esto permite que el servidor web permanezca a salvo tras el cortafuego, quedando sólo expuesto el puerto 80. Desde el exterior del cortafuego, el servidor cortafuego parece ser un servidor web, aunque no sea este el caso. La figura 16.1 compara el funcionamiento del enmascaramiento y la NAT inversa o de destino que se acaba de describir.

En la figura 16.1, dos cajas representan las transformaciones que sufren los paquetes de red debido al cortafuego. La caja de la izquierda representa DNAT, en la que las peticiones entrantes a un servidor de la red protegida son recibidas realmente por el cortafuego y se redirigen transparentemente hacia el servidor real. En esta caja, las líneas sólidas representan los paquetes reales que se transmiten desde las máquinas, mientras que las líneas discontinuas representan los paquetes tras su

transformación en el cortafuego. A lo que se reduce esta figura es a que cada máquina (tanto dentro como fuera del cortafuego) "cree" que sólo habla con el cortafuego, en lugar de estar hablando entre sí a través de un intermediario. El cortafuego maneja todas las transformaciones apropiadas para completar esa ilusión sin que las máquinas lo adviertan. Todo el software en ambas máquinas se ejecuta normalmente.

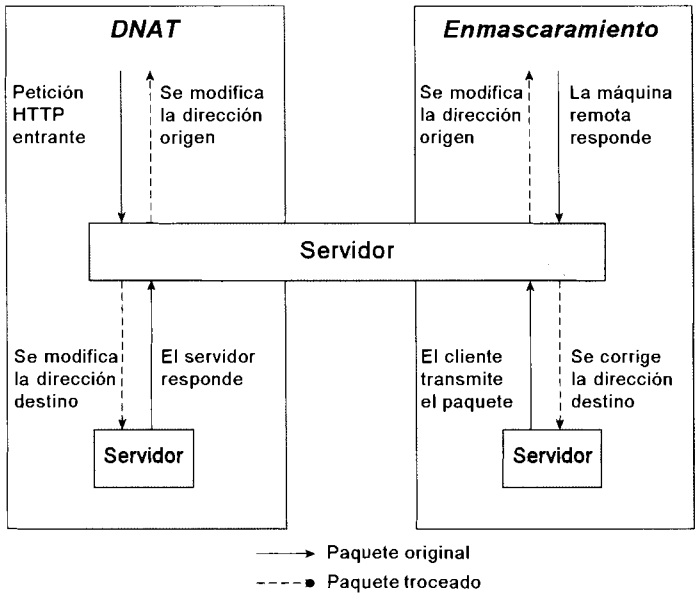


Figura 16.1. DNAT frente a enmascaramiento.

La caja de la derecha representa el enmascaramiento, también conocido aproximadamente como NAT de fuente (*Source NAT*). Esta figura es el análogo de la caja de la izquierda, y representa a un cliente de la red protegida que intenta acceder a un servidor que se encuentra en el exterior, en Internet. En este caso, el cliente cree realmente que se comunica con el servidor real de Internet, y no tiene ni idea de que haya un cortafuego. Mientras tanto, el cortafuego transforma los paquetes originales del cliente (que se muestran como líneas sólidas) para que parezcan provenir del cortafuego (los paquetes transformados se representan como líneas discontinuas). Se vuelven a transformar los paquetes de respuesta del servidor, esta vez para ajustar la dirección de destino para que puedan llegar finalmente al cliente que los solicitó originalmente.

Simplemente recuerde este resumen: en el caso de DNAT, ambas máquinas creen estar hablando directamente con el cortafuego y no saben que el cortafuego actúa simplemente como un intermediario. Sin embargo, con el enmascaramiento, el cliente protegido en el interior cree que habla directamente con el servidor, mientras que el servidor cree hablar con el cortafuego. En ambos casos, el cortafuego se encarga de todas las transformaciones que se necesita que sucedan. Tenga presente este con-

cepto mientras lee el resto de este capítulo, le ayudará volver a él si los diversos conceptos sobre los cortafuegos se confunden en su mente.

El núcleo Linux soporta cortafuegos y NAT. Aun así, cada versión principal del núcleo tiene su propio mecanismo. En la serie 2.0, el núcleo sólo soportaba realmente el enmascaramiento IP y una capacidad limitada de cortafuego. En la serie 2.2, se desarrolló un sistema mejorado conocido como ipchains, y en la serie 2.4 la funcionalidad ipchains se llevó más allá y se llamó iptables (aunque también se conoce como netfilter).

Red Hat Linux 7.3 utiliza los núcleos de la serie 2.4, y por eso este libro describe el uso de iptables en lugar del de versiones anteriores. Sin embargo, Debian GNU/Linux y Slackware Linux utilizan por defecto la antigua serie 2.2 de núcleos; si utiliza alguno de esos sistemas, tendrá que leer acerca de ipchains para poder adaptar el contenido de este capítulo. Afortunadamente, las diferencias no son muy grandes, así que ahora esto no debería resultarle muy difícil. (Si necesita información sobre ipchains, un buen punto de partida sería el documento Linux IPCHAINS-HOWTO que se encuentra en www.tldp.org/HOWTO/IPCHAINS-HOWTO.html.)

Servicio de red

La tarea final de un cortafuego es proporcionar algunos servicios básicos a otras máquinas de la red. En general, esto implica ejecutar un servidor DHCP para que otras máquinas puedan obtener direcciones IP, y un servidor DNS para que otras máquinas puedan realizar búsquedas de nombres de dominio. Más adelante en este mismo capítulo, encontrará comentarios sobre cómo preparar estos servicios en su cortafuego.

Hay quien podría argumentar que no resulta apropiado instalar servicios como DNS y DHCP en el propio ordenador cortafuego, ya que hacer esto embrollaría el cortafuego con procesos innecesarios y también haría el cortafuego vulnerable ante debilidades que existan en el software de los servidores DNS y DHCP. Por supuesto, para muchas redes podría ser más apropiado disponer de servidores DHCP y DNS específicos. No obstante, este capítulo describe una configuración de cortafuego para una red de un tamaño modesto, y por eso probablemente no cambie mucho las cosas ejecutar estos servicios directamente en la máquina del cortafuego.

Ahora que comprende los conceptos básicos de un cortafuego y qué es lo que tiene que realizar uno, las siguientes sección comentará los detalles de cómo construir un cortafuego a partir de un sistema Red Hat Linux 7.3.

Construcción del cortafuego

Esta sección describe el proceso de la construcción de un cortafuego a partir de un sistema Red Hat Linux 7.3 estándar. Dados los requisitos de seguridad y hardware algo más complejos de un cortafuego, tiene que tener algo más de cuidado con su

instalación. Las siguientes secciones comienzan explicando cómo manejar la necesidad de múltiples tarjetas Ethernet en su cortafuego, y después avanzan hacia las tareas de selección de paquetes para instalar y configurar el propio cortafuego real.

Hardware necesario: tarjetas Ethernet

Un cortafuego, si lo piensa, no tiene que hacer gran cosa. Todo lo que tiene que hacer es arrancar, iniciar unos pocos servicios para le red interna, y después preparar y comenzar la ejecución de reglas de cortafuego. A partir de esto, podría conclur acertadamente que no necesita un hardware muy especial para ejecutar un cortafuego. Al menos para sus propósitos, ¡podría descubrir que necesita un poco más de potencia para construir un cortafuego para una gran red corporativa de lo que necesita para el humilde cortafuego para el SOHO de este ejemplo!

El cortafuego descrito en esta sección se implementó sobre un viejo sistema sobrante con un procesador AMD de 166 MHz y 128 MB de memoria. Incluso esta configuración bastante modesta resulta excesiva: seguramente puede trabajar con un viejo procesador de la clase 486 y quizás 16 MB de memoria. Obviamente no es necesaria gran cosa para construir un cortafuego y es probable que necesite incluso menos potencia que esa. ¿Tiene algún sistema "antiguo" que emplee como tope de puerta?

Problemas de las tarjetas Ethernet

Lo único notable, desde el punto de vista del hardware, sobre el propio cortafuego es que necesita tener dos tarjetas Ethernet. Después de todo, está conectando dos redes. Aún así, bajo Linux esto puede resultar un poco complicado de administrar. Cuando el núcleo arranca, detecta las tarjetas Ethernet y les asigna nombres de interfaz de acuerdo con el orden en que las detecta. Sin embargo, es difícil predecir el orden en que el núcleo detectará los dispositivos, especialmente si los dispositivos son gestionados por el mismo controlador (es decir, si son el mismo modelo de tarjeta).

Además, las reglas del cortafuego obviamente necesitan que conozca los nombre de interfaz. (Por ejemplo, ¿cómo si no podría decirle que habilite el enmascaramiento para la interfaz externa si no sabe qué interfaz es la externa?) Por eso, está claro que necesita controlar el orden en que se detectan los dispositivos Ehternet, o al menos saber cuál es cuál.

Nombrado de los dispositivos Ethernet

Hay dos maneras de conseguir que sus dispositivos Ethernet sean nombrados correctamente. El primer modo es no preocuparse realmente en absoluto por renombrarlos. En su lugar, simplemente descubra cuál será el externo y cuál el interno (O eth0 es externo y eth1 es interno, o viceversa). Después simplemente

edite el archivo de parámetros /etc/sysconfig/firewall que se comenta más adelante en la sección "Análisis del guión" para fijar las variables `extiface` e `intiface` correctamente.

Si prefiere obligar a que el dispositivo eth0 sea externo y eth1 sea interno y está utilizando dos tipos diferentes de tarjetas Ethernet, entonces puede que tenga que intercambiarlas. Por fortuna, Red Hat hace que esto resulte bastante sencillo. Red Hat Linux carga los controladores de dispositivo reales para las tarjetas Ethernet en el orden en que se nombran; es decir, el controlador para la tarjeta que se supone que es eth0 se carga en primer lugar, y el controlador para eth1 en segundo lugar. Es posible hacer esto porque los módulos se asocian con los nombres de "eth0" y "eth1" en el archivo /etc/modules.conf. Si sus dispositivos eth0 y eth1 están intercambiados, simplemente edite `/etc/modules.conf`, y cambie la línea "alias eth0 <nombre de controlador de dispositivo>" para que se corresponda con "eth1" (y viceversa para el otro dispositivo). Entonces puede tanto descargar los módulos mediante el comando `rmmod` (no se olvide de cerrar los dispositivos en primer lugar mediante el comando `ifdown`, ya que puede que estén conectados), o simplemente reinicie el sistema.

Si va a utilizar dos tarjetas del mismo modelo, entonces la vida le resultará algo más complicada. Como las tarjetas del mismo modelo serán administradas por el mismo controlador de dispositivo, no puede jugar a este pequeño juego de intercambio de nombres que se acaba de describir. La única alternativa real que tiene es descubrir qué tarjeta es eth0 y cuál es eth1, y después modificar el archivo /etc/sysconfig/firewall tal y como se indica en el listado 16.2 que aparece más tarde en este capítulo. Si se siente ambicioso, según el controlador que esté utilizando puede que sea capaz de pasar parámetros al núcleo mediante el cargador de arranque (normalmente LILO o GRUB, en los sistemas basados en la arquitectura i386). Si está realmente interesado, puede explorar este método, pero probablemente sea más rápido descubrir qué tarjeta es cuál mediante prueba y error.

Para evitar la confusión, el resto de este capítulo asume que el dispositivo Ethernet externo se llama "eth0" y que el dispositivo interno se llama "eth1". Tenga esto presente, y ajústelo si su configuración es diferente.

Nota: Uso del cortafuego con una conexión de marcado

Este capitulo supone que la conexión de su cortafuego con el mundo exterior emplea una red Ethernet. Este es típicamente el caso para las conexiones de banda ancha habituales como DSL y los módems de cable, y para otras conexiones permanentes como las líneas T1 (en Europa, E1) o ISDN (en castellano, RDSI). Sin embargo, si su conexión es un enlace telefónico mediante un módem acústico, necesitará realizar algo de trabajo extra.
En ese caso, su ISP probablemente utilice el protocolo punto a punto (*Point-to-Point Protocol*, PPP) y su interfaz externa posiblemente sea ppp0. Por ello, probablemente su dispositivo interno será eth0. Los guiones que se muestran

en este capítulo pueden hacerse cargo de este caso; sin embargo, tendrá que preparar el enlace PPP por sí mismo. Afortunadamente, esto se describe con detalle en el documento PPP HOWTO que se encuentra en www.linuxdoc.org/HOWTO/PPP-HOWTO/index.html.

Configuración de las interfaces

Una vez que ha conseguido nombrar sus dispositivos Ethernet en el orden adecuado, es hora de configurarlos. En general, utilizará el comando `/usr/sbin/netconfig` de Red Hat para configurarlos. Este es un programa basado en texto que le solicita toda la información necesaria para configurar el dispositivo. Si lo prefiere, también puede editar los archivos /etc/sysconfig/network-scripts/ifcfg-<nombre_de_dispositivo> y editar sus contenidos manualmente. Probablemente sea más fácil usar simplemente `netconfig`; puede pasarle el argumento `-d` para especificar qué interfaz se usará; por ejemplo, `netconfig -d eth1` permitirá configurar el dispositivo eth1.

Necesitará configurar la interfaz externa (que en este ejemplo es eth0) del modo en que su proveedor de servicio de Internet (ISP) le indique; típicamente esto supondrá configurar la dirección IP estática o el protocolo DHCP. Configure el dispositivo externo del mismo modo en que lo haría si estuviera conectando una estación de trabajo independiente a través de la conexión con su ISP.

Necesitará configurar la interfaz interna estáticamente. Su cortafuego será el encaminador para su red interna, por eso siempre tiene que tener la misma dirección IP. Típicamente esto significa que le asignará una dirección IP del tipo 192.168.0.1, una máscara de red 255.255.255.0, y dejará los campos de la pasarela (*gateway*) y el servidor de nombres por defecto en blanco. (El propio cortafuego funcionará como pasarela y servidor de nombres para el resto de sus ordenadores.) Para saber qué dirección IP debería utilizar, consulte una de las próximas secciones titulada "Presentación de un guión iptables".

Para ayudarle a visualizar esto, la figura 16.2 representa la configuración de red que se acaba de describir. También incluye algunos componentes de red adicionales como un conmutador Ethernet y un punto de acceso inalámbrico (WAP) opcional, que se comentará más adelante en la sección "Integración de una LAN inalámbrica".

En la figura 16.2, el servidor de cortafuego de la red se incluye, junto con el conmutador Ethernet y un WAP, como parte de la infraestructura de la red. La infraestructura es colectivamente la tecnología que permite que las cuatro máquinas representadas puedan comunicarse entre sí, y en última instancia con Internet. El cortafuego también se encarga de la tarea adicional de mantener las máquinas protegidas y aisladas de Internet, en la que se tiene poca confianza. De las cuatro máquinas, dos se representan como si utilizaran conexiones Ethernet tradicionales (con hilos), mientras que las otras dos se representan utilizando la tecnología Ethernet

inalámbrica 802.11b, más reciente. El cortafuego no es consciente de esta diferencia, y proporciona servicio a las máquinas con la misma eficacia.

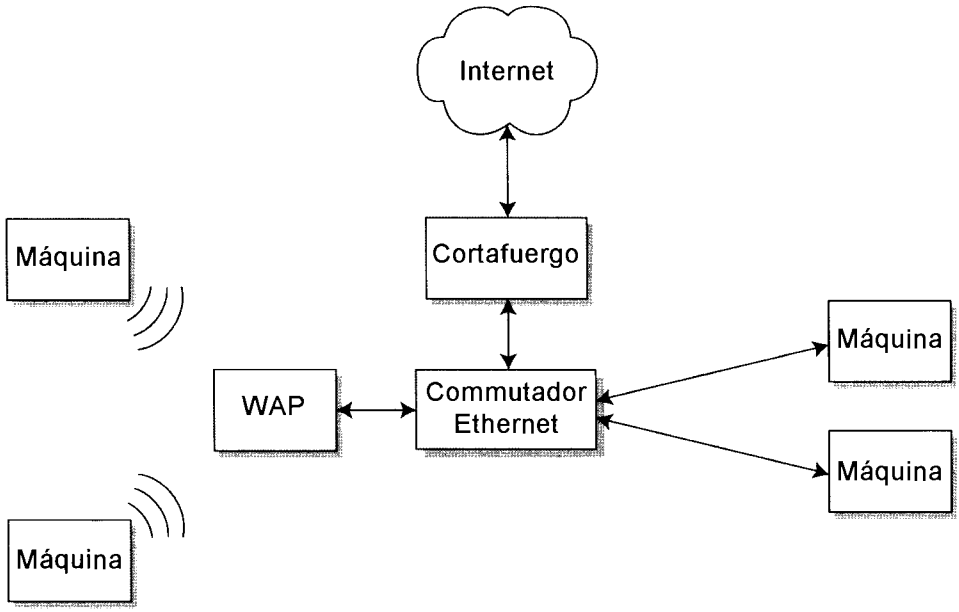


Figura 16.2. Ejemplo de topología de red.

Ahora que comprende los conceptos básicos de la configuración de red que necesita, es hora de considerar cómo instalar realmente el sistema. Las siguientes subsecciones le guían a lo largo de esta tarea; la siguiente sección realiza algunos comentarios sobre las fuentes que puede utilizar como medio de instalación.

Selección de un medio de instalación

Ahora que ya tiene preparado el hardware, necesitará instalar la distribución. No hay nada demasiado sorprendente en esta parte; simplemente tiene que instalar Red Hat Linux como lo haría normalmente. Sin embargo, tiene que prestar algo de atención al soporte a partir del cual realizará la instalación (es decir, el medio de instalación que va a utilizar). Específicamente, puede que no quiera realizar una instalación a partir de la red, por dos motivos.

El primer motivo es que está realizando una instalación desde la red, y es posible que alguien introduzca un "caballo de Troya" como sustitución para los paquetes con un código hostil que abra puertas traseras en su sistema. Se ha informado de muchos casos en los que reventadores de sistemas han comprometido programas de software y los han reemplazado con caballos de Troya; probablemente el caso más destacado sea el del programa tcp_wrappers, ¡que es él mismo un programa

relacionado con la seguridad! Lo mismo podría, en teoría, suceder a un repositorio público de instalación como el sitio de Red Hat. Esto significa que al instalar su sistema a través de la red desde un servidor público FTP o HTTP podría estar sometiéndose a sí mismo a un caballo de Troya. ¿Es esta una posibilidad real, o sólo teórica? Bien, parece improbable que suceda, pero una vez más cualquier reventador de sistemas desearía tener la oportunidad de comprometer un servidor de instalación de este modo.

La segunda (y mucho más probable) situación es simplemente que la instalación requiere un tiempo, lo que significa que su sistema es (o al menos puede ser) vulnerable durante el breve periodo de tiempo que se necesita para completar su configuración. Por ejemplo, en este caso de estudio, hay muchas tareas que tienen que completarse después de la instalación del sistema Red Hat Linux básico. En el tiempo que se necesita para que complete estas tareas, un atacante podría comprometer su nuevo y flamante cortafuego. En estos días de análisis automático de puertos a gran escala y con un "tiempo de vida" de quince minutos, es una posibilidad muy real que alguien pudiera fijarse en la presencia de su nuevo cortafuego en la red y comprometerlo antes de pueda protegerlo completamente.

Estas son posibilidades muy reales, aunque puedan parecer remotas. Por eso es probablemente una buena idea evitar las instalaciones de red tanto como sea posible, e instalar a partir de medios en los que se pueda confiar, como los CD. Si es imperativo que realice la instalación a partir de la red, entonces tenga en cuenta estos problemas. La instalación de cortafuego más segura es la que se lleva a cabo con los cables Ethernet desenchufados.

Reducción de la lista de paquetes

Recuerde que es mejor hacer que la lista de paquetes instalados en su servidor sea tan pequeña como resulte posible. Por ejemplo, no instale ni siquiera un paquete relacionado con el sistema de ventanas X, ya que es una idea realmente mala permitir que los usuarios puedan entrar y trabajar en el ordenador que se utiliza como cortafuego. En general, para un cortafuego no son necesarios los siguientes tipos de aplicaciones y por eso no se necesita realmente instalarlas:

- Software de escritorio de usuario (XFree86, GNOME, KDE, software multimedia, y cosas similares)
- Herramientas de desarrollo (gcc, el depurador de GNU, el lenguaje Perl, y similares)
- Software de servidores (como los servidores web e inetd)
- Programas de usuario generales (clientes FTP, editores de texto, herramientas de impresión, etc...)

Al tener una lista de paquetes reducida, no sólo reducirá el sistema a un estado ajustado y limpio, sólo con lo necesario para funcionar, sino que también ahorrará espacio (lo que le permitirá instalarlo en viejos discos duros, más pequeños) y será

más sencillo gestionar programas como Tripwire. (Tripwire se comentará hacia el final de este capítulo, en la sección "Uso del sistema de detección de intrusiones Tripwire".) La tabla 16.1 es la lista de los paquetes instalados en el sistema cortafuego que se empleó para crear este caso de estudio. Su lista puede variar, según lo que necesite, ¡simplemente asegúrese de tener una razón legítima para cada paquete que se va a instalar!

Tabla 16.1. Un ejemplo de lista de paquetes para un cortafuego.

anacron	ed	kernel	openssh-server	sh-utils
apmd	file	kernel-headers	openssl	slang
ash	filesystem	ksymoops	pam	slocate
at	fileutils	kudzu	parted	statserial
basesystem	findutils	less	passwd	syslogd
bash	ftp	libstdc++	pciutils	SysVinit
bdflush	gawk	libtermcap	popt	tar
bind	gdbm	logrotate	propcps	tcl
bind-utils	glib	losetup	psmisc	telnet
bzip2	glibc	mailcap	pump	termcap
bzip2-libs	glibc-common	MAKEDEV	pwdb	textutils
chkconfig	gmp	man	python	time
console-tools	gnupg	man-pages	python-popt	tmpwatch
cpio	gpm	mingetty	python-xmlrpc	traceroute
cracklib	grep	mkinitrd	rdate	up2date
cracklib-dicts	groff	mktemp	readline	utempter
crontabs	grub	modutils	redhat-logos	util-linux
db1	gzip	mount	redhat-release	wim-common
db2	hdparm	ncompress	rhn_register	vim-minimal
db3	hotplug	ncurses	rootfiles	vixie-cron
db3-utils	info	netconfig	rpm	wget
dev	initscripts	net-tools	rpm-python	which
dhcp	iproute	newt	sed	whois
diffstat	iptables	nmap	setserial	words
diffutils	iptables-ipv6	openssh	setup	zlib
e2fsprogs	iputils	openssh-clients	shadow-utils	

Nota

La lista de la tabla 16.1 contiene sólo los nombres de paquetes RPM sin ninguna referencia a la versión (como "openssh-clients"), ya que las versiones específicas variarán de distribución a distribución, y de versión a versión. Puede comparar esta lista con la suya, pero recuerde que el nombre real del paquete tal y como se instala en su sistema incluirá la versión.

Adición de paquetes de seguridad

Una vez que ha instalado la distribución básica, es hora de comenzar a personalizarla. Quizás lo primero que quiera hacer sea instalar en él unas pocas herramientas relacionadas con la seguridad bastante populares. Esta sección comenta brevemente una de estas herramientas, el creador de mapas de red (*network mapper*) nmap. El programa nmap es una herramienta para el descubrimiento de topologías de red. La mayor parte de los lectores estarán al corriente de las herramientas de diagnóstico de red habituales como ping, traceroute y telnet. (Para aquellos que no estén familiarizados con estas herramientas, merece la pena realizar una lectura rápida de sus páginas de manual, ya que son herramientas verdaderamente muy útiles.) El programa nmap se utiliza para crear un mapa de la topología de red y es un gran añadido a este arsenal.

En esencia, nmap trata de resolver las debilidades de cada uno de esos otros programas. El programa ping, por ejemplo, puede emplearse para comprobar si una máquina se encuentra en funcionamiento y conectada a la red, pero sólo puede comprobar una máquina en cada momento. Mientras tanto, el programa telnet permite comprobar puertos TCP aleatorios para ver si una máquina tiene algún servidor a la escucha en cada uno de ellos; sin embargo, sólo puede comprobar un puerto cada vez. El programa nmap es capaz de analizar rangos de direcciones IP y comprobar la presencia de una amplia variedad de servidores según diversos métodos. Ya que nmap es un analizador de puertos, es una herramienta extremadamente popular entre el grupo de reventadores de sistemas. Esto quiere decir que un atacante podría utilizar perfectamente nmap para analizar su red o su cortafuego en busca de vulnerabilidades potenciales. Por fortuna, nmap también está disponible para que lo pueda utilizar y compruebe su propia red en busca de esas mismas vulnerabilidades.

Red Hat Linux también incluye un paquete para nmap, pero una vez más no se instala por defecto. Una vez que lo tenga instalado, es bastante fácil de utilizar, y su página de manual y ayuda en línea de comandos resultan bastante adecuadas. La documentación electrónica disponible en la red en el sitio www.insecure.org es también una interesante fuente de información. Sin embargo, el comando que como administrador de red encargado de la protección de su red utilizará probablemente con mayor frecuencia es este:

```
$ nmap -sT -p 1-1024 <nombre_de_máquina>
```

Este comando analiza la máquina <nombre_de_máquina> (que puede ser también un rango de direcciones IP, si es que desea comprobar su red al completo de una sola pasada) mediante el método estándar de análisis TCP en busca de servicios abiertos en los puertos 1 a 1024. La mayoría de los servicios de Unix históricamente vulnerables residen en estos puertos por debajo del 1024; probablemente también debería analizar los puertos altos, ya que ocasionalmente los reventadores de sistemas instalarán programas que hagan de puerta trasera y abran puertos en las máquinas comprometidas.

Durante el ensamblaje de su cortafuego, ejecute periódicamente ese comando, y observe los cambios que sufre su salida. Cuando finalice, su sistema debería parecer casi invisible ante nmap (o cualquier otra herramienta) excepto por los puertos que desea que permanezcan abiertos. A lo largo del resto de esta sección, el texto hará referencia a nmap y describirá la salida que debería esperar ver.

Preparación de las cuentas de usuario

Durante la instalación del sistema Red Hat Linux, se le solicitará que cree al menos una cuenta de usuario. Generalmente, es una buena idea emplear la cuenta del usuario *root* sólo para el mantenimiento de un sistema, ya que minimiza el daño potencial que puede producir si comete una equivocación. Sin embargo, en un cortafuego las cosas son un poco distintas.

Anteriormente en este capítulo, dentro de la sección "Minimizar la exposición", leyó acerca de las debilidades remotas. Estas son vulnerabilidades en programas servidores (normalmente debido a un error de programación habitual conocido como desbordamiento de buffer) que permite que los atacantes consigan un acceso no autorizado a un sistema. Sin embargo, los programas que no funcionen como servidores también pueden presentar vulnerabilidades. Este tipo de debilidad también podría permitir a un atacante lograr acceso como *root*, pero sólo si ya dispone de una cuenta como usuario normal.

Por eso, disponer de cuentas de usuario en un sistema cortafuego le expone ante otra clase de vulnerabilidades. Es como si se ofreciera a los reventadores de sistemas un segundo frente de batalla en la guerra contra ellos. Aún más, los usuarios normales ni siquiera encuentran ningún uso para el cortafuego; es una máquina que sólo realiza una tarea. Por estos motivos, es una buena idea no crear ninguna cuenta local de usuario. Si emplea un esquema centralizado de autenticación de usuario como NIS, Kerberos o LDAP (*Lightweight Directory Access Protocol*), no debería conectar su cortafuego con estos sistemas.

Configuración de un nuevo núcleo

Tal y como ya sabe, el núcleo es el corazón del sistema operativo. En un sistema cortafuego, toda la actividad real (es decir, el proceso de las reglas del cortafuego) sucede en el núcleo. Entonces, desde esta perspectiva, el núcleo es el cortafuego. Es tan importante mantener protegido el núcleo como el resto del sistema.

El núcleo Linux tiene la facultad de utilizar módulos de carga dinámica para cosas como los controladores de dispositivos y otras características adicionales. (Por ejemplo, la propia funcionalidad de filtrado de red [*netfilter*] se carga como un módulo del núcleo.) Esto supone una gran conveniencia; sin embargo, también significa que puede modificar el comportamiento del núcleo mientras que el sistema está en funcionamiento.

Debido a esto, hay gente que argumenta que los sistemas seguros y cruciales como los cortafuegos no deberían construirse con soporte para módulos dinámicos. Esto impide que los atacantes puedan cargar módulos hostiles y provocar un daño incluso mayor. Sin embargo, también es cierto que si los atacantes disponen de algún modo de cargar módulos, entonces, por definición, tienen acceso como *root* al sistema, y por eso podrán incluso reemplazar el núcleo si están decididos a hacerlo. Por supuesto, aún así se necesitaría reiniciar el cortafuego, pero al menos es más probable que se de cuenta de que un cortafuego se ha reiniciado (ya que se impedirá temporalmente el acceso a la red externa) que de que hay alguien jugando con el cortafuego. En cualquier caso, es una idea que merece la pena considerar. (Para ver otra alternativa, consulte la sección "Arranque desde un medio de sólo lectura", que se encuentra más adelante, hacia el final de este capítulo.)

Creación de los guiones de inicio

La mejor configuración de cortafuego del mundo resulta inútil si el sistema no está en funcionamiento. Esta sección comentará cómo configurar el sistema para que inicie el cortafuego durante su arranque. Esta sección también entra en detalle sobre las reglas de cortafuego exactas. Probablemente esta sea la sección más importante del capítulo; hasta ahora ha aprendido acerca de los conceptos, pero en este momento conocerá los detalles.

Selección de los servicios de arranque

Tal y como se comentó en el capítulo 4, Red Hat Linux utiliza el modelo SysV para sus guiones de inicio. Red Hat también proporciona unas cuantas herramientas (concretamente chkconfig y service) que facilitan la administración de estos guiones. Utilizará estas herramientas para preparar el sistema. De manera específica, necesita inhabilitar la configuración del cortafuego por defecto (si es que está activado) para reemplazarlo con el sistema particular desarrollado en este capítulo.

Inhabilitación del cortafuego por defecto

La configuración del cortafuego de Red Hat Linux está contenida en el servicio SysV ipchains. Esto significa que el propio archivo del guión real es /etc/init.d/ipchains. (Este guión funciona leyendo las reglas reales del archivo /etc/sysconfig/ipchains, así que si siente curiosidad sobre cómo es el cortafuego por defecto de Red Hat, ese es el archivo que debería leer.) Para poder inhabilitar este guión, tiene que modificar los enlaces simbólicos que se encuentran en los directorios rc.d (por ejemplo, /etc/rc5.d para el nivel de ejecución 5) o emplear la herramienta chkconfig.

Además de inhabilitar el guión ipchains, debería inhabilitar también el guión iptables. (Los guiones iptables e ipchains son mutuamente exclusivos; sólo se puede usar un sistema simultáneamente. Normalmente ipchains se encuentra habilitado e iptables no, a no ser que altere la configuración por defecto, pero podría inhabilitar igualmente ambos guiones.) Los siguientes comandos finalizarán e inhabilitarán permanentemente ambos guiones:

```
$ chkconfig --del ipchains
$ chkconfig --del iptables
$ service ipchains stop
$ service iptables stop
```

Al ejecutar los comandos precedentes, se inhabilitará el cortafuego por defecto de Red Hat en su sistema. Esto puede que le deje expuesto temporalmente a ataques. Si escoge de manera apropiada su lista de paquetes e inhabilita todos los servicios, entonces no debería suponer ningún problema. Sin embargo, puede que aún así desee proseguir la instalación con el sistema desconectado de la red externa hasta que acabe de configurar el guión iptables.

Realizar un análisis

En este momento, merece la pena realizar un rápido análisis de puertos para ver en qué estado se encuentra. Si ejecuta el comando `nmap` que se mostró anteriormente en este capítulo dentro de la sección "Adición de paquetes de seguridad", debería ver que el resultado indica que la máquina se encuentra activa pero no hay ningún puerto abierto. Si nmap informa de que hay algún puerto abierto, probablemente aún existen servicios en funcionamiento. Mediante el comando `netstat -a -p` puede determinar qué procesos se encuentran en ejecución, y después emplear los comandos `service` y `chkconfig` para finalizarlos.

Nota: ipchains frente a iptables

Probablemente se habrá dado cuenta de que incluso aunque Red Hat Linux emplea un núcleo de la serie 2.4, la configuración por defecto del cortafuego se basa en la funcionalidad ipchains de la serie 2.2.
Por esta razón, y debido a que se han descubierto (y corregido) unos cuantos problemas de seguridad, puede que desee considerar el uso del más antiguo y comprobado soporte ipchains (que aún está disponible con la serie de núcleos 2.4). Aún así, para hacer esto, tendrá que transformar el contenido de este capítulo al sistema antiguo. No obstante, el código de filtrado de red debería estar bien, mientras que mantenga su núcleo actualizado.

Presentación de un guión iptables

Ahora que ya ha inhabilitado la configuración por defecto del cortafuego, es hora de instalar el "nuevo y mejorado" cortafuego. Esto es, por supuesto, simple-

mente un guión (como los guiones /etc/init.d/ipchains y /etc/init.d/iptables) que se ejecuta como un servicio SysV. El guión que utilizará, llamado simplemente "firewall", se incluye en el listado 16.1. Para instalar este guión, sólo tiene que copiarlo en el directorio /etc/init.d como /etc/init.d/firewall, y ejecutar a continuación el comando `chkconfig --add firewall`. Las siguientes sección analizan el guión y comentan sus características.

Listado 16.1. Un guión de configuración de cortafuego.

```
#!/bin/sh
#
# firewall               Fija y activa las reglas del cortafuego e IPmasq
#
# chkconfig:    2345    70    25
# description:    Activa/Desactiva las reglas del cortafuego y
enmascaramiento

PATH=/sbin:/bin:/usr/sbin:/usr/bin
. /etc/rc.d/init.d/functions
. /etc/sysconfig/network
[ ${NETWORKING} = "no" ] && exit 0

if [ -x /etc/sysconfig/firewall ]; then
    ./etc/sysconfig/firewall
else
    action "Configuración de cortafuego: /etc/sysconfig/firewall
inexistente" /bin/false
    exit 1
fi

case "$1" in
    start)
        # habilita el filtrado inverso de paquetes (evita el impersonación
IP (IP spoofing)
        # entre interfaces; por ejemplo, impide que alguien del exterior
        # simule encontrarse dentro)
        for iface in $extiface $intiface lo; do
            echo 1 > /proc/sys/net/ipv4/conf/$iface/rp_filter
        done

        # borra la tabla y fija las normas por defecto: tirar (DROP) todo
excepto
        # lo que permitamos explícitamente
        iptables -F
        iptables -t nat -F
        iptables -P INPUT DROP
        iptables -P OUTPUT ACCEPT
        iptables -P FORWARD DROP

        # la máquina local (localhost) es amigable
        iptables -A INPUT -i lo -j ACCEPT
        iptables -A OUTPUT -o lo -j ACCEPT

        # acepta cualquier entrada relacionada con conexiones legítimas
```

```
        iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

        # acepta que salga el tráfico que se origina en esta máquina
        iptables -A OUTPUT -s $extip -j ACCEPT

        # permite específicamente la redirección de la interfaz interna a
la externa
        # y activa el enmascaramiento (también conocido como SNAT con auto-
troceado)
        iptables -A INPUT -i $intiface -j ACCEPT
        iptables -A FORWARD -i $intiface -o $extiface -j ACCEPT
        iptables -A FORWARD -m state -i $extiface -o $intiface \
                        --state ESTABLISHED,RELATED -j ACCEPT
        iptables -t nat -A POSTROUTING -o extiface -j MASQUERADE

        # fija las redirecciones DNAT, para que redirigir las conexiones
ssh, la web, etc..
        # al servidor interno
        for port in $redir_ports; do
            # en primer lugar permite que los paquetes entren en el sistema
            iptables -A FORWARD -i $extiface -o $intiface -p tcp \
                        --dport $port -j ACCEPT
            # … y después los envía a donde pertenezcan
            iptables -t nat -A PREROUTING -p tcp -i $extiface --dport $port
\
                        -j DNAT --to $intserver
        done

        # permite las conexiones ssh a esta máquina SÓLO desde la red
interna
        iptables -A INPUT -i $intiface -p tcp --dport ssh -j ACCEPT

        # permite que el tráfico generado en esta máquina vaya a la red
interna
        iptables -A OUTPUT -o $intiface -j ACCEPT

        # tira los paquetes IP reservados/privados que provengan del
exterior
        # ya los tira la norma por defecto, pero al tirarlos aquí
        # impide que se anoten más abajo, con lo que se reducen las
anotaciones basura
        iptables -A INPUT -i $extiface -s 192.168.0.0/16 -j DROP
        iptables -A INPUT -i $extiface -s 10.0.0.0/8 -j DROP
        iptables -A INPUT -i $extiface -s 172.16.0.0/16 -j DROP

        # anota todo lo que está a punto de tirarse -- por ejemplo, anota
anomalías
        iptables -A INPUT -m limit --limit 5/second -j LOG
        iptables -A OUTPUT -m limit --limit 5/second -j LOG
        iptables -A FORWARD -m limit --limit 5/second -j LOG

        # finalmente, habilita la redirección IP ahora que todo está seguro
        echo "1" > /proc/sys/net/ipv4/ip_forward
        action "Arranque de cortafuego" /bin/true
        ;;
    stop)
```

```
        iptables -F
        iptables -X
        iptables -P INPUT ACCEPT
        iptables -P OUTPUT ACCEPT
        iptables -P FORWARD ACCEPT
        echo "0" > /proc/sys/net/ipv4/ip_forward
        action "Detención de cortafuego [CUIDADO: el sistema está
desprotegido]" \
                        /bin/true
        ;;
    status)
        [ 0 != `cat /proc/sys/net/ipv4/ip_forward` ] && \
                        echo "Redirección habilitada."
        echo "Reglas de cortafuego:"
        echo "**********************"
        iptables -L
        echo
        echo "Reglas NAT:"
        echo "***********"
        iptables -t nat -L
        ;;
    restart)
        $0 stop
        $0 start
        ;;
esac
```

Análisis del guión

El guión firewall es un guión para la shell Bourne (/bin/sh), y las primeras líneas son comentarios de shell. Son particularmente notables estas dos líneas, que permiten que el guión se utilice junto con la herramienta chkconfig de Red Hat:

```
# chkconfig:    2345    70    25
# description:    Activa/Desactiva las reglas del cortafuego y enmascaramiento
```

Estas líneas configuran el guión firewall para que se ejecute en los niveles de ejecución 2 a 5 (que son los niveles de ejecución para los que se habilita la red en Red Hat Linux). El servicio también se ejecuta con una prioridad de 20; esto lo sitúa después de que la red se haya activado, pero antes de que inicien la mayoría de los demás servicios. En otras palabras, el cortafuego se ejecuta tan pronto como es posible durante el proceso de arranque.

Nota

Si desea más información sobre el uso de chkconfig y la creación de guiones compatibles con chkconfig, consulte el capítulo 4.

Las líneas inmediatamente siguientes fijan algunas variables de entorno básicas, y cargan varios guiones comunes que utilizan todos los guiones SysV de Red Hat Linux:

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
. /etc/rc.d/init.d/functions
. /etc/sysconfig/network
[ ${NETWORKING} = "no" ] && exit 0
```

Específicamente, el archivo /etc/rc.d/init.d/functions contiene varias funciones útiles para escribir guiones SysV, y el archivo /etc/sysconfig/network contiene valores para diversos parámetros de red. También hay una comprobación para ver si la red está habilitada; obviamente, no se necesita ningún cortafuego si la red está cortada.

Creación del archivo /etc/sysconfig/firewall

En las siguientes líneas se comprueba la presencia de un archivo /etc/sysconfig/firewall, que contiene valores para diferentes variables importantes para el guión:

```
if [ -x /etc/sysconfig/firewall ]; then
    ./etc/sysconfig/firewall
else
    action "Configuración de cortafuego: /etc/sysconfig/firewall
inexistente" /bin/false
    exit 1
fi
```

Si el archivo no está presente, el guión genera un mensaje de error y finaliza.

Como recordará del capítulo 4, Red Hat Linux utiliza el directorio /etc/sysconfig para almacenar los archivos que contienen los datos necesarios para otros guiones y programas. Por ejemplo, el archivo /etc/sysconfig/mouse contiene parámetros que se utilizan para configurar el ratón del sistema. Se espera que los archivos del directorio /etc/sysconfig sean volátiles, lo que significa que sus contenidos cambian de sistema a sistema. Esto hace que sea más fácil gestionar paquetes RPM, ya que todos los archivos volátiles se encuentran en un único sitio; si desea un comentario más completo, consulte el capítulo 4. El listado 16.2 contiene un archivo /etc/sysconfig/firewall de muestra con valores típicos.

Nota: Enfrentarse a un potencial conflicto de archivos

El guión /etc/sysconfig/firewall mostrado en este capítulo puede entrar en conflicto con otro archivo instalado por un paquete estándar de Red Hat Linux. Concretamente, el paquete RPM "firewall-config" instala un archivo llamado /etc/sysconfig/firewall. Esto significa que si tiene instalado este paquete, surgirá un conflicto. Sin embargo, el paquete firewall-config es realmente un programa del entorno de ventanas X. Ya que aquí está instalando un cortafuego y no tiene instalado X, no hay manera de que tenga instalado firewall-config, y por ello no debería aparecer el conflicto.

Aún así, si va a construir su sistema para que además funcione como máquina de escritorio, puede que surja este conflicto, así que resulta útil estar al tanto de él. (Si necesita tener instalados tanto el paquete como el guión, para evitar

el conflicto puede renombrar el archivo de configuración del cortafuego como algo distinto de /etc/sysconfig/firewall. Sólo recuerde modificar su copia del listado 16.1 para que se corresponda con el nuevo nombre.) En cualquier caso, el programa firewall-config es siempre opcional.

Listado 16.2. Archivo de configuración /etc/sysconfig/firewall de ejemplo.

```
#
extiface=eth0
intiface=eth1
intip=192.168.0.1
extip=`/sbin/ifconfig $extiface 2> /dev/null | sed -n \
        's/^.*inet addr:\([0-9]*\.\)*[0-9]*\) *.*$/\1/p'`
redir_ports="ssh http https"
intserver=192.168.0.3
```

En esencia, el archivo /etc/sysconfig/firewall simplemente tiene que asignar valores a seis variables. Estas variables aparecen en la tabla 16.2 junto con sus significados. La variable extip merece especial atención: la línea que la define en el listado 16.2 es realmente un comando de shell que extrae la dirección IP de la interfaz externa a partir de la respuesta del comando `ifconfig`. Esta técnica resulta útil para los casos en que la dirección IP del propio cortafuego pueda variar, como suele ser típico de las conexiones DSL o de cable populares en muchos entornos SOHO y de las conexiones de marcado que utilizan el protocolo PPP. Si su situación particular le permite emplear una dirección IP estática, debería sustituir la definición de extip por su dirección IP escrita directamente en el archivo /etc/sysconfig/firewall.

Tabla 16.2. Variables de configuración del cortafuego.

Variable	Valores típicos	Significado
extiface	eth0, ppp0	El nombre de la interfaz de la red externa que conecta el cortafuego a Internet.
intiface	eth1	El nombre de la interfaz de red que está conectada con al red interna protegida.
extip	Asignado por el ISP	La dirección IP del cortafuego en Internet.
intip	192.168.x.x	La dirección IP del cortafuego en la red interna protegida; suele ser una de las direcciones IP reservadas no encaminables.
redir_ports	Varía	Una lista de los puertos separados por espacios que se redirigirán transparentemente hacia un servidor interno.

(Continuación)

Variable	Valores típicos	Significado
intserver	Dirección IP	La dirección IP de un servidor de la red interna que se encarga de las peticiones efectuadas en los puertos de la lista redir_ports.

Los valores que podría situar en /etc/sysconfig/firewall variarán de acuerdo con su situación particular. Para empezar, la variable extiface dependerá del modo en que conecte su cortafuego a Internet. Si utiliza un módem de cable o DSL, lo más probable es que este sea un dispositivo Ethernet como "eth0". Sin embargo, si emplea una conexión de marcado, podría estar utilizando PPP, y el dispositivo podría ser "ppp0".

Como valor de la variable de la interfaz interna, intiface probablemente utilizará un dispositivo Ethernet como "eth1". Los dispositivos Ethernet reciben su nombre de acuerdo con el orden en que los detecta el núcleo; el primer dispositivo será "eth0", el segundo "eth1" y así en continuación. Como resultado, un sistema en particular podría arrancar de tal modo que su tarjeta de red interna se detectara realmente en primer lugar, haciendo que fuera eth0, y que la interfaz externa fuera eth1. En este caso, tendrá que asegurarse de que sus variables extiface e intiface se fijan correctamente. (Encontrará una situación similar si utiliza una conexión PPP para conectarse a Internet: su extiface podría ser "ppp0", pero probablemente su intiface será "eth0".)

Nota: Uso del guión firewall con conexiones PPP

Si va a usar su red interna y su cortafuego para compartir una conexión de marcado PPP entre varios ordenadores, entonces probablemente no pueda ejecutar el guión del listado 16.1 como un guión SysV. Esto es así porque las conexiones de marcado sólo están activas cuando el módem se encuentra en funcionamiento, así que hasta que pueda marcar ¡no dispone de una dirección IP externa (o interfaz de red PPP) que el guión pueda utilizar!

En lugar de ejecutarlo como un servicio SysV, tendrá que ejecutar el listado 16.1 después de que haya marcado. Además, tendrá que utilizar el listado 16.1 para finalizar las reglas del cortafuego cuando cuelgue y se rompa la conexión PPP. Si quiere más información sobre cómo realizar esto, consulte la página del manual sobre pppd (el demonio PPP).

Otra alternativa es utilizar la característica de "solicitud de marcado" de pppd, que habilita del dispositivo PPP (por ejemplo, ppp0) pero no marca físicamente. Esto permite que continúe utilizando el listado 16.1 como un servicio SysV. Una vez más, consulte la página del manual si quiere más información.

Una vez que haya identificado correctamente sus interfaces de red, determinar sus direcciones IP es fácil. Para la interfaz externa, su dirección IP probablemente le

sea asignada por su ISP, en cuyo caso se detectará automáticamente en el listado 16.2. Sin embargo, ahora debe escoger una dirección IP para la red interna.

Para la mayor parte de los entornos SOHO, su red interna será probablemente bastante pequeña. Por ello sólo necesitará una única subred Ethernet (es decir, no necesitará un mecanismo extenso de encaminamiento). Para este caso, puede asignar a su red interna la dirección IP en el rango 192.168.0.x. Por ejemplo, su cortafuego podría tener la dirección 192.168.0.1, y otras máquinas direcciones a partir de 192.168.0.2.

Nota: Selección de un rango de direcciones IP

El cuerpo de estándares de la IANA (*Internet Assigned Numbers Authority*) que se encuentra en www.iana.org asigna rangos de direcciones IP a organizaciones que los necesiten. La IANA, sin embargo, reserva tres rangos de direcciones para pruebas, investigaciones y otros propósitos. (Hay tres "clases" de redes IP [A, B y C] con tamaños variables. La de clase A es la más grande y la de clase C la más pequeña.) Estos rangos IP se consideran "no encaminables", que significa que en general son ignorados por los encaminadores y equipos similares. No obstante, ya que están reservados, están libres sin ninguna restricción para quien quiera utilizarlos. Además, ya que el cortafuego que se está construyendo en este capítulo realiza la traducción de direcciones de red (NAT), las direcciones IP de la red interna protegida resultan invisibles para quien observe desde Internet. Por esta razón, estos rangos de red no encaminables resulta útiles (¡y de uso seguro!) para este caso básico de SOHO.

Puede utilizar realmente cualquiera de los rangos de red no encaminables; escogí simplemente el rango 192.168.x.x porque es el rango de clase C, y resultaría muy inusual que un entorno SOHO necesitara más que eso. Sin embargo, puede escoger cualquier rango que desee. Los rangos son 10.x.x.x para direcciones de clase A, 172.16.x.x (o cualquier rango de red hasta 172.31.x.x) para las direcciones de clase B, y 192.168.x.x para las direcciones de clase C.

Determinación de las variables DNAT

Las restantes variables del archivo /etc/sysconfig/firewall son redir_ports e intserver. Estas variables se utilizan para configurar las reglas de DNAT que permiten el acceso público a servidores (como un servidor web) que funcionen tras su cortafuego. Cuando se activan, estas reglas indican al núcleo que redirija cualquier conexión entrante en los puertos indicados hacia otra máquina que se encuentra dentro del cortafuego. La máquina protegida permanece completamente invisible: para un observador externo, parecerá que el propio cortafuego está aceptando las conexiones.

Hay dos variables en /etc/sysconfig/network que activan esta funcionalidad: una es una lista de puertos que se deben redirigir a un servidor interno, y la otra es la dirección IP del servidor interno. La dirección IP es simple: sólo hay que introducir la dirección IP del ordenador que se va a utilizar como servidor como valor de la variable intserver. (Consulte la sección "Instalación de DHCP para máquinas internas" para obtener más información.) Mientras tanto, la lista de los puertos se almacena en la variable redir_ports, y es una lista de puertos (o los nombres simbólicos de los puertos procedentes del archivo /etc/services) separados por espacios en blanco.

Nota

¡Recuerde incluir la lista de puertos entre comillas, ya que tiene que ser una única variable!

Una vez que ha fijado estas variables, el guión configurará automáticamente reglas que preparen las redirecciones DNAT para cada puerto listado hacia la dirección IP especificada. El ejemplo del listado 16.2 configura tres puertos: HTTP, HTTP seguro (HTTPS), y SSH. Lea la siguiente sección para obtener más detalles sobre cómo funciona realmente DNAT y cómo se configura.

Comprensión de las reglas de cortafuego

Ahora que ha visto cómo es el archivo /etc/sysconfig/firewall (gracias al ejemplo del listado 16.2), puede volver al listado 16.1. Ya que el listado 16.1 es un guión de inicio SysV, soporta estos cuatro objetivos o comandos:

- `start`
- `stop`
- `restart`
- `status`

Este no es un libro sobre programación de shell, así que no habrá un comentario exhaustivo de la estructura del guión de shell. Específicamente, no leerá acerca de cómo funciona el "marco de trabajo" del guión, ni sobre cómo se ejecuta cada objetivo individual. Si el listado 16.1 no le resulta claro, debería consultar su referencia favorita sobre la escritura de guiones de shell.

Debería tener presente que este capítulo no hará necesariamente de usted un experto en la construcción de cortafuegos. Es importante recordar que el enfoque adoptado en este capítulo es mostrar un ejemplo sobre cómo tomar una instalación estándar de una distribución Linux y personalizarla de manera extensa para adaptarla a sus propias necesidades, en lugar de profundizar en el mundo de la seguridad informática y el diseño de cortafuegos, que resulta muy complicado. (Si está

interesado en esos temas, una excelente referencia sería el libro Linux Firewalls, Second edition, de Ziegler.) Recuerde esto mientras continúa con la lectura del resto de esta sección. Es de esperar que encuentre en ella suficiente detalle como para comprender lo que sucede, pero el objetivo no es alcanzar el dominio del diseño de cortafuegos.

Arranque del cortafuego

El caso `start` en el guión /etc/init.d/firewall ejecuta el cortafuego mediante la configuración de una gran variedad de reglas de filtrado de red (*netfilter*) gracias al comando `iptables` y a la ejecución de unos cuantos comandos más.

El primer paso en el arranque del cortafuego es habilitar el filtrado de rutas inversas. Esta es una característica proporcionada por el núcleo Linux que tira aquellos paquetes cuya "ruta inversa" no se corresponda con la dirección IP. Por ejemplo, su cortafuego sólo espera "ver" direcciones IP dentro del rango 192.168.x.x procedentes del interior del cortafuego (es decir, procedentes de la interfaz eth1). Si el núcleo ve en algún momento que algún paquete procedente de una dirección 192.168.x.x entra a través del dispositivo eth0, puede suponer razonablemente que el paquete es erróneo o que indica un ataque hostil conocido como impersonación IP (*IP spoof*). El núcleo tendrá que compilarse para incluir soporte al filtrado de rutas inversas.

Estas líneas habilitan el filtrado de rutas inversas tanto en la interfaz de red interna como en la externa (al igual que para el dispositivo de bucle de retorno local, sólo por cuestión de seguridad):

```
for iface in $extiface $intiface lo; do
    echo 1 > /proc/sys/net/ipv4/conf/$iface/rp_filter
done
```

El siguiente bloque de comandos establece las normas por defecto para el cortafuego. Estas son las normas de seguridad; el núcleo las utiliza para decidir los destinos de todos los paquetes que no disparan ninguna de las demás reglas. Un buen diseño de cortafuego indica que siempre debería tirar todo por defecto y aceptar explícitamente sólo los paquetes que quiera permitir.

Las siguientes líneas vacían en primer lugar el núcleo de cualquier regla preexistente, y después fijan las normas por defecto:

```
iptables -F
iptables -t nat -F
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

Fíjese en que la cadena OUTPUT se fija para que por defecto acepte paquetes; es así porque cualquier paquete que se cree en el cortafuego (o que se permita que lo atraviese) casi con toda certeza será seguro. Las otras cadenas (INPUT y FORWARD) se fijan para que todos los paquetes se rechacen por defecto.

Después, el guión comienza a ejecutar reglas que permitan el paso de los paquetes deseados. La primera de estas reglas proporciona esencialmente acceso sin restricciones al dispositivo de bucle de retorno local, llamado "lo". Este es el dispositivo de red "fantasma" que siempre tiene la dirección IP 127.0.0.1. Ya que el dispositivo de bucle de retorno es por definición local a la máquina, puede confiar en él completamente. Estas líneas aceptan los paquetes que circulan por el dispositivo del bucle de retorno:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Las siguientes líneas del guión /etc/init.d/firewall merecen la atención por que no se utilizan, ni siquiera aparecen en el listado en este punto, aunque podrían estar inhabilitadas como comentarios, como lo están aquí:

```
# iptables -A INPUT -i $intiface -j ACCEPT
# iptables -A OUTPUT -o $intiface -j ACCEPT
```

Lo que harían estas líneas si estuvieran presentes y habilitadas sería confiar implícitamente en el dispositivo de la red interna (es decir, eth1) del mismo modo en que se confía en el dispositivo de bucle de retorno. El razonamiento para esto sería que las máquinas que se encuentren en el interior del cortafuego deberían ser amigables, ¿no es así? Bien, quizás a primera vista, pero si piensa en ello, no hay ningún motivo real para que ni siquiera las máquinas internas necesiten un acceso completo al cortafuego. Suponga, por ejemplo, que prepara una red inalámbrica Ethernet y la conecta a su cortafuego. Podría ser posible que un atacante "secuestre" una dirección IP de esta red inalámbrica y ataque a su cortafuego desde el interior. Para evitar este tipo de riesgo, estas reglas ni siquiera están presentes, de manera que el cortafuego queda protegido incluso de los atacantes hostiles que se encuentren en el interior.

Llegados a este punto, el cortafuego está configurado para tirar todo lo que provenga de los dispositivos Ethernet; sólo se permite hacer cualquier cosa al dispositivo de bucle de retorno local. ¡Daría igual que el servidor de cortafuego ni siquiera tuviese tarjetas de red! Los subsiguientes conjuntos de comandos son los que realmente comienzan a establecer el comportamiento del cortafuego.

La siguiente regla indica al cortafuego que acepte cualquier paquete que se sepa asociado con conexiones legítimas solicitadas por un usuario desde el propio sistema del cortafuego o tras él:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Este es un paso crucial: sin esta regla, los ordenadores de su red podrían enviar comunicaciones a cualquier máquina que se encuentre en Internet que ellos deseen (¡pero jamás recibirían una respuesta!). Por supuesto, esta regla sólo se aplica a los paquetes entrantes asociados con conexiones conocidas; cualquier paquete entrante que no haya sido solicitado por un ordenador desde el interior del cortafuego aún seguirá siendo ignorado.

El siguiente conjunto de reglas activa la funcionalidad de enmascaramiento IP, comentada con anterioridad en este capítulo en la sección "Enmascaramiento". El término "enmascaramiento" es específico de la comunidad Linux; el mismo proceso suele denominarse frecuentemente como traducción de direcciones de red (*Network Address Translation*, NAT) en otros sistemas operativos. Realmente, incluso en Linux, el enmascaramiento es sólo un caso especial de lo que se llama NAT de fuente (*Source NAT*, SNAT). En general, NAT se refiere al proceso por el cuál un cortafuego modifica la cabecera de un paquete IP para que parezca tener un origen o destino diferente; SNAT especifica el primer tipo de modificación y DNAT indica el último.

Sin embargo, ¡el enmascaramiento IP obviamente no puede encargarse de todo si los paquetes se tiran antes incluso de que lleguen a la puerta! Por eso, los dos primeros comandos que vienen a continuación permiten que los paquetes procedentes de la interfaz interna entren en el núcleo y les deja ser redirigidos hacia la interfaz externa.

Esto permite que se realice el proceso de enmascaramiento IP con ellos. Fíjese que al activar la redirección precisa dos reglas (una para los paquetes salientes y otra para las respuestas entrantes; la regla de entrada es más restrictiva para que sólo confíe en paquetes relacionados con conexiones conocidas).

Las siguientes líneas del archivo /etc/init.d/firewall habilitan la entrada y redirección para la interfaz interna, y preparan el enmascaramiento IP:

```
iptables -A INPUT -i $intiface -j ACCEPT
iptables -A FORWARD -i $intiface -o $extiface -j ACCEPT
iptables -A FORWARD -m state -i $extiface -o $intiface \
                --state ESTABLISHED,RELATED -j ACCEPT
 iptables -t nat -A POSTROUTING -o extiface -j MASQUERADE
```

En la sección "Determinación de las variables DNAT", leyó acerca de las variables redir_ports e intserver especificadas en /etc/sysconfig/firewall. Las siguientes reglas del listado 16.1 utilizan realmente esas variables. Para cada uno de los puertos especificados mediante la variable redir_ports se crean dos reglas. Una regla permite que los paquetes que provengan de Internet hacia ese puerto a través de la interfaz externa (eth0) entren realmente en el sistema y sean redirigidos, y la segunda hace que sean redirigidos hacia el servidor interno.

Esto le permite configurar un sistema servidor que ejecute un servicio como el servidor HTTP Apache, colocarlo tras el cortafuego donde se encuentra protegido, y aún así permitir que sea accesible desde la red pública Internet. En otras palabras, esto expone los puertos de su servidor protegido a Internet. Las siguientes líneas llevan a cabo esta tarea:

```
for port in $redir_ports; do
    # en primer lugar permite que los paquetes entren en el sistema
    iptables -A FORWARD -i $extiface -o $intiface -p tcp \
                --dport $port -j ACCEPT
    # ... y después los envía a donde pertenezcan
    iptables -t nat -A PREROUTING -p tcp -i $extiface --dport $port \
                -j DNAT --to $intserver
done
```

Nota: Exposición de puertos

Probablemente resulte obvio, pero aún así sigue siendo importante recordar que si determina la redirección de puertos DNAT tal y como se comenta en esta sección, ¡entonces esos puertos de su servidor estarán realmente expuestos a Internet! Por ejemplo, si expone un servidor de correo SMTP como sendmail, entonces su servidor será vulnerable a cualquier riesgo de seguridad que exista en sendmail. Esto aún pone a su sistema servidor ante el riesgo, así que utilice DNAT con precaución, y no exponga puertos a no ser que realmente lo necesite.

Lo más probable es que periódicamente tenga que realizar tareas rutinarias de mantenimiento en su cortafuego. Sin embargo, podría querer que su cortafuego estuviera en algún lugar apartado, como en un armario o una esquina perdida, y por eso puede que no dispusiera de un monitor ni de un teclado conectados a él. Por este motivo, es útil ser capaz de acceder remotamente al cortafuego, mediante el protocolo SSH. (El servidor OpenSSH puede configurarse en el cortafuego tal y como se comenta en el capítulo 8.)

El siguiente comando permite aceptar conexiones SSH con el cortafuego, ¡pero sólo desde la interfaz interna, eth1!

```
iptables -A INPUT -i $intiface -p tcp --dport ssh -j ACCEPT
```

Al restringir OpenSSH para que sólo atienda peticiones en la interfaz interna, se asegura de que sigue sin ser accesible para el mundo exterior, incluso aunque el guión del cortafuego se inhabilite temporalmente.

La última línea relevante en el bloque de arranque del guión /etc/init.d/firewall habilita la redirección IP:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

En esencia, esta línea indica al núcleo que comience a comportarse como un encaminador de red. Hasta la ejecución de este comando, el cortafuego no hará realmente nada útil, ya que el núcleo simplemente ignorará cualquier paquete que no esté destinado a él. Al habilitar la redirección IP se permite que los paquetes sean procesados realmente por las reglas de filtrado de red.

Detención del cortafuego

Desactivar el cortafuego es mucho más fácil que arrancarlo. Todo lo que se necesita hacer es vaciar y limpiar la lista de reglas de filtrado del paquete del núcleo, volver a fijar las normas por defecto, e inhabilitar la redirección IP. Las siguientes líneas hacen exactamente eso, y cierran el cortafuego. (Esto también implicará que el cortafuego queda expuesto, así que no permanezca conectado a Internet si pone fin al servicio de cortafuego.)

```
iptables -F
iptables -X
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
echo "0" > /proc/sys/net/ipv4/ip_forward
```

Una característica interesante de este guión /etc/init.d/firewall es que el comando `start` también comienza con el comando `iptables -F`. Esto significa que en cierto sentido, el comando `start` también cierra implícitamente el cortafuego en primer lugar, antes de volver a crearlo. En otras palabras, no necesita ejecutar `service firewall stop` antes de ejecutar el comando `service firewall start`.

Comprobación del estado del cortafuego

No existe un buen sistema para comprobar si el cortafuego se encuentra en ejecución o no, ya que no hay un programa demonio que se pueda simplemente buscar en la lista de procesos en ejecución. En su lugar, el cortafuego es un conjunto de reglas del núcleo, y el único modo de comprobar su estado es mostrar una lista de todas las reglas actuales. Esto es exactamente lo que hace el comando `status` de /etc/init.d/firewall, mediante la ejecución de los siguientes comandos:

```
[ 0 != `cat /proc/sys/net/ipv4/ip_forward` ] && \
                echo "Redirección habilitada."
echo "Reglas de cortafuego:"
echo "**********************"
iptables -L
echo
echo "Reglas NAT:"
echo "***********"
iptables -t nat -L
```

El guión también informa acerca de si la redirección IP está habilitada o no.

Reinicio del cortafuego

El último comando, `restart`, es el más simple de todos. Lo único que hace, literalmente, es volver a llamar al guión con el comando `stop`, y una vez más con el comando `start`. (Por supuesto, esto resulta redundante, ya que como se mencionó anteriormente, técnicamente no se necesita ejecutar el comando `stop` antes que `start`.) Estas líneas son todo lo que se necesita para implementar `restart`:

```
$0 stop
$0 start
```

Validación de la instalación

Ahora que ya tiene instalado el sistema básico y las reglas de cortafuego están en funcionamiento, es un buen momento para realizar otro análisis rápido de los puer-

tos de su sistema mediante el programa nmap presentado anteriormente en la sección "Adición de paquetes de seguridad". Sin embargo, descubrirá que esto no resulta tan sencillo como antes, ¡porque el resultado del análisis de puertos de su cortafuego será distinto según desde dónde lo realice!

Por ejemplo, si conecta su máquina a Internet y la analiza desde una máquina remota mediante nmap, ni siquiera parecerá estar conectada a la red, excepto por los puertos indicados en la lista de redir_ports. Si la analiza desde el interior del cortafuego, conseguirá resultados diferentes, y si la analiza desde la propia máquina local, ¡una vez más los resultados serán distintos! Esto es de lo que se trata, por supuesto.

Cuando se analiza desde el exterior, el cortafuego debería filtrar todos los puertos excepto los que se indican en la lista redir_ports, que son el 22 para SSH, 80 para HTTP, y 443 para HTTPS, en este ejemplo. (Estos puertos aparecerán como cerrados o abiertos, según si su servidor interno está realmente en funcionamiento en ese momento.)

Cuando se analiza desde la red interna, todos los puertos deberían aparecer como cerrados excepto el puerto 22 para el protocolo SSH. Cuando se analiza desde la propia máquina, podría ver SSH según cómo haya configurado OpenSSH.

Resumen y limitaciones

Esta sección describió el guión /etc/init.d/firewall que prepara las reglas de filtrado de paquetes de red del núcleo. Ahora debería tener una buena comprensión de lo que hace el guión, de cómo cumple sus objetivos como cortafuego. No obstante, esta no es la historia completa, y el guión tiene algunas limitaciones.

Por ejemplo, considere las reglas que preparan DNAT. Estas reglas provocan que se redirija una lista de puertos del cortafuego a un servidor interno. Sin embargo, del modo en que está escrito este guión, todo los puertos se redirigirán al mismo servidor. ¿Qué haría si quiere que las peticiones HTTP se envíen a un servidor interno y las peticiones SSH a otro? No es complicado modificar el listado 16.1 para tener en cuenta esta situación, pero ese soporte no está incluido por defecto.

También es probable que se haya dado cuenta de que no se entró en mucho detalle sobre el modo en que funcionan el marco de trabajo de filtrado de red del núcleo y el comando `iptables`. Lamentablemente, eso se debe a que este libro no trata sobre cortafuegos. Sería fantástico proporcionar más detalles, pero se han escrito libros enteros sobre este tema. Además, incluso una vez que comprenda lo que hacen el filtrado de red y los códigos iptables, la construcción de buenas reglas de cortafuego sigue teniendo algo de arte.

En otras palabras, si desea o necesita información más detallada, tendrá que consultar otras referencias que se centran en estos temas. El libro de Ziegler, Linux Firewalls, Second edition (que se cita en la bibliografía al final de este capítulo), es un excelente punto de partida, al igual que el documento Linux iptables HOWTO (también incluido en la bibliografía). Sin embargo, la mejor manera de dominar los matices de las reglas del cortafuego , es preguntar a los expertos; muchos de ellos

pueden encontrarse a través de la lista de correo Bugtraq (una vez más, incluida en la bibliografía). Este libro se centra en describir cómo convertir un sistema Linux global en un cortafuego, en lugar de en los matices de las reglas de cortafuego. Si sólo desea crear una instalación de este material, debería bastar, pero si prefiere modificarlo, ¡debe asegurarse de comprenderlo meticulosamente en primer lugar!

Soporte a las máquinas internas

Ahora que su cortafuego está en funcionamiento, querrá comenzar a conectar otros ordenadores a su recientemente protegida red interna. Esta sección comentará cómo montar dos aspectos importantes: un servidor de nombres DNS y un servidor DHCP.

Si dispone de una conexión de banda ancha como un módem de cable o DSL, su ISP le proporciona varios servicios para sus ordenadores, quizás de manera más importante DHCP y DNS. El protocolo de configuración dinámica de máquinas (*Dynamic Host Configuration Protocol*, DHCP) se utiliza para asignar direcciones IP a ordenadores que forman parte de una red, y muchos proveedores de acceso de banda ancha utilizan este protocolo para configurar las máquinas de los usuarios. (Sin embargo, hay otras posibilidades, como el protocolo PPP sobre Ethernet [llamado PPPoE] y la simple utilización de direcciones IP asignadas estáticamente.) El servicio de nombres de dominio (*Domain Name Service*, DNS)es, en cambio, el modo estándar de resolver nombres de máquina en un formato legible para los humanos en forma de direcciones IP.

Los ordenadores de su red interna necesitarán acceder a ambos servicios. Sin embargo, como acaba de instalar un cortafuego, ¡ha cortado su acceso a estos servicios! Por eso necesitará configurar su cortafuego para que proporcione estos servicios directamente.

Instalación de DHCP para máquinas internas

El primer paso para conseguir que sus ordenadores internos estén activos para la red es asignarles direcciones IP. Podría asignar simplemente a cada sistema su propia dirección IP, configurarlos estáticamente y ya está, pero a veces puede resultar tedioso; por ejemplo, añadir un nuevo ordenador a la red puede ser aburrido, y podría olvidar qué direcciones IP ya están asignadas y cuáles no. Si va a configurar una máquina que también tiene que poder acceder a otra red (como la red de su oficina), puede ser inconveniente volver a modificar sus parámetros de red cada vez. DHCP hace frente a estos problemas y hace la vida generalmente más sencilla. Casi cualquier cliente que pueda querer conectar a su red será capaz de funcionar como un cliente DHCP, así que todo lo que necesita para que funcione DHCP en su red interna es instalar un programa servidor DHCP en su cortafuego.

Instalación del demonio DHCP del ISC

El Internet Software Consortium (ISC) ofrece una implementación de un servidor DHCP. Este software puede obtenerse a partir de su propio sitio web, que se

encuentra en www.isc.org. Si escoge construir este software a partir del código fuente, descubrirá que es un demonio servidor bastante típico, similar a OpenSSH o a Apache, que se comentan en los capítulos 8 y 11. Sin embargo, Red Hat Linux incluye un paquete RPM para el demonio DHCP del ISC, y existen pocos motivos para no utilizarlo si no tiene otras necesidades específicas.

Para instalar el servidor (que se llama "dhcpd"), simplemente tiene que instalar el RPM desde el CD, o utilizar el programa up2date de Red Hat para conseguir la última versión disponible. El paquete que se incluye en los CD de instalación de Red Hat Linux se llama dhcp-2.0pl5-8.i386.rpm.

El paquete también instalará un guión de inicio SysV para el demonio DHCP, llamado dhcpd. Por eso, para arrancar o detener el servidor DHCP una vez que lo haya instalado, sólo tiene que emplear el comando `service`: utilice `service dhcpd start` y `service dhcpd stop` para arrancarlo o detenerlo, respectivamente.

Nota

El archivo binario del programa servidor DHCP del ISC se llama "dhcpd", pero su paquete en Red Hat se llama "dhcp".

Configuración del demonio DHCP

Tras la instalación del servidor DHCP del ISC, necesitará configurarlo. El archivo de configuración para dhcpd es /etc/hdcpd.conf. Este archivo podría volverse bastante complicado para redes grandes, pero para este ejemplo sigue siendo simple; el listado 16.3 es un ejemplo del código de un archivo /etc/dhcpd.conf. Los contenidos de este archivo se comentan a continuación, pero sólo en lo referente a los detalles de configuración de alto nivel; si quiere información completa sobre la sintaxis de este archivo, debería consultar la página de manual sobre dhcpd.conf.

Listado 16.3. Un archivo de configuración de dhcpd.

```
option domain-name-servers 192.168.0.1;
option routers 192.168.0.1;
#option domain-name "tacals.net";

# host wap.tacals.net {
#     hardware ethernet 00:00:00:00:00:02;
#     fixed-address 192.168.0.2;
#     option host-name "wap";
# }

host server {
    hardware ethernet 00:00:00:00:00:03;
    fixed-address 192.168.0.3;
    option host-name "server";
}
```

```
shared-network INTERNA {
    default-lease-time 3600;
    max-lease-time 3600;
    subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.4 192.168.0.20;
    }
}
```

El formato de /etc/dhcpd.conf es bastante sencillo. En esencia, puede fijar un cierto número de parámetros y opciones, y puede vincularlos a diversos ámbitos. Por ejemplo, puede fijar opciones para máquinas específicas de la red, puede crear grupos de máquinas y fijar opciones para ellas de manera colectiva (mientras que determina opciones distintas para distintas máquinas),o puede fijar opciones globales que afecten a todas las máquinas. (Compare éste con el formato de configuración jerárquica de la configuración del servidor HTTP Apache, tal y como se comenta en el capítulo 11; a pesar de que la sintaxis de /etc/dhcpd.conf es muy distinta, el concepto es parecido.)

En el listado 16.3 están presentes diversos ámbitos. Las tres primeras líneas están en el ámbito global, ya que afectan a todas las máquinas. El siguiente bloque no comentado, encerrado entre llaves ({}), define una máquina y fija las opciones específicas para esa máquina. El último bloque define una subred y fija opciones para el modo en que se asignan direcciones en usufructo DHCP (*DHCP lease*) a las máquinas.

Nota: Comprensión de los usufructos DHCP

Cuando un servidor DHCP asigna una dirección IP, la asigna en usufructo, lo que significa que el cliente sólo puede utilizar esta dirección durante una cantidad de tiempo determinada. Cuando vence ese plazo, se indica que el usufructo ha expirado, y la máquina cliente debe volver a solicitar una dirección IP. Con frecuencia, se asignará al cliente la misma dirección IP que tuvo la última vez, lo que simplemente renueva el usufructo, pero no se garantiza que sea así.

Las opciones globales que se especifican en las dos primeras líneas determinan los valores de varias opciones útiles que el servidor DHCP envía a los clientes junto con su respuesta. Como probablemente sabrá, la simple obtención de una dirección IP no basta para que un ordenador pueda unirse a una red; el ordenador también necesita información de encaminamiento como la dirección IP del ordenador que actúa como encaminador pasarela que lo conecta con el resto de la red y con Internet. También necesita la dirección IP de un servidor DNS, para que pueda realizar búsquedas de direcciones IP de máquinas de la red e Internet.

En este ejemplo, tanto el encaminador como el servidor se fijan para que sean la dirección IP del ordenador cortafuego (192.168.0.1). (La siguiente sección comenta la configuración del cortafuego para que también funcione como un redirector

DNS.) Estas líneas se emplean para fijar información de encaminamiento en los clientes; debería ser capaz de conectar cualquier ordenador cliente (como un sistema de escritorio) configurado para utilizar DHCP, y esta información bastaría para que la máquina estuviera activa en la red.

La tercera línea fija otro parámetro global, que es el nombre de dominio de la red local. Esta línea está comentada (anteponiendo a la línea el carácter "#" del mismo modo que en muchos otros archivos de configuración de Unix), porque es importante no escoger simplemente un nombre de dominio ficticio que se nos ocurra y esperar que funcione.

Si dispone de su propio nombre de dominio y sus servidores DNS están correctamente configurados, entonces puede habilitar esta línea; sin embargo, en la red SOHO típica que es el objetivo de este ejemplo, probablemente no tenga esa configuración. Por eso, debería dejar esta línea comentada. La parte negativa de esto es que debe referirse a sus máquinas internas mediante sus direcciones IP numéricas, lo que puede resultar inconveniente. Es posible configurar el servidor DHCP del ISC para que funcione junto con el programa bind que se comentará en la próxima sección, pero esta es una situación más complicada y no resulta importante para el objetivo del ejemplo de este capítulo. Si está interesado, debería consultar la documentación del ISC que se encuentra en su sitio web, www.isc.org.

Las siguientes líneas forman un bloque de máquina (*host*), y contienen opciones que se fijan sólo para una máquina específica. Este bloque es realmente bastante simple: sólo asigna una dirección IP fija, específica, a un ordenador en particular de la red. Hace esto mediante la línea de la opción "hardware ethernet". Esta línea hace que DHCP compruebe la dirección particular de acceso al medio Ethernet (*Media Access Control*, MAC) para ver si se corresponde con la dirección indicada. Si lo hace, asigna la dirección IP fija determinada e indica al cliente que adopte el nombre de máquina señalado. La clave en todo este proceso es la dirección MAC: ya que no puede haber dos tarjetas Ethernet con la misma dirección MAC, el servidor puede utilizar la dirección para identificar unívocamente una máquina; después simplemente asigna la misma dirección IP a esa máquina cada vez que la solicite.

Desde la perspectiva de la máquina cliente, esta parece ser una transacción DHCP normal, y no tiene ni idea de que utiliza efectivamente una dirección IP estática. Sin embargo, esto resulta útil para las reglas de cortafuego. El guión de configuración de cortafuego comentado anteriormente incluye comandos que configuran las reglas DNAT para redirigir ciertos tipos de tráfico (como el tráfico SSH y web) a alguna otra máquina que se encuentra en la red interna. Sin embargo, la red utiliza DHCP, así que las direcciones IP están sujetas al cambio, lo que obviamente inutiliza las reglas DNAT del cortafuego cuando cambie la dirección IP del servidor. Al introducir la dirección MAC de su ordenador servidor en esta regla de máquina, puede asegurar que siempre reciba la misma dirección IP, de manera que las reglas del cortafuego mantendrán su consistencia con la topología de la red interna. En este ejemplo, se designa que el servidor reciba la dirección IP 192.168.0.3; sin embargo, debería asegurarse de escribir la dirección MAC de su propio servidor en la línea "hardware ethernet".

El listado 16.3 contiene un bloque de máquina similar al comentado. Este bloque hace lo mismo que el que se acaba de describir; sin embargo, utiliza la dirección IP 192.168.0.2 y asigna el nombre de máquina "wap". Esto hace referencia a un punto de acceso inalámbrico (*Wireless Access Point*), que es un dispositivo para conectar ordenadores mediante el estándar inalámbrico Ethernet 802.11b con su red Ethernet principal, la de los cables por el suelo. A lo largo de la red se pueden configurar varios de estos WAP (típicamente mediante el protocolo de administración simple de red [*Simple Network Management Protocol*, SNMP]). Si su red dispone de un WAP de este tipo, puede que desee retirar el comentario de estas líneas y fijar la dirección MAC como sea apropiado, de modo que su WAP, al igual que su ordenador servidor, siempre tenga la misma dirección IP.

El último bloque fija las opciones para una red compartida interna, llamada apropiadamente "INTERNA". Aquí es donde se fijan los parámetros que se debe utilizar para el resto de los ordenadores que se unan a la red. Las líneas "default-lease-time" y "max-lease-time" tienen fijado un valor de 3600 segundos, lo que indica que el usufructo de las direcciones asignadas por el servidor DHCP será siempre de una hora de duración. (Las unidades para estas líneas son los segundos; 3600 segundos es una hora, como ya habrá calculado.) Una vez que se cumple este plazo, los clientes DHCP tendrán que enviar una nueva solicitud al servidor DHCP para renovar el usufructo de sus direcciones. Este es el modo en que el servidor detecta que los clientes no siguen conectados a la red: les solicita que se den de alta periódicamente.

El bloque de red compartida (*shared-network*) también contiene un bloque interior que configura parámetros de red reales. La primera línea de este bloque interior indica que la red en cuestión es la red 192.168.0.0 con la máscara de red (*netmask*) típica de estas redes (255.255.255.0). Las direcciones incluidas por esta red comienzan en 192.168.0.1 y llegan hasta 192.168.0.254. (192.168.0.255 será la dirección de difusión [*broadcast*] a la que responderán todos los ordenadores de la red.) Sin embargo, el bloque de subred (*subnet*) restringe al servidor DHCP de modo que sólo pueda asignar físicamente direcciones IP hasta la 192.168.0.20; además, sólo comenzará a asignar direcciones IP a partir de la 192.168.0.4. (Recuerde que 192.168.0.1 es el cortafuego, 192.168.0.3 es su servidor interno, y 192.168.0.2 podría ser utilizada por un WAP.) No existe ninguna razón real para el límite superior de 192.168.0.20; no añade seguridad porque los clientes hostiles que consigan acceso a su red pueden simplemente comenzar a utilizar una dirección IP estática, sobre la que el servidor DHCP no puede hacer nada.

Una vez que instale el listado 16.3 como /etc/dhcpd.conf, tendrá que arrancar el servidor mediante el comando `service dhcpd start`. Cuando arranque con éxito, otros dispositivos de su red podrán conseguir direcciones IP, así que ya puede arrancar su servidor, conectar su WAP, si es que dispone de uno, y arrancar sus sistemas de escritorio. Estos dispositivos serán ahora capaces de acceder a la red; sin embargo, sólo serán capaces de hacerlo mediante el uso de direcciones IP numéricas, ya que el cortafuego ha cortado todo acceso a los servidores DNS que normalmente utilizarían para buscar nombres de máquinas legibles por los huma-

nos. En la siguiente sección se comenta cómo configurar un redirector DNS para rectificar esta limitación.

Instalación de un servidor de nombres local

Un servidor DNS (también conocido como servidor de nombres [*nameserver*]) es sencillamente un programa de software que se ejecuta como un demonio y convierte nombres de máquinas con significado para los usuarios humanos en direcciones IP numéricas. Ya que las reglas del cortafuego de la sección anterior cortaron el acceso al servidor de nombres de su ISP, tendrá que instalar uno en su propio cortafuego para ofrecer servicios DNS a su red interna.

El programa Unix utilizado habitualmente para hacer esto es el programa bind, que, al igual que el servidor DHCP comentado en la sección anterior, ha sido escrito por el Internet Software Consortium (ISC). (El nombre "bind" son las siglas de "Berkeley Internet Name Domain" [dominio de nombres de Internet de Berkeley].) El programa bind puede funcionar de muchas maneras distintas: puede funcionar como un servidor de dominio completamente dotado para gestionar su propia zona de nombres de dominio, o actuar simplemente como un redirector o repetidor hacia otro servidor. El primer caso es una tarea muy sustanciosa, pero por fortuna el segundo caso resulta mucho más simple y es todo lo que necesita para el cortafuego que está construyendo.

La configuración de bind como repetidor es bastante sencilla: simplemente necesita instalar el software, y después crear su archivo de configuración (que es /etc/named.conf) correctamente.

Instalación del servidor de nombres bind

Red Hat Linux incluye un paquete RPM para bind junto con su distribución. El paquete se llaman bind-9.1.3-4.i386.rpm (para los sistemas con procesadores Intel; consulte la tabla 4.1 del capítulo 4 si utiliza otra arquitectura). Puede elegir instalar este paquete desde el CD, o mediante el programa up2date para conseguir así la última copia procedente de Red Hat.

Igualmente puede escoger descargar el código fuente de bind desde el sitio web del ISC (www.isc.org) e instalarlo por sí mismo. En este caso, la instalación sería bastante parecida a la de OpenSSH, tal y como se comenta en el capítulo 8. Sin embargo, en esta situación el paquete suministrado por Red Hat debería funcionar bien.

Configuración de bind

El servidor de nombres bind necesita ser configurado como repetidor. Un repetidor DNS es un término de bind para referirse a la utilización de un segundo servidor DNS en el que bind delega sus funciones. En este caso, no va a utilizar bind para gestionar su propio dominio, sino para proporcionar servicios DNS a sus ordenadores internos. Para realizar esto, todo lo que necesita realmente es que su instalación local de bind redirija todas las peticiones que se le envían a los servidores DNS oficiales que de otro modo estaría utilizando.

Se necesita fijar tres opciones en /etc/named.conf para configurar bind como repetidor DNS. Una de ellas indica a bind que funcione únicamente como redirector, otra proporciona la lista de las direcciones IP de los otros servidores DNS a los que bind redirigirá las peticiones, y la tercera limita los clientes a los que atenderá bind. El listado 16.4 es el contenido de un archivo /etc/named.conf completo que configura bind como redirector DNS.

Listado 16.4. Un archivo de configuración de bind.

```
options {
        directory "/var/named";
        forward only;
        listen-on { 127.0.0.1; 192,168/16; };
        forwarders { 1.2.3.4; 5.6.7.8; };
}

include "/etc/rndc.key";
```

El listado 16.4 es bastante simple. Consiste en un bloque de opciones (*options*) que fija las tres propiedades importantes, y además contiene otras dos líneas que especifican opciones de instalación. En primer lugar, comentaré las opciones específicas de la instalación para quitárnoslas de en medio.

La primera línea dentro del bloque de opciones especifica un directorio en el que bind deber mirar cuando busque archivos de configuración adicionales. (Sin embargo, este directorio, debería estar realmente vacío ya que va a utilizar bind simplemente como redirector.) Mientras tanto, la última línea del archivo incluye un archivo independiente que contiene una clave secreta que se utiliza para autenticar clientes y otros servidores; sin embargo, una vez más, esta configuración no hará uso de esa clave. Las tres líneas restantes fijan las tres propiedades mencionadas antes. La línea "forward only;" indica a bind que sólo se comporte como repetidor (es decir, que no atienda a peticiones por sí mismo, sino que las reenvíe a otro servidor). La línea "listen-on" le indica que sólo responda a peticiones procedentes de las máquinas de confianza (específicamente, la máquina local 127.0.0.1 y las máquinas de la subred 192.168.x.x.). Si utiliza un rango de direcciones IP diferente para su red, tendrá que introducir aquí la información correcta; consulte la página del manual sobre named.conf si quiere más información.

La última línea del bloque de opciones fija las direcciones IP de los servidores DNS a los que deberían retransmitirse las peticiones. Las direcciones IP que aparecen en el bloque "forwarders" deberían ser direcciones de servidores DNS reales y funcionales. Típicamente, estos serán los servidores DNS ofrecidos por su ISP, aunque técnicamente pueden apuntar a cualquier parte. Si no conoce de antemano las direcciones IP de los servidores DNS de su ISP, tendrá que contactar con él para descubrirlas. No debería, por supuesto, emplear los valores de ejemplo "1.2.3.4." y "5.6.7.8" que se utilizan en el listado 16.4.

Una vez que haya instalado el listado 16.4 como /etc/named.conf, debe arrancar el servicio bind mediante el comando `service named start`. (Si ya está en funcionamiento, puede que necesite utilizar `service named restart`.) Una vez

que arranque con éxito, su redirector DNS estará en marcha, ¡y las máquinas internas dispondrán de acceso a los servicios DNS!

Lo que ya ha conseguido

Si ha leído el capítulo entero hasta aquí y mientras tanto ha ido construyendo su cortafuego, al llegar a este punto habrá realizado las siguientes tareas:

- Conseguir una comprensión rudimentaria sobre los cortafuegos y el modo en que proporcionan seguridad a una red.
- Comenzar con un sistema Red Hat Linux básico y reducirlo hasta el mínimo necesario.
- Configurar un conjunto de reglas de filtrado de red para implementar un cortafuego simple a la par que muy funcional, apropiado para una pequeña red doméstica o de oficina.
- Configurar NAT e instalar servidores DHCP y DNS para proporcionar un acceso completamente transparente a los ordenadores de su red.

¡Esta es una hazaña importante, si lo piensa! Ahora ya dispone de un cortafuego completamente funcional y de una red interna, lo que supone un gran paso en el camino para mantener su privacidad y la integridad de su sistema contra atacantes hostiles procedentes de Internet. Y lo que es mejor, ¡lo ha construido por sí mismo!

Es de esperar que vea cómo el montaje de un cortafuego no es realmente muy distinto de la configuración de cualquier otro ordenador; se trata simplemente de instalar lo que necesita y comprender cómo configurar adecuadamente lo que instala. Sin embargo, el montaje de un cortafuego necesita un cierto dominio sobre el montaje de redes y la creación de un conjunto seguro de reglas de cortafuego, y ciertamente esta breve introducción no puede convertirle en un experto en esas áreas. Sin embargo, ¡debería ser capaz de ajustar y personalizar el propio sistema Linux! Si descubre que le interesan estos otros temas más exóticos y misteriosos, entonces debería consultar la bibliografía que se suministra al final de este capítulo para encontrar excelentes fuentes de documentación.

En fin, ahora ya tiene un cortafuego. ¿Qué va a hacer con él? Bien, por lo menos tendrá que vigilarlo. Ningún cortafuego será jamás seguro al cien por cien, y por eso necesita preocuparse por las debilidades de seguridad que puedan descubrirse en el software que ha instalado, y también necesita vigilar el sistema para que pueda saber lo que le sucede. Estos temas se comentan en la próxima sección.

Vigilancia y mantenimiento del sistema

Incluso tras la instalación de su cortafuego, no puede sencillamente olvidarse de él. Tendrá que realizar un cierto esfuerzo de mantenimiento, y siempre debería

tener un ojo puesto en el sistema para estar atento ante ataques potenciales o incluso compromisos de la seguridad. Esta sección comenta algunas técnicas habituales que podría desear utilizar.

Configuración del sistema de anotaciones del sistema

Quizás lo más importante que puede hacer para mantener protegido y en funcionamiento su sistema de cortafuego (o cualquier otro sistema) es comprobar periódicamente sus archivos de anotaciones. Sin embargo, ya que probablemente no entrará muchas veces en el ordenador cortafuego, sería bueno ser capaz de ver las anotaciones del cortafuego desde otro ordenador que utilice de manera más regular. Esto puede hacer mediante la configuración del cortafuego para que utilice un sistema de anotaciones remoto (*remote syslog*). La idea básica es que configurará un servidor para que acepte mensajes de anotaciones, y después configurará el cortafuego para que envíe sus mensajes de anotaciones al servidor interno en lugar de anotarlas en su disco duro local. Realizar estos cambios resulta fácil en ambos casos. Aún así, para hacer estos cambios tendrá que conocer la dirección IP del servidor interno. En el ejemplo de este libro, la dirección IP del servidor es 192.168.0.3. (En la sección anterior, "Configuración del demonio DHCP", se fijó esta dirección.) Si su servidor se encuentra en una dirección distinta asegúrese de tenerlo en cuenta.

Configuración del servidor de anotaciones interno

Red Hat Linux incluye un paquete RPM llamado "sysklogd" que implementa un sistema de anotaciones. Este paquete incluye una implementación típica de la prestación de anotaciones de sistema (*syslog*) de Unix. Para configurar el servidor interno para que actúe como el anotador de la red y acepte mensajes remotos enviados desde el cortafuego, simplemente tiene que habilitar ese comportamiento mediante un parámetro en tiempo de ejecución que se pasa al demonio de anotaciones del sistema cuando arranca.

Nota

Esta sección asume obviamente que su servidor interno también ejecuta un sistema Red Hat Linux. Si no se trata de este caso, estas instrucciones podrían no tener aplicación, y puede que tenga que configurar syslog de algún otro modo.

Probablemente pueda adivinar qué archivo va a tener que modificar: /etc/sysconfig/syslog, en el servidor al que enviará sus anotaciones el cortafuego. (¿Ya empieza a resultar algo conocido? ¡Estupendo!) En ese archivo, encuentre la siguiente línea:

```
SYSLOGD_OPTIONS="-m 0"
```

Necesita modificar esta línea añadiendo el parámetro `-r` a la variable que contiene los parámetros que se le pasan a syslogd cuando se arranca desde el guión /etc/rc.d/init.d/syslog. (Este parámetro indica al servidor de anotaciones del sistema que acepte conexiones remotas.) La línea debería ser ahora algo parecido a lo siguiente:

```
SYSLOGD_OPTIONS="-m 0 -r"
```

Ahora debería reiniciar syslog en ese servidor mediante el comando `service syslog restart`.

Perforación del cortafuego local

En este punto, el demonio de anotaciones que se encuentra en su servidor de anotaciones aceptará sin problemas conexiones desde el cortafuego. Sin embargo, si dispone de un cortafuego local habilitado en el propio servidor de anotaciones (que es algo que debería hacer), entonces el cortafuego aún impedirá que el servidor de anotaciones llegue a ver realmente los paquetes. Son necesarios dos pasos para corregir esto.

En primer lugar, edite el archivo /etc/sysconfig/ipchains que se encuentra en el servidor de anotaciones, que es el archivo de configuración que contiene las reglas reales (en la vieja sintaxis de ipchains) para el cortafuego local. Necesita añadir esta línea a ese archivo:

```
-A input -s 192.168.0.1 syslog -p udp -i eth0 -j ACCEPT
```

Esta línea añade una regla al cortafuego que permite que los paquetes del sistema de anotaciones entren en el sistema. (Tal y como puede ver, la sintaxis ipchains resulta muy parecida a la de iptables.)

La segunda modificación que necesitará hacer en el servidor de anotaciones es corregir una singularidad del archivo /etc/sysconfig/network-scripts/ifup-post de Red Hat. Entre otras cosas, este archivo crea las reglas de iptables que permiten explícitamente el paso de los paquetes DNS procedentes de los servidores DNS especificados por el servidor DHCP. Esto tiene que hacerse de forma dinámica, en lugar de almacenar estas reglas junto con las demás reglas del cortafuego local en /etc/sysconfig/ipchains, porque los servidores DNS podría potencialmente cambiar cada vez que el servidor DHCP renueva el usufructo. El archivo ifup-post tiene código para comprobar si ya existe una regla para el servidor de nombres, y si existe alguna, entonces no realiza ninguna acción. Lamentablemente, en este caso de estudio, el cortafuegos es también el servidor de nombres, y por eso la regla que se acaba de añadir a /etc/sysconfig/ipchains también hace referencia al servidor de nombres. Esto impediría que ifup-post proporcionara correctamente acceso al DNS. En otras palabras, ¡no tendría soporte para el servidor de nombres! Por fortuna, el arreglo es simple y sólo precisa un sencillo cambio en el archivo /etc/sysconfig/network-scripts/ifup-post. Para realizar el cambio, encuentre esta línea:

```
if ! ipchains -L input -n | grep -q $nameserver ; then
```

Cambie la línea por esta:

```
if ! ipchains -L input -n | grep "\<53\>" | grep -q $nameserver ; then
```

(Lo que hace es añadir el comando `grep "\<53\>"` a la línea.) Una vez que haya realizado estos dos cambios, el cortafuego local que se encuentra en su servidor permitirá el acceso tanto a DNS como al sistema de anotaciones. Red Hat ha sido advertido sobre este problema, y puede (o no) que lo solucione en una versión posterior.

Ahora que ha configurado los archivos relevantes y ha corregido el guión ifup-post, puede reiniciar todo lo necesite ser reiniciado mediante el comando `service`. Concretamente, necesita reiniciar el cortafuego local basado en ipchains, el servicio syslog (de modo que pueda atender las conexiones realizadas desde el servidor cortafuego de la red), y la red. Es decir, ejecute estos comandos:

```
$ service ipchains restart
$ service syslog restart
$ ifdown eth0; ifup eth0
```

Después eso, su servidor interno debería aceptar conexiones syslog procedentes del servidor cortafuego, ¡y registrar sus anotaciones! En la siguiente sección, configurará el servidor cortafuego de la red para que comience a enviar sus mensajes de anotaciones al recién estrenado servidor de anotaciones.

Configuración de syslog en el cortafuego

Una vez que ha determinado que syslog acepte en el servidor interno los mensajes de anotaciones enviados desde el cortafuego de red, tiene que configurar el cortafuego para que envíe realmente estos mensajes al servidor. Puede conseguir esto mediante la edición de /etc/syslog.conf. Este archivo contiene reglas y patrones que el servidor syslog utiliza para determinar dónde se registran determinados mensajes. Estas reglas se almacenan directamente en el archivo /etc/syslog.conf en lugar de en un archivo como /etc/sysconfig/syslog porque en general no son volátiles; no es habitual modificar las reglas de anotaciones por defecto, aunque esto sea lo que va a hacer ahora. Para los propósitos de su cortafuego, en realidad sólo quiere enviar todos los mensajes al servidor interno.

Es muy simple configurar syslog para que envíe sus mensajes a través de la red hasta un servidor de anotaciones autónomo. En primer lugar, comente todas las líneas del archivo /etc/syslog.conf que no estén comentadas previamente, añadiendo un carácter "#" al comienzo de estas líneas. Esto inhabilita todas las reglas que normalmente provocarían que los mensajes de anotaciones se colocaran en archivos como /var/log/messages y /var/log/secure.

Una vez que haya comentado las reglas por defecto, sólo tiene que añadir esta regla; no olvide cambiar la dirección IP del ejemplo por la dirección IP de su servidor si es que este no tiene la dirección 192.168.0.3:

```
*.*@192.168.0.3
```

(Los dos campos están separados por un tabulador, ¡tenga cuidado con este detalle!) Esta regla envía incondicionalmente todos los mensajes de anotaciones del sistema a la máquina 192.168.0.3 a través de la red, lo que significa que syslog no registrará ningún mensaje en el disco. Si prefiere que sólo se envíen algunos mensajes al servidor de anotaciones interno, o si desea que las anotaciones se registren tanto en la red como en el disco, o casos parecidos, puede añadir a /etc/syslog.conf las reglas adicionales que considere apropiadas. El formato no es muy complicado y ya ha visto suficientes ejemplos de distintas configuraciones como para que le suponga un problema añadir estas reglas por sí mismo. ¡Sólo recuerde asegurarse de consultar la página del manual sobre syslog.conf!

Una vez que haya realizado estos cambios, todos los mensajes de anotaciones sobre el cortafuego aparecerán en el archivo de anotaciones de su servidor (o de cualquier máquina a la que haya indicado que tiene que enviar los mensajes el cortafuego). Entonces ya podrá ver estas anotaciones junto con los mensajes de su servidor y no tendrá que entrar en el sistema del cortafuego para comprobar su estado.

Sin embargo, debería tener presente un par de cosas si decide adoptar este enfoque. En primer lugar, si su servidor interno está desactivado, ¡todos los mensajes de anotaciones que genere el cortafuego se perderán! Por esta razón, puede que desee dejar el cortafuego configurado para que siga almacenando sus archivos de anotaciones en su disco duro local. En segundo lugar, todos los mensajes de anotaciones procedentes del cortafuego estarán mezclados junto con los mensajes del servidor, lo que puede resultar un poco lioso; tenga esto presente cuando revise las anotaciones.

Vigilancia del tráfico del cortafuego

Las reglas del cortafuego comentadas con anterioridad en la sección "Presentación de un guión iptables" incluyen reglas que registran los paquetes entrantes en el sistema de anotaciones syslog. Al mismo tiempo, en la sección anterior se acaba de mostrar cómo configurar el sistema de anotaciones del cortafuego para que se transmita estas anotaciones a un segundo servidor, que es donde revisará los archivos de anotaciones. Al poco tiempo de poner un funcionamiento su cortafuego, sin duda comenzará a ver como muchos mensajes del núcleo informan sobre paquetes que se tiran. Es una buena idea saber cómo se deben interpretar esas líneas.

El objetivo de la anotación de todos los paquetes que tira el cortafuego es mantenerle informado de los sucesos que afectan a su cortafuego. Ya que las reglas del cortafuego se han construido para dejar pasar únicamente el tráfico legítimo y nada más, cualquier cosa que se rechace tendrá que ser ilegítimo. Esto suele suceder por uno de estos dos motivos: o un ordenador o red que se encuentra en alguna parte de Internet está mal configurado y le envía paquetes erróneos, o alguien realiza pruebas o ataques contra su sistema desde una máquina hostil.

Lamentablemente, no existe un buen modo de diferenciar los dos casos. Diversos fabricantes producen software que afirman que puede identificar análisis de

puertos (encontrando la aguja en el pajar, si la hubiera) con diversos grados de éxito. Al final, la herramienta más fiable es el juicio humano. Así que, cuando compruebe las anotaciones sobre paquetes tirados, depende de usted decidir si se trata de paquetes inocuos o peligrosos.

Una entrada de anotaciones típica generada en respuesta a un paquete tirado se parece a algo como esto (excepto que todo se encuentra en una misma línea):

```
Mar 25 23:06:04 192.168.0.1 kernel: IN=eth0 OUT=
MAC=xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx
SRC=xxx.xxx.xxx.xxx DST=xxx.xxx.xxx.xxx LEN=60 TOS=0x00 PREC=0x00
TTL=51 ID=31194 DF PROTO=TCP SPT=3780 DPT=500 WINDOW=65535
RES=0x00 SYN URGP=0
```

Esta línea contiene todos los campos relevantes del paquete. Es más probable que se fije en los campos "SRC=", que es la dirección IP de la máquina desde la que se envió el paquete; "DST=", que es el destino del paquete; y "SPT=" y "DPT=", que son los puertos de partida y de destino, respectivamente. La mayor parte de los demás campos indican qué opciones TCP están activadas y otra información de bajo nivel como el tiempo de vida (*time to live*, TTL) y los parámetros de tamaño del paquete. Esta información puede utilizarse para identificar ataques y, por ejemplo, un ataque podría tener habilitadas las opciones TCP/IP denominadas SYN y RST, así que cualquier paquete que se parezca a esa descripción probablemente sea un paquete hostil.

Este ejemplo se trata de un simple análisis de puertos TCP en el puerto 500. Ya que lo normal es que no haya nada que ejecutándose en su puerto 500, probablemente pueda asumir que se trataba de una persona o programa que analizaba su máquina en busca de vulnerabilidades. El hecho de que quedara registrado, sin embargo, significa que algo se tiró, así que el atacante resultó completamente bloqueado. ¡Desde esta perspectiva, parecerá que su ordenador ni siquiera existe para la red!

Nota: Tirar frente a rechazar paquetes

El cortafuego de este capítulo está configurado para tirar todos los paquetes que no se acepten explícitamente. Esto significa que el paquete simplemente se ignora y no se envía una respuesta de ningún tipo. Debido a esto, la máquina que envió el paquete no tiene modo alguno de saber siquiera que existe el sistema de cortafuego (a menos, por supuesto, que envíe un paquete a un puerto y que este paquete sea aceptado). Este es un comportamiento típico para un cortafuego y proporciona la mayor seguridad.

Sin embargo, existe otra alternativa. En lugar de tirar los paquetes, puede "rechazarlos". Los paquetes rechazados no son "silenciosos" como los paquetes tirados; en su lugar se envía realmente una respuesta en forma de paquete del protocolo de mensajes de control de Internet (*Internet Control Message Protocol*, ICMP) a la máquina que envío el primer paquete, mientras que los

paquetes que se tiran simplemente se descartan sin notificarlo al ordenador remoto (es decir, los paquetes tirados son silenciosos). Esto puede (o no) reducir ligeramente la seguridad, pero a veces es un comportamiento necesario para los cortafuegos más sofisticados. Para la aplicación SOHO comentada en este libro, resulta adecuado tirar simplemente los paquetes en lugar de rechazarlos, pero es bueno tener presente esta alternativa.
Los paquetes se rechazan mediante el uso de `-j REJECT` en lugar de `-j DROP` en los comandos `iptables`. Si desea más información, consulte el libro de Ziegler, Linux Firewalls, Second edition, mencionado en la bibliografía que se encuentra al final de este capítulo.

Clasificación de paquetes

Identificar ataques de entre la mezcla de datos que se consigue no es fácil. Como usuario final, puede que ni siquiera le interese; su objetivo al revisar las anotaciones es simplemente prestar atención a las posibles anomalías. La identificación de patrones de ataques hostiles de entre el flujo normal de paquetes es un tema muy amplio, y en algunos casos una cuestión de grandes negocios. Lamentablemente, no hay aquí espacio suficiente como para tratarlo adecuadamente; lo mejor que puede hacer es inspeccionar sus anotaciones durante un tiempo para hacerse a la idea de lo que "parece" su tráfico normal, y después vigilar las desviaciones de ese patrón. Para obtener más información sobre este tema, probablemente lo más aconsejable sea de nuevo el texto de Ziegler, o alguno de los diversos recursos en formato electrónico como los que se mencionan en la bibliografía que se encuentra al final de este capítulo. ¿Cuál es el modo apropiado de responder ante estos casos? Bueno, esta es un pregunta difícil, y una cuestión de preferencias personales. Si le preocupa, puede rastrear el sondeo o ataque e intentar emprender acciones contra su perpetrador a través de su ISP. O puede adoptar simplemente una posición más relajada y preocuparse sólo ante los grandes ataques. Esta es un área muy difusa, y es una decisión que debe tomar por sí mismo. Si desea más información, debería consultar los sitios web y los documentos que se mencionan en la bibliografía que se incluye al final de este capítulo.

Cuidado y mantenimiento de su cortafuego

En muchos sentidos la administración de un cortafuego (o en realidad la de cualquier ordenador, incluso la de su PC de escritorio) se parece mucho al mantenimiento de un automóvil; periódicamente tendrá que cambiar el aceite y reemplazar los neumáticos. Esta sección proporciona una lista de alguna de las cosas que debería hacer periódicamente para mantener su sistema de cortafuego protegido.

La analogía del mantenimiento del automóvil realmente sólo llega hasta ese punto. El motivo por el que necesita mantener un coche es porque a lo largo del

tiempo el uso normal reducirá sus prestaciones (a niveles peligrosos, si no se trata el problema). Sin embargo, su cortafuego nunca se "desgastará" debido a un uso normal; ¡el ordenador simplemente seguirá funcionando como se le ha indicado hasta que se le indique que se detenga o falle su hardware!

El motivo principal por el que necesita mantener un ordenador no es porque se desgaste, sino porque si deja que se las arregle sólo, el mundo le adelantará. Vulnerabilidades que no eran conocidas en el momento de instalar su cortafuego pueden haberse generalizado más tarde. Por ejemplo, se han identificado varias vulnerabilidades en el código de filtrado de red del núcleo Linux desde que se hizo público. Una vez que estas debilidades sean conocidas, se pueden emplear contra su máquina. Por eso, es importante saber tanto como pueda sobre lo que sus enemigos (es decir, los reventadores de sistemas hostiles) saben, y defenderse con ese conocimiento.

En otras palabras, el conocimiento es poder. El resto de esta sección describe algunas fuentes de conocimiento bastante útiles. Debería considerar beneficiarse de estas fuentes para poder emplear prácticas informáticas seguras.

La lista de correo Bugtraq

La lista de correo Bugtraq se dedica a, bueno, a rastrear errores (*track bugs*). Se encarga de la lista el sitio web SecurityFocus, que se encuentra en www.securityfocus.com, y es un foro para que particulares y organizaciones puedan informar de problemas de seguridad y otros problemas importantes de software popular. La lista incluye información sobre la mayoría de los sistemas similares a Unix, al igual que sobre los sistemas operativos de Microsoft. Las listas Bugtraq se siguen con atención por parte de la industria, y debería considerar seriamente subscribirse a las listas relevantes para cada sistema operativo que tenga instalado en su red.

Las listas de correo Bugtraq pueden encontrarse en online.securityfocus.com/cgi-bin/subscribe.pl.

Listas de fabricantes

Otro recurso valioso pueden ser los servicios ofrecidos por el fabricante de su sistema operativo. Por ejemplo, Red Hat, Inc. rastrea los errores que se producen en su distribución y hace públicas actualizaciones vitales para la corrección de estos errores y la mejora de la seguridad. (El servicio up2date de Red Hat facilita la descarga e instalación de estas correcciones de errores. Si está modificando este caso de estudio para utilizarlo junto con un sistema Debian, podría utilizar en su lugar apt-get.)

Cada sistema operativo o distribución proporcionará periódicamente actualizaciones y correcciones de errores. Siempre merece la pena mantener actualizado su sistema a no ser que tenga explícitamente razones para que no sea así. En la tabla 16.3 se proporcionan los URL de las páginas de soporte de los fabricantes para Red Hat Linux, Slackware Linux y Debian GNU/Linux.

Nota

El URL para Slackware Linux es un enlace con la página de las listas de correo de Slackware; tendrá que subscribirse a la lista "slackware-security" para acceder a la información sobre la seguridad.

Tabla 16.3. Páginas de actualización para distribuciones Linux.

Distribución	URL para información de actualizaciones
Red Hat Linux	www.redhat.com/apps/support/errata/index.html
Slackware Linux	www.slackware.com/lists/
Debian GNU/Linux	www.debian.org/security/

Otras fuentes

A medida que visite estos sitios y listas y profundice un poco en la comunidad, descubrirá que hay muchos más sitios similares. La mayoría de los sitios adoptan un cierto enfoque sobre los problemas relacionados con la seguridad, así que definitivamente es una buena idea buscar y descubrir los sitios que más le gustan. En general, todos tienen la misma información, así que no intente subscribirse a todos ellos (en algún momento hasta el último bit de información llega hasta el rincón más perdido).

No toma mucho tiempo estar al tanto de la seguridad de su sistema. Tomarse algunos minutos al día para asegurarse de que no se es vulnerable ante debilidades bien conocidas es la responsabilidad de cada usuario de Internet. Después de todo, si su sistema se ve comprometido, puede utilizarse para atacar al mío, ¡y ese no es el modo en que me gustaría que se me presentaran mis lectores!

Seguridad en profundidad

Su cortafuego no es el escudo Aegis de Atenea, la diosa griega. Lamentablemente, no se es invencible simplemente por tener un cortafuego. Si instala el cortafuego del modo descrito en este capítulo, habrá dado un gran paso para bloquear a los atacantes casuales (los denominados "chavales de los guiones"). Sin embargo, casi ningún sistema está garantizado para mantener fuera a un atacante determinado, y es posible que no esté cubierta una debilidad posterior que haga que el cortafuego sea vulnerable. Por estos motivos, no puede simplemente instalar el cortafuego y esconder la cabeza bajo tierra, esperando que no pase nada. Existe un principio conocido como seguridad en profundidad, o profundidad en capas, que debería seguir.

En pocas palabras, la seguridad en profundidad significa que no confía en una única fuente para su seguridad. Por ejemplo, si confía solamente en un cortafuego y su cortafuego queda comprometido, su red al completo queda al descubierto y sin protección. Para impedir esto, tiene que proteger cada sistema, no sólo el cortafuego.

Por ejemplo, si está ejecutando un sistema Red Hat Linux desde el interior del cortafuego, no inhabilite su propio cortafuego local; no es realmente necesario, y en el improbable caso de que el cortafuego externo se vea comprometido, aún dispondrá de la protección interna incorporada al sistema. Las últimas versiones de Microsoft Windows también incluyen un cortafuego interno básico; no los deshabilite, por el mismo motivo dado anteriormente.

También debería mantener los sistemas internos actualizados con los últimos parches de seguridad. Una vez más, si el cortafuego externo se ve comprometido de algún modo, aún así un atacante no podrá hacer nada a sus sistemas internos si no resultan vulnerables a ningún ataque conocido.

Todo se reduce al viejo dicho: "No pongas todos tus huevos en la misma cesta". Puede ser fácil descartar este pensamiento como uno de los componentes de la paranoia, o como un "riesgo calculado" que no le importe asumir. No obstante, se revientan sistemas todo el tiempo, y cuando le suceda deseará haber sido más cuidadoso. Intente no aprender las cosas por la vía dolorosa.

Una vez que tiene en marcha una seguridad básica y que sigue la información relevante más reciente sobre su sistema, puede comenzar a pensar sobre algunos problemas más avanzados. La siguiente sección describe algunos de estos.

Investigación de técnicas de seguridad avanzada

Ahora dispone de un cortafuego, y de una comprensión básica acerca de cómo gestionarlo y mantenerlo. También tiene montada una red, con algunos ordenadores conectados. Se ha dado cuenta de que hay un incremento en la productividad, y es algo más consciente acerca de la seguridad de sus sistemas.

¿Qué sucede ahora? Esta sección describirá algunas técnicas interesantes, más avanzadas, que en este momento debería ser bastante capaz de implementar, si lo desea. Con un poco de interés e investigación, debería ser capaz de aunar todas estas técnicas.

Uso del sistema de detección de intrusiones Tripwire

Tripwire es un sistema de detección de intrusiones (*Intrusion Detection System*, IDS). Un IDS hace exactamente lo que sugiere su nombre: detecta intrusiones. Esta

es una herramienta útil para disponer de ella en un cortafuego, ya que proporciona avisos cuando un sistema se ve comprometido y la pronta notificación resulta crucial para reducir el daño.

La tarea de la detección de intrusiones es obviamente un problema de difícil resolución y hay muchas maneras de aproximarse a ella. Algunos IDS funcionan en un ordenador autónomo conectado a la red y vigilan (o "husmean" [*sniff*]) el tráfico de red. Se cuidan de determinados patrones en el flujo del tráfico que podrían señalizar un ataque, mediante una variedad de métodos estadísticos y analíticos. Típicamente pueden configurarse para notificar a los administradores de red el momento en que se detectan ataques. Este tipo de sistema probablemente sea más útil en grandes redes.

Otro enfoque de la detección de intrusiones consiste en estar atento a los archivos modificados. Lo que esencialmente se garantiza cuando un sistema se ve comprometido es que el atacante modificará archivos; después de todo, si no modifica ningún archivo, no puede hacer nada con el sistema que acaba de reventar. Software IDS como Tripwire inspecciona sistemas individuales en busca de archivos que se modifiquen, y avisa a los administradores cuando sucede. Ya que un cortafuego es un servidor que existe únicamente para un solo propósito, no se espera que sus archivos cambien mucho; esto significa que es un buen candidato para el uso efectivo de Tripwire. Tripwire (y otros IDS parecidos) no puede protegerle de los ataques, pero podría ayudarle a aprender rápidamente acerca de los compromisos y a minimizar el daño si su sistema fuera reventado de algún modo.

El problema con un IDS simplista es que le notificará cada vez que es modificado un archivo del sistema. Podría sorprenderse ante el número de archivos que cambian en un sistema, incluso en un servidor cortafuego (por ejemplo, archivos de anotaciones en /var/log, el estado de la semilla aleatoria para el generador de números aleatorios en /var/lib/random-seed, el registro de las entradas del usuario [wtmp y utmp], y cosas así). Como resultado, un IDS demasiado simple abrumará al administrador con falsas alarmas.

Sin embargo, Tripwire no es una herramienta simplista, y dispone de características extensas que le permiten estipular qué archivos podrían cambiar, cuáles no, e incluso identificar las maneras específicas en que puede cambiar un archivo (como un cambio en tamaño pero en propiedad, como en el caso de un archivo de anotaciones). Por supuesto, esta flexibilidad tiene un precio: una configuración de Tripwire típica no es trivial, por decir algo. En realidad, Tripwire requiere un conocimiento exhaustivo de una distribución, y hasta cierto punto el conocimiento de los detalles particulares de una instalación dada. En otras palabras, configurar Tripwire desde la nada es una tarea demasiado grande como para cubrirla en este libro.

Por fortuna, Tripwire es un paquete opcional incluido con Red Hat Linux. Puede instalarlo de los modos habituales (ya sea instalándolo desde la nada o mediante la utilidad up2date). Obviamente, Red Hat también incluye una buena configuración por defecto para Tripwire, lo que supone recorrer un gran trecho para ayudarle a montar correctamente este IDS particular.

Si quiere tener instalado un IDS en su cortafuego, entonces la configuración de Tripwire de Red Hat es un buen punto de partida. Para activar Tripwire, debería

instalar el paquete "tripwire" y después consultar la documentación. El propio paquete contiene alguna documentación (use el comando `rpm -qi tripwire` para ver una lista), y también dispone de una documentación completa a través de la red en www.tripwire.org.

Probablemente el único problema con el que se encuentre como resultado del contenido de este capítulo es que necesitará tener instalado el demonio sendmail; si no es así, Tripwire no podrá enviarle notificaciones mediante correo electrónico. Si siguió los pasos de este capítulo, entonces habrá eliminado sendmail y tendrá que volver a añadirlo. Si reinstala el paquete sendmail, ¡no olvide utilizar `chkconfig` para inhabilitar el servidor sendmail!

Tripwire (o cualquier IDS) puede ser un añadido muy valioso a su cortafuego, y una gran ayuda para la seguridad. Aún así, al igual que con todos estos tipos de herramientas, no es infalible ni universal, así que no confíe solamente en él.

Integración de una LAN inalámbrica

Una característica muy popular que mucha gente añade a sus redes es una LAN inalámbrica basada en el estándar 802.11b del IEEE. Típicamente esto implica añadir un punto de acceso inalámbrico (WAP) a una red Ethernet existente. Lo realmente importante en este caso es asegurarse de que el nuevo segmento inalámbrico de la red funciona adecuadamente con el cortafuego.

Cuando antes generó la configuración del servidor DHCP, reservó una dirección IP para un WAP. En la mayoría de los casos, conectar un WAP a su red será tan sencillo como enchufar el WAP a su conmutador Ethernet. El WAP hará normalmente de puente entre la red inalámbrica y la red principal, y los ordenadores conectados a la red inalámbrica se comportarán del mismo modo que los ordenadores conectados a la de cables. Es decir, los sistemas de la red inalámbrica tendrán acceso a los demonios DHCP y DNS internos, y quedarán protegidos por el cortafuego.

No obstante, hay que hacer algunas advertencias sobre este caso. En primer lugar, muchos (si no la mayoría) de los WAP baratos disponibles en el mercado necesitan un sistema Windows para su configuración. Si su sistema es estrictamente Unix o basado en Linux, puede que encuentre algunos problemas a este respecto. Incluso algunos WAP que se publicitan como si tuvieran soporte para el estándar SNMP en realidad no tienen disponible la suficiente información como para que eso resulte posible. Esto puede ser un inconveniente menor o un gran dolor de cabeza, según cuál sea su situación.

Un problema más grande, e inmensamente más peligroso, es que el puente del WAP es un arma de doble filo. Proporciona a las máquinas de la red inalámbrica un acceso transparente a la red interna, pero esto puede suponer una inmensa vulnerabilidad si se configura mal. Específicamente, el estándar 802.11b soporta un esquema de privacidad equivalente a cable (*Wired Equivalent Privacy*, WEP) que incluye el cifrado de las transmisiones entre los nodos y el WAP. Sin embargo, la mayoría de los WAP se suministran con esta prestación inhabilitada; incluso aunque esté habi-

litada, se ha reventado de manera bastante completa, hasta el punto en que realmente hay que considerar que resulta insegura. Además, algunos WAP no son muy seguros más allá del soporte de WEP y pueden ser secuestrados por atacantes.

El problema potencial en este caso es que después de que se haya tomado tanto trabajo en proteger su red privada de la salvaje Internet, añade un WAP, así que vuelve a abrir la red de nuevo ante diversos ataques. Quizás su vecino se dé cuenta de que tiene en funcionamiento una red inalámbrica y decida gorronear su ancho de banda (lo que puede violar las condiciones de servicio con su ISP), o un atacante puede fijarse en su LAN inalámbrica desprotegida y decidir hacer algo de daño real.

Lamentablemente, comentar soluciones para estos problemas queda más allá del alcance de este libro. El objetivo en este caso es simplemente asegurarse de que sabe en lo que se está metiendo. La configuración del cortafuego comentada a lo largo de este capítulo estará disponible para su red inalámbrica, pero eso puede cortar en ambos sentidos. Asegúrese de hacer sus deberes sobre el estándar 802.11b, su esquema de seguridad WEP, y el hardware particular del WAP que adquiera antes de apretar ese botón. Sobre todo, sea consciente de que utilizar una LAN inalámbrica reduce ampliamente la seguridad de su red. Su cortafuego aún le protegerá de los típicos ataques de los "niños de los guiones" procedentes de Internet, pero al añadir una LAN inalámbrica le convertirá en una presa fácil para cualquiera que se sienta verdaderamente motivado para atacar sus sistemas.

Mejora del servidor DNS

Algo de lo que probablemente se dará cuenta rápidamente es de que si dispone de dos máquinas en la red interna, sólo pueden hacerse referencia entre sí mediante las direcciones IP numéricas. Por ejemplo, si quiere acceder al propio cortafuego, tendrá que referirse a él como 192.168.0.1, como en este ejemplo para el acceso SSH: `ssh -l root 192.168.0.1`. Según cómo utilice su red, esto puede que no suponga un problema, pero una vez más, podría desear una solución mejor.

Una alternativa es asignar nombres de máquina comunes en /etc/hosts. (Este archivo funciona como una especie de "miniDNS" para el ordenador local, asignando direcciones IP a nombres.) Lamentablemente, esto significa que tiene que mantener las mismas entradas de /etc/hosts en múltiples máquinas, y no funciona muy bien junto con las direcciones IP dinámicas que emplea DHCP. Resulta deseable una opción mejor. Bien, ya está haciendo funcionar un servidor DNS, así que, ¿por qué no extenderlo? Al añadir algunas entradas a /etc/named.conf, puede indicarle que cree una "zona" interna para su red. El servidor bind del ISC realmente tiene un cierto grado de integración con el servidor DHCP dhcpd del ISC, lo que hace que resulte bastante llevadero actualizar dinámicamente los nombres de máquina cuando cambien las direcciones IP..., al menos una vez que ha conseguido poner el sistema en marcha.

El reto principal es que tendrá que crear un archivo de zona para su red interna, lo que puede resultar un proceso bastante misterioso. Su mejor opción en este caso

es consultar la documentación del ISC sobre bind y dhcpd. Esto podría suponer un gran reto, y puede que no merezca la pena si hay un pequeño número de máquinas, pero si tiene algo más que unos cuantos ordenadores en su red interna, debería considerar esta alternativa.

Arranque desde medios de sólo lectura

Suponga que el cortafuego se ve comprometido, y de algún modo un atacante consigue entrar. El riesgo principal en este caso es que haga algo permanente al sistema que le permita volver a entrar cada vez que lo desee. Sin embargo, ¿qué sucede si el sistema de archivos del cortafuego se montara como de sólo lectura? En esa situación, no sería capaz de escribir en el disco, y sería imposible que afectara al sistema de esa manera.

Esto implica algo más que simplemente montar las particiones del disco duro como de sólo lectura, por supuesto, porque un atacante podría limitarse a volverlas a montar como de lectura/escritura. Lo que realmente necesita es un medio de sólo lectura (algo en lo que físicamente no se pueda escribir, como en un CD).

Normalmente esto no es posible en una estación de trabajo típica o incluso en un servidor, ya que hay muchos archivos que tienen que cambiar frecuentemente, como los archivos de anotaciones, los archivos de colas para las impresoras y otros archivos similares. Sin embargo, este sistema es un cortafuego, y no hay gran cosa que cambie en absoluto. Lo único dinámico que genera el cortafuego son los mensajes de anotaciones, y si ha implementado el anotador remoto descrito con anterioridad, ni siquiera eso se escribirá en el disco local.

Por eso, resulta posible arrancar un cortafuego desde un medio de sólo lectura como un CD y ejecutarlo desde allí. Esto requiere una cantidad significativa de trabajo, por supuesto, y tendrá que atar muchos cabos sueltos para que funcione correctamente. Hay disponibles herramientas que le ayudarán a hacer esto, como el trabajo sobre un Linux de sólo lectura (Read-Only Linux) que se encuentra en www.ultimeth.net/linux/, mantenido por UltiMeth Systems.

Arrancar su cortafuego desde un medio de sólo lectura como un CD o un disco protegido físicamente frente a escritura podría suponer mucho trabajo, pero por ahora debería disponer de los conocimientos básicos de lo que necesita hacer. Si no por otro motivo, puede tratar esta situación simplemente como un ejercicio académico: ¿Cómo actuaría para hacer esto?

Configuración de una VPN mediante SSH

Si la red que ha montado necesita estar conectada ocasionalmente a otra (como al conectarse con la red de una empresa desde un domicilio), es bastante probable que esté familiarizado con el concepto de una red privada virtual (*Virtual Private Network*, VPN). Una VPN consiste simplemente en dos redes en ubicaciones separadas, que de otro modo estarían desconectadas, que se conectan mediante algún tipo

de conexión cifrada. El método más popular es el protocolo IP seguro (*IP Secure*, IPSec) desarrollado para el proyecto IP versión 6 de Internet de nueva generación. Sin embargo, la mayoría de los dispositivos y el software que soportan IPSec tiende a ser caro, de prestaciones limitadas o difíciles de configurar.

No obstante, si desea arremangarse y hacer las cosas por sí mismo, puede formar una VPN realmente ingeniosa mediante el uso de PPP sobre una conexión SSH cifrada. Usar PPP le permite crear interfaces de red reales en su sistema, así que puede realizar cualquier encaminamiento que necesite hacer para conectar sus dos ordenadores o redes de manera transparente.

En pocas palabras: necesita disponer de sistemas Linux en ambos extremos del conducto VPN, y necesita acceso como *root* a ambos. La máquina de destino necesita tener en funcionamiento un servidor SSH. Entonces, sólo tiene que conectarse a esa máquina mediante el protocolo SSH (como por ejemplo al utilizar OpenSSH), y ejecutar posteriormente el demonio PPP llamado pppd en el extremo remoto. Localmente, también ejecuta pppd, y conecta ambos demonios. ¡Ya está! Una VPN instantánea. La figura 16.3 representa esta situación.

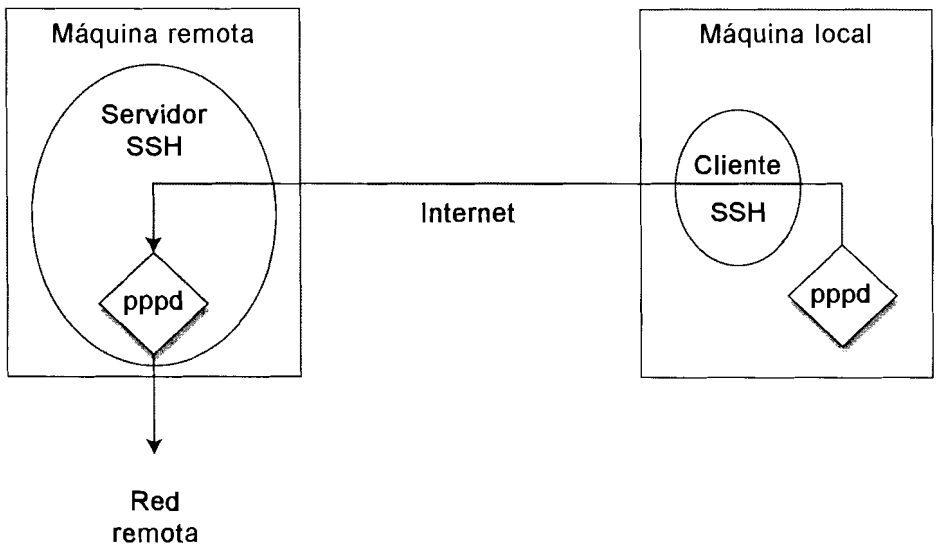


Figura 16.3. Una VPN que emplea PPP sobre SSH.

Por supuesto, no es así de simple, pero casi. El auténtico reto es conectar los dos procesos pppd, que están funcionando sobre sistemas distintos; hacer que se comuniquen adecuadamente puede ser un poco complicado. Lo que tiene que conseguir de algún modo es asignar sus entradas estándar y salidas estándar entre sí. Si busca en Internet, descubrirá una gran variedad de programas en Perl o C que hacen esto

en su lugar. Sin embargo, incluso con estas herramientas, aún tendrá que establecer físicamente las conexiones entre los servidores, lo que significa que también tiene que automatizar varias cosas.

Entrar en detalle en este asunto realmente significaría perder el sentido de este libro. Sin embargo, una solución completa para una VPN que utilice PPP sobre SSH automatizada por un guión de espera está disponible en el CD-ROM que se incluye junto con el libro. Si está interesado, consúltelo.

Resumen

En este capítulo ha aprendido cómo construir un cortafuego. Comenzó con una instalación Red Hat Linux estándar y personalizó la lista de paquetes para excluir todo lo que no resultase estrictamente necesario. Después instaló un guión para que estableciera un conjunto de reglas de cortafuego que preparan realmente las protecciones del sistema.

Finalmente, instaló y configuró otros componentes de software que proporcionan servicios importantes a sus máquinas internas protegidas.

Deténgase un momento y piense sobre eso. Comenzó con una instalación estándar y la ajustó y personalizó para que cumpliera un conjunto de necesidades muy específicas.

Fue capaz de hacer esto porque en la segunda parte aprendió acerca de las diversas distribuciones Linux, y en la tercera parte aprendió cómo instalar software en su sistema. Por supuesto, no puede decir que sea un experto en seguridad tras la lectura de este capítulo, pero es que no se trataba realmente de eso.

Esperemos que ahora sienta haber conseguido un dominio respetable sobre los sistemas Linux (y no sólo sobre una distribución). Este caso de estudio se basó en Red Hat Linux, pero es probable queahora los engranajes de su cabeza estén girando, considerando cómo realizaría esta misma tarea en un sistema Debian o Slackware. Sí, es muy posible hacer lo mismo en ambos. Es incluso posible en otras distribuciones, y lo que es más importante, es posible para usted.

Una despedida

Bien, ¡ya está! Ha terminado: ya no hay más que leer. Puede que yo me sienta algo sentimental, pero, si he conseguido mi objetivo, ¡usted no! Con suerte le habrá echado un vistazo al mundo de Linux. Aunque, en realidad, debería haberle echado algo más que un vistazo; debería haber seguido una introducción guiada y recibido una educación básica.

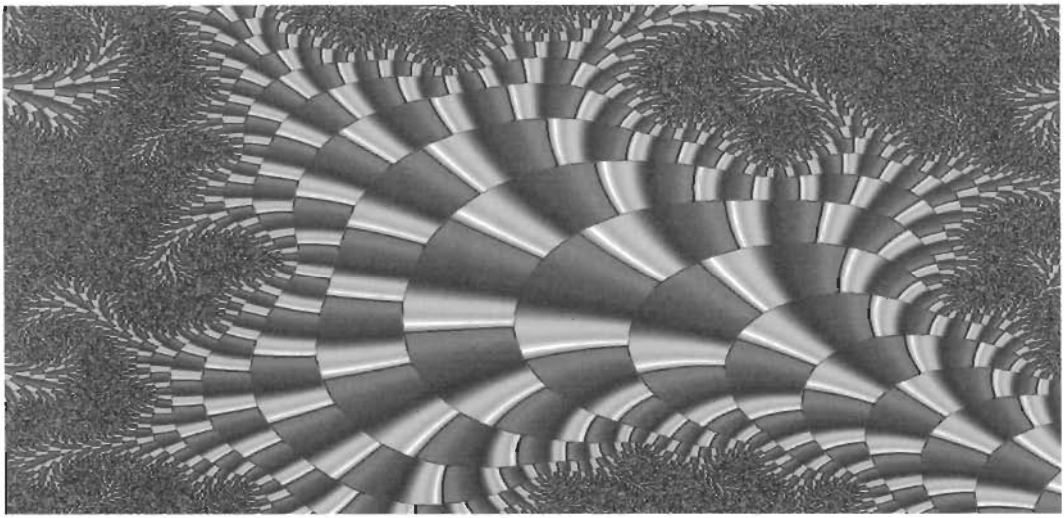
Si he cumplido mi objetivo, debería sentirse entusiasmado sobre las posibilidades que le esperan y la libertad que ahora tiene para hacer las cosas a su modo. ¡Diviértase, y disfrute de su ordenador y su maestría!

Bibliografía

- Drake, Joshua. Linux Networking HOWTO. CommandPrompt, Inc., 2000. www.linuxdoc.org/HOWTO/Net-HOWTO/index.html.
- Russell, Rusty. Linux 2.4 NAT HOWTO. netfilter.samba.org/documentation/HOWTO/NAT-HOWTO.html.
- Russell, Rusty. Linux 2.4 Packet Filtering HOWTO. netfilter.samba.org/documentation/HOWTO/packet-filtering-HOWTO.html.
- Russell, Rusty. Linux iptables HOWTO. www.linuxgurus.org/iptables/howto/iptables-HOWTO.html.
- SecurityFocus. Archivo de la lista de correo Bugtraq. online.securityfocus.com/archive/1.
- Ziegler, Robert L. Linux Firewalls, Second edition. New Riders Publishing, 2002. ISBM 0735710996.

Parte V

Apéndices



Apéndice A
Reuniéndolo todo

¡Lo ha conseguido! Ha llegado al final de este libro. Si ha leído realmente (o al menos ha revisado) esta criatura completamente, entonces es de esperar que sepa mucho más de lo que sabía anteriormente sobre las entrañas de los sistemas Linux. Este apéndice pone fin a todos estos temas, hace algunas sugerencias sobre lo que puede hacer con su flamante conocimiento, e intenta jugar un poco a ser vidente, comentando las tendencias futuras.

Escapada en solitario: configuraciones adicionales

Los casos de estudio de este libro le han presentado tres configuraciones de sistemas bastante habituales. ¿Adónde dirigirse desde aquí? Bien, con un poco de creatividad, puede crear algunas cosas realmente interesantes. Aquí están algunas sugerencias:

- **Comience a partir de la configuración de escritorio e instale los guiones iptables de la configuración del cortafuego.** Con algunos ajustes menores, puede montar un sistema de escritorio que funcione como un cortafuego simple para una red, lo que le ahorrará tener que disponer de un ordenador independiente para el cortafuego.

- **Comience a partir de la configuración de cortafuego e instale un servidor web y de correo electrónico (SMTP).** (¡Aún así, permanezca siempre atento a la seguridad!) Con una conexión DSL o un módem de cable adecuado, esta es una buena manera de hacer funcionar su propio dominio de Internet desde su hogar u oficina.
- **Como un proyecto más avanzado, construya su propia red privada virtual (VPN) mediante el uso de** dos **sistemas Linux en distintas ubicaciones.** (El capítulo 16 comentó esta posibilidad brevemente mediante el uso del protocolo PPP sobre SSH; puede comenzar a partir de este punto o emplear un paquete de software IPSec estándar.) Este puede resultar tan simple o tan complicado como necesite que sea, y puede hacer lo que quiera conectando simplemente su PC doméstico con la red de su oficina para conectar permanentemente dos sitios distintos en forma de "subred por extrusión" (*extruded subnet*).

Con un poco de experiencia y creatividad, se sorprenderá incluso a sí mismo con las cosas realmente ingeniosas y prácticas que puede hacer con los sistemas operativos basados en Linux. Es en verdad una sensación increíble la que sentirá cuando domine lo salvaje que hay en Linux y extraiga de ello una solución elegante.

Educación continua

No es un secreto que el mundo Linux cambia a gran velocidad. Como resultado, el nivel de dominio que alcance será directamente proporcional a la cantidad de tiempo que emplee educándose a sí mismo. Por eso, si es realmente serio respecto a su intención de dominar Linux, tiene que estar deseando emplear tiempo en mantenerse informado. Esta sección incluye algunas fuentes que pueden serle de gran utilidad.

Lista del núcleo Linux

Si quiere codearse con los verdaderos expertos de Linux, necesita seguir la lista de correo de los programadores del núcleo, conocida como la lista linux-kernel (y a veces abreviada como "l-k").

Gran parte del contenido es de naturaleza extremadamente técnica pero tiene mucha información útil para los no desarrolladores sobre problemas y tendencias relativas al núcleo, así que merece la pena prestar atención a esta lista. Si desea información sobre la suscripción a la lista linux-kernel puede consultarla en vger.kernel.org/majordomo-info.html.

La lista genera una cantidad de tráfico muy alta, así que puede estar interesado en consultar en su lugar alguno de los excelentes sitios de resumen, como Kernel Traffic en kt.zork.net/kernel-traffic/.

Listas de correo de las distribuciones

Otro recurso de valor incalculable es la lista de correo para su distribución. Típicamente, cada distribución tendrá listas separadas para la versión con calidad de producción y para cualquier otra versión beta o en pruebas que esté en desarrollo. Si está realmente interesado en estar al tanto de las tendencias y desarrollos futuros, siga las listas beta, porque es ahí donde está la acción. Los siguientes enlaces pueden iniciarle en las listas de correo de cada distribución de muestra comentada en este libro:

- **Red Hat Linux:** www.redhat.com/mailing-lists/
- **Slackware Linux:** slackware.com/lists/
- **Debian GNU/Linux:** www.debian.org/MainlingLists/

Quizás la mayoría de ellas le resulten útiles, tenga una mentalidad abierta. Muchos usuarios frecuentemente se familiarizan con su distribución favorita y tienden a ignorar o incluso a desestimar (¡o cosas peores!) las demás distribuciones. Este es un error para cualquiera que realmente desee dominar un sistema, ya que siempre habrá compromisos en una distribución dada, y cada una es mejor en algún campo. Cada vez que se vaya a preparar para actualizar su sistema, considere probar alguna otra distribución; puede que la encuentre superior y decida cambiar a ella, y si no, al menos recordará lo buena que ya es la distribución que utiliza. Con toda probabilidad simplemente aprenderá algo nuevo a lo largo del proceso. Después de todo, ¡yo no he aprendido lo suficiente como para escribir este libro pasando todo mi tiempo con una única distribución!

Seguir la pista del trabajo activo y diversas iniciativas

El mundo Linux es un lugar muy activo. Lo crea o no, este libro en realidad se centra exclusivamente en las aplicaciones tradicionales de los sistemas Linux. De hecho, hay una gran cantidad de investigación e ingeniería dedicada a lanzar a Linux (el núcleo) por caminos que pocos sistemas operativos (o quizás ninguno de ellos) se han atrevido a recorrer. Aquí comentaremos algunas de estas iniciativas, tradicionales e innovadoras, que pueden tener su impacto sobre el futuro de Linux.

En busca de la seguridad

El mundo de hoy en día es un lugar cada vez más complicado y peligroso, y en ninguna parte resulta esto más evidente que en el mundo de la informática. La seguridad de la información se ha convertido en la prioridad principal de muchas

de las principales empresas, y sin duda el interés no dejará de crecer. Linux, por supuesto, recibe su parte de esta atención. La U.S. National Security Agency (NSA), por ejemplo, encabeza un esfuerzo orientado al desarrollo de un núcleo Linux de seguridad mejorada (*"Security-Enhanced" Linux*, SE-Linux). Puede encontrar más información en www.nsa.gov/selinux/. La NSA también apoya múltiples proyectos de menor escala relacionados con la seguridad. Algunas compañías privadas y otras organizaciones también están desarrollando tecnologías como sistemas de archivos cifrados, redes seguras y tecnologías VPN, y características de seguridad a nivel del núcleo.

A medida que la seguridad de la información vaya cobrando relevancia en la industria, es de esperar que un progreso similar tenga lugar en el mundo Linux.

Mejoras del núcleo

Hay mucho debate hoy en día (en parte objetivo, en su mayor parte nada objetivo) sobre si Linux (como núcleo) se encuentra "preparado para la empresa". La respuesta es probablemente una cuestión de opiniones profesionales y, como todo lo demás, hay cosas en las que Linux es bueno y cosas en las que resulta bastante pobre. Esta situación también se aplica a si Linux está "preparado para el escritorio".

Sin embargo, es una verdad indiscutible que se está realizando una gran cantidad de esfuerzo de ingeniería para mejorar el rendimiento del núcleo Linux en un cierto número de áreas. Algunas de estas mejoras tratan de convertir las cosas buenas en algo mejor, y otras se orientan hacia las debilidades más pronunciadas. Hay un esfuerzo activo en ambas áreas de las prestaciones como servidor y como escritorio.

Por ejemplo, hay un cierto número de proyectos dedicados a la mejora del rendimiento del núcleo y de sus características como servidor. Hay en desarrollo varios sistemas de archivos de tipo *journaling*, en un estado que va desde los sistemas listos para un entorno de producción a la profunda investigación académica. En la parte del escritorio, hay diversos esfuerzos orientados a mejorar la receptividad de la "máquina virtual" central para proporcionar un rendimiento más fluido en los sistemas de escritorio.

Es de esperar que a medida que se vayan uniendo empresas al movimiento del software libre y el software de código abierto (y dados los recursos financieros que las acompañan) asistamos a un gran progreso de Linux y los sistemas operativos basados en Linux en los años venideros. Quién sabe, quizás algún día Linux realmente consiga dominar el mundo.

Recorte y compresión del sistema

Hay quien dice que hay un cambio radical en ciernes en la industria informática. Señalan al rápido crecimiento de los mercados de sistemas inalámbricos y "siste-

mas incrustados" (*embedded systems*) como una prueba de que el mundo está emigrando desde los sistemas de escritorio con hilos hacia el concepto no muy bien definido de "informática móvil". Aquí, una vez más, se está manipulando Linux, recortándolo y comprimiéndolo, para que encuentre su lugar en este potencial nuevo orden del mundo.

Como en cualquier nueva área, este es un campo extremadamente nebuloso. Es difícil precisar exactamente qué significa "informática incrustada", por ejemplo. Sin embargo, dadas las tendencias actuales, puede estar bastante seguro de que no importa hacia donde se dirija el mundo, también Linux estará allí. Hay un cierto número de empresas que trabajan con la intención de escalar Linux para que quepa en dispositivos incrustados, e incluso hay varias empresas que hacen públicos dispositivos embarcados de mano reales que funcionan mediante Linux.

Nota

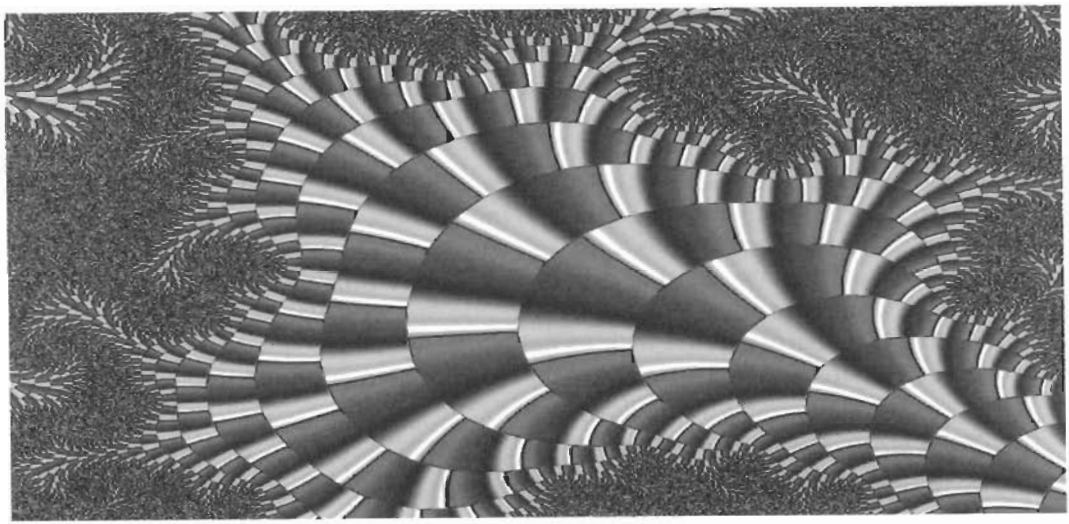
Ya hay al menos un PDA basado en Linux disponible en el mercado, concretamente, el producto Zaurus SL-5500 de Sharp, que puede encontrar actualmente en su tienda de electrónica local. Puede encontrar más información sobre Zaurus en el sitio web de Sharp (www.sharp-usa.com).

Realmente, el propio Linux (esto es, el núcleo) es bastante enjuto y humilde en tamaño como para comenzar a partir de él, así que se requiere una cantidad de esfuerzo comparativamente pequeña para conseguir que Linux funcione en estos dispositivos incrustados. Sin embargo, ya debería haberse dado cuenta de que la mayor parte del trabajo real no lo hace el núcleo, sino la distribución que funciona sobre él. Por eso, la tarea del desarrollo de un Linux incrustado es menos una labor de desarrollo del núcleo que de desarrollo de la distribución (la interfaz de usuario, las bibliotecas del sistema y demás). Esté atento y verá como una gran cantidad de trabajo se desarrolla igualmente en esta área en los años venideros.

Resumen

Linux está en constante evolución. Puede que el mundo futuro de la informática sea simplemente como en la actualidad pero con ordenadores más rápidos, o puede que resulte radicalmente distinto. Sea como sea, es bastante probable que Linux esté ahí de algún modo. Manténgase al tanto de las cosas, siga informado y ¡disfrute del viaje!

Bienvenido al mundo de Linux.



Apéndice B
Contenido del CD-ROM

El CD-ROM que acompaña a *Configuración de sistemas Linux* contiene todos los archivos de configuración y los guiones shell que se describen en el libro.

Están organizados por capítulo, por lo que no deberías tener ninguna dificultad para encontrar lo que buscas, de todas formas aquí tienes un índice.

Directorio:	Archivos:	Destino Propuesto:
Capítulo 8	Listado-8-1.sh	/etc/init.d/sshd
Capítulo 10	Listado-10-1.txt	/etc/socks.conf
Capítulo 12	Listado-12-1.txt	/etc/xinetd.d/cvs
Capítulo 13	Listado-13-1.sh	/etc/profile.d/java.sh
	Listado-13-1.csh	/etc/profile.d/java.csh
Capítulo 15	Listado-15-1.sh	/etc/profile.d/java.sh
	Listado-15-1.csh	/etc/profile.d/java.csh
	Listado-15-2.sh	/usr/local/java
Capítulo 16	Listado-16-1.sh	/etc/init.d/firewall
	Listado-16-2.sh	/etc/sysconfig/firewall
	Listado-16-3.txt	/etc/dhcpd.conf
	Listado-16-4.txt	/etc/named.con :

Observaciones a tener en cuenta por el lector:

Listado-13-1.sh es en realidad el listado que aparece en el libro como Listado 13-1; mientras que Listado-13-1.csh es el mismo archivo en sintaxis csh para su comodidad.

Listado-15-1.sh es una versión actualizada del Listado-13-1.sh para el material del Capítulo 15. (Listado-15-1.csh es el mismo archivo en sintaxis csh)

Soporte Técnico de Anaya Multimedia

Si tiene algún problema relacionado con el contenido de nuestras publicaciones, o con el material suministrado en las mismas, por favor comuníquelo a nuestro **Departamento de Soporte Técnico** a través de alguno de estos medios:

- TELEFONO, en el número **91 320 90 36**. El horario es de **Lunes a Viernes desde las 9 a las 15 horas**. En el resto del día existe un contestador automático donde podrá dejar sus mensajes, que serán respondidos a la mayor brevedad posible.
- FAX, en el número **91 320 44 19**.
- CORREO A NUESTRA DIRECCIÓN, **C/ Juan Ignacio Luca de Tena 15. 28027 Madrid**.
- CORREO ELECTRÓNICO a la dirección, **a_multimedia@anaya.es**.

Anaya Multimedia dispone de un servicio especial mediante el cual, cualquier persona que disponga de una DIRECCIÓN DE CORREO ELECTRÓNICO puede recibir información detallada de las nuevas publicaciones de nuestra editorial.

Si está interesado en suscribirse, puede hacerlo directamente a través de nuestro Web:

http://www.anayamultimedia.es

Una vez aquí, diríjase a la sección **Servicios** y después, al apartado **Novedades a usuarios**.

También puede contactar con nosotros a través de cualquiera de los medios antes mencionados.

Índice alfabético

/, explicación del directorio, 67
/boot, directorio, 66, 100
/bin, directorio, 66, 100
/etc, 66
    en Red Hat Linux, 100
    tabla de subdirectorios significativos en Debian, 195
/etc/alternatives, comprensión de, 197-198
/etc/auto.master, 385
/etc/bashrc, archivo de configuración, 127
/etc/csh.login, archivo de configuración, 127
/etc/cshrc, archivo de configuración, 127
/etc/default, directorio de Debian, 194-195
/etc/fstab, 167, 353-354, 385
/etc/inittab, archivo, 105-110, 144-147, 191-192, 202
    en Debian GNU/Linux, 191-192, 198-199
    en Red Hat Linux, 102, 105-110
    en Slackware Linux, 144-147, 151-152
/etc/network, tabla de contenidos en Debian, 196
/etc/nsswitch.conf, archivo, 381
/etc/passwd, mecanismo, 133, 307-308, 380-381
/etc/rc.d/rc.0 y /etc/rc.d/rc.6, 148-149
/etc/rc.d/rc.inet1 y /etc/rc.d/rc.inet2, guiones, 149-150
/etc/rc.d/rc.K, 147
/etc/rc.d/rc.local, 105, 113, 167
/etc/rc.d/rc.M, 147-148
/etc/rc.d/rc.S, 147
/etc/rc.d/rc.sysvinit, 105, 148, 150, 152, 243

/etc/security/console.perms, archivo, 133
/etc/services, archivo, 156
/etc/X11/default-display-manager, archivo, 199
/home, directorio, 67, 100
/lib, directorio, 66, 100
/mnt, directorio, 67, 100
/mnt/zip100.0, punto de montaje, 354
/opt, directorio, 100
  instalación en, 213-214
/sbin, directorio, 67, 100
/sbin/mingetty, programa, 109
/tmp, directorio, 67, 100
/usr/bin/updfstab, 131
/usr, directorio, 67, 100-101
/usr, explicación del directorio, 67
/usr/local, directorio, 100
  instalación en, 212-213
/usr/src/linux-2.4, enlace simbólico, 125
/usr/src/redhat, contenidos del directorio, 223-224
/usr/X11R6, directorio, 67
/var, directorio, 67, 100-101
--, sintaxis de parámetros, 216
--enable-mods-shared, 281
--enable-so, opción, 281
--nodeps, opción de RPM, 92, 96
--prefix, opción
  precaución, 222
-M, opción del comando man, 291
-o loop, argumento, 364
-p, argumento de mkdir, 305, 324
-preferxf4, argumento, 118
./configure, programa, 221
  opciones comunes, tabla de, 221
.K, extensión de /etc/rc.d/rc.K, 147
.M, extensión de /etc/rc.d/rc.M, 147-148
.sh y .csh, diferencia entre extensiones, 127-128
.so, extensión de bibliotecas compartidas, 115, 262-263
  ubicación estándar, 66
.spec, archivos, 223
.src.rpm, extensión de RPM fuente, 222
.tar.bz2, extensión de archivo, 140
.tar.gz, extensión de archivo, 140
.tar.Z, extensión de archivo, 140
.tgz, extensión de archivo, 140
.ttf, extensión, 131
.txt, extensión como tipo MIME, 288
{}, uso en biblioteca SOCKS Dante, 267
<*Directory* Y>, 292
  {} y, 293
#, comentario de líneas, 162, 449
802.11b, estándar IEEE, 457
8086, 210

A

ab, 285-286
Abiword, 344-345
Acceso remoto como root, 386-387
access_log, archivo, 289
account, pila PAM, 256
Advanced Server, edición de Red Hat Linux, 122
alien, programa Debian, 187-188
Alpha 21164pca, 40
Alternativas a programas en Debian GNU/Linux, 197-198
Ámbitos, 293
anaconda, programa, 120
ant, 389-391
    formato de archivos de construcción, 389
Apache, 275-297
    bloque *Directory*, 292-293
    configuraciones de ejemplo, 288
    directorios, 284-291
    directorios de instalación de, 278-279
    habilitar webdav, 283
    httpd.pid, archivo, 289
    inhabilitar, 163-164
    magic, archivo, 288
    manual de usuario, 290
    mime.type, archivo, 288
    módulos DSO, 280-281, 286
    módulos MPM, 281-282
    niveles de ejecución, 295-296
    objetivos, 275-276
    origen, 275
    PID, 289
    programas de utilidad, tabla de, 285
apachectl, 285-286, 290, 294-296
    arranque en Slackware Linux, 296
    compatible con chkconfig, 295-296
    manual, 290
Applixware, 344-345
apt-get, uso del programa, 176-178
    apt-get dist-upgrade, 156
    comparación con up2date, 182
    operaciones comunes, tabla de, 181
apxs, 285-286
arch/i386/bzImage, ubicación del núcleo al crear uno nuevo, 126
Argumentos, elemento en reglas PAM, 253
Asset Insight, 379, 386
Atómica, operación, 86
auth, pila PAM, 255-256
autoconf, tabla de opciones comunes, 221
autofs, servicio, 384-385
    particularización de configuración, 385

Avifile, 367-368
DLL y, 368
awk, 197
gawk, 197, 420

B

Base Estándar de Linux, 80
bash, 115-116, 153, 164-165, 192
implementación de sh, 127, 331
límite, 392
ubicación, 66
Berkeley
Berkeley Internet Name Domain, 444
distribución de sistemas, 30
University of California, 30
Biblioteca de componentes
GTK+, 75
Qt, 63, 75
Bibliotecas
compartidas, 262-263
de usuario, 219
nomenclatura de, 72
bind, 442, 444-445, 458-459
archivo de configuración, 445
configuración, 444-445
descarga, 444
Bit adhesivo, 308-310
ventajas e inconvenientes, 394
BlackBox, 342-343
Bloqueo de archivos NFS, demonio, 384
BOOTPROTO, variable de DHCP, 379
BSD, 30-32, 35-36
BSDi, 31
Darwin, 36, 41
estructura de archivos, 65-66
licencia, 52-54
modelo de guiones de arranque, 101
archivos de, 144-151
ejecución, 151-152
en Slackware Linux, 151-152
en Slackware Linux, 144-152
Slackware Linux y, 144
Bucle de retorno local, 434
Bugtraq, lista de correo, 453
BUILD, subdirectorio de /usr/src/redhat, 224

C

Caja de arena y CVS, 299-300
Caldera, distribución, 134-135
Cargadores de arranque, 122-123

CDE, 117
cdrecord, 360, 364-365
  cdrecord -scanbus, comando, 365
CF (*Compact Flash*), tarjetas, 357
CGI, estándar, 290, 401
  lenguajes, 290
changelog, archivo de paquete Debian, 227
checkgid, 285-286
chgrp, comando, 308
chkconfig, 114-115
chmod, comando, 309
chown, comando, 308
Cinta, archivo de, 88-89
COM, modelo de componentes, 77
Compiladores, 72, 210, 214
Comprobación estricta de claves, 238
Configuración global, 217
Continuo Linux, 57-59
Control
  archivo de paquete Debian, 227
  elemento en reglas PAM, 252
copyleft, 51
copyright
  archivo de paquete Debian, 227
  referencia en licencia BSD, 52
Cortafuego
  arranque, 433-436
  configuración de syslog en, 449-450
  detención, 436-437
  DHCP para máquinas proteguidas por, 439-444
  dirección IP de un servidor DNS, 441
  DNAT, 412-414
  DNS, mejora, 458-459
  documentación electrónica, 462
  estado, 437
  guión de configuración de, 425-437
  inhabilitación, 423-424
  lista de paquetes, 419-421
  niveles de ejecución, 427
  objetivos, 409-414
  por defecto de Red Hat Linux, 132-133
  reinicio, 437
  tirar o rechazar paquetes, 451-452
  variables de configuración, tabla de, 429-430
  variables DNAT, 431-432
cpio, 88-89
cron, 130
CrossOver, extensión para navegadores, 368-369
Cuentas de usuario
  cuenta única, 307
  cuentas del sistema, 306-307
  en CVS, 306-311

CVS, 299-318
actualización de /etc/services en Red Hat Linux, 312-313
autenticación híbrida, 311
directorio de repositorio, 305
funcionamiento, 299-300
manual, 300
modos, 302-304
opciones en tiempo de compilación, tabla de, 302
parámentros, 311-312
del archivo de configuración, 315-316
permisos de usuarios, 307-311
selección de modo, 304
CVSROOT, variable de CVS, 310-311

D

Dante, biblioteca, *Véase* SOCKS Dante, biblioteca
db2 , db3, paquetes de Red Hat Linux, 116
dbmmanage, programa, 285, 287
dc2xx, controlador para cámara digital, 357
Deb, archivos, 64, 173
construcción, 187
debconf, uso, 184
Debian GNU/Linux
actualización del sistema, 185-186
administración
de código fuente, 187
de paquetes en, 179-185
archivo de paquete Debian, 227
bibliotecas centrales, tabla de, 192-193
contrato social Debian, 172-173
directorios
de configuración, 194-198
SysV, 189-190
formato de empaquetado, 174-175
gestor de pantalla X, 198-199
herramientas
de paquetes, 175
SysV propias, 190-191
instalación de paquetes no Debian, 187-188
manipulación de archivos de paquete, 186-187
niveles de ejecución, 191-192
políticas Debian, 173-174
proteger, 199-201
X Windows, particularidad, 192
Dejar caer configuraciones, 111-112
Dependencia, comprensión, 90
Desarrollo
software
herramientas, 388
ant, 389-391
JBuilder, 390-392

tradicional, 401
versión del núcleo, 61
Deshonesto, colaborador, 53-54
DEVICE, variable de DHCP, 379
DHCP
bloque
de máquina, 442-443
de red compartida, 443
dhcpd, 439-441
opciones globales, 441-442
usufructos, 441, 443
Direcciones IP, rangos, 431
DirectoryIndex, directriz, 293
disable, parámetro de configuración de xinetd, 314
Disquetera ZIP, 351-354, 357, 377
advertencia sobre particiones, 353
IDE, 352-353, 359
montaje, 353-354, 385
particiones, 354
puerto paralelo, 352
SCSI, 351-352
USB, 351
Distribución
aspectos, 60-61
objetivos, 60
DLL
infierno, 96
y Avifile, 368
DNS
arranque de servidores, 69
cortafuegos y, 414, 439
logresolve y, 286
necesidad del servidor, 149
similitud de services con, 156
Dominio de Internet, 466
DOOM, 369
dpkg, uso del programa, 175-176
operaciones comunes, tabla de, 180
dpkg-deb, uso, 186-187, 227
operaciones comunes, tabla de, 187
dpkg-preconfigure, 184
dpkg-reconfigure, 184-185, 199
DPT=, valor de anotaciones, 451
dselect, uso del programa, 178, 183
DSO, 280-281, 286
seguridad, 281
DST=, valor de anotaciones, 451

E

EJB, servidor, 395
emacs, 119, 127

Empaquetado Debian, 64, 174-175
archivos de un paquete con, 227
conversión de archivos a, 65
medios fuente, administración, 177
Enmascaramiento, 409, 411-414, 435
error_log, archivo, 289
Escritorio, definición, 79, 338
Estable, versión del núcleo, 61
Estación de trabajo frente a escritorio, 79
Ethernet, tarjetas, 415-418
conexión de marcado y, 416-417
configuración de interfaces de, 417-418
nombrado, 415-417
Ewing, Marc, 83
Extensiones del navegador, 348-350
extiface, significado de la variable, 429-430
extip, significado de la variable, 429-430

F

FHS, estándar de jerarquía del sistema de archivos, 66-68, 202, 212
elección de directorios de instalación, 211-212
relación
con Debian GNU/Linux, 173-174, 194
con Red Hat Linux, 100-101
con Slackware Linux, 143-144
resumen del estándar, 66-67
fileutils paquete en
Debian GNU/Linux, 192
Red Hat Linux, 116
Slackware Linux, 153
Filtrado de red, 412
find, comando, 68
Flanagan, David, 321
flex, herramienta de desarrollo, 388
Formato de empaquetado
explicación, 63-64
herramienta mejor, 189
y distribuciones, 64-65
fortune, programa, 73
forward only, línea de named.conf, 445
FORWARD, cadena en reglas de cortafuego, 433
forwarders, bloque de named.conf, 445
FSF, 34, 41 , 45-46
Fuentes de paquetes, gestión en Debian, 181
fvwm, 342-343

G

Galeon, navegador web, 348
gcc, compilador, 155, 214, 388, 419
gdbm paquete en
Debian GNU/Linux, 192

Red Hat Linux, 116
Slackware Linux, 153
gdm, 198-199
ghostscript, 132
GIMP, 363
glibc, biblioteca, 72-73
Globalizar con dpkg, 180-181
GNOME, 75-79
gnoRPM, 98
GNU
FSF, 34, 41, 45-46
GNU/Linux, debate, 40
historia, 34-40
núcleo HURD, 35, 41
GNUCash, 345-346
GNUStep, 342-343
GPG, 97
gphoto
necesidad, 357
uso, 362-363
Grabadora de CD, 359-360, 363-365
configuración, 360
programas útiles, 364
groupadd, 308
GRUB, cargador de arranque, 122
Guión de cortafuego y conexiones PPP, 430
Guiones de arranque, 69-71
niveles de ejecución, 70
gzip, programa, 140

H

HancomOffice, 344-345
htdigest, programa, 285
htdocs, directorio de Apache, 284
htpasswd, programa, 285
httpd
de Apache, 285-286
original del NCSA, 275
httpd.pid, archivo, 289
HURD, 35, 41

I

IANA, 313, 431
icons, directorio de Apache, 284
id Software, 369
IDE, nomenclatura, 352
ide-floppy, módulo, 352-353
ide-scsi, módulo, 353, 359-360, 363
IDS, 455-457
enfoques, 456
IETF, 231, 313

ifconfig, comando, 429
  ubicación, 67
imm, controlador, 352
inetd
  archivo de configuración, 157-159
  configuración en Slackware Linux, 156-159
  inhabilitar servicios, 161
  instalación de un nuevo servicio, 159
  servidores arrancados mendiante, 218
Infierno DLL, 96
init, programa, 104
initdefault, nivel de ejecución por defecto, 109
INPUT, cadena en reglas de cortafuego, 433
Instalación,
  autoconf, 220-222
  fuentes, 209-211
  herramientas
    de Debian GNU/Linux, 226-227
    de Red Hat Linux, 222-224
    de Slackware Linux, 224-226
  ubicación, 211-214
installpkg, 141
Integración de red inalámbrica, 457
Intermediario, ataques del, 237-238
intiface, significado de la variable, 429-430
intip, significado de la variable, 429, 431
intserver, significado de la variable, 429, 431
invoke-rc.d, uso de, 191
ipchains, 132, 399, 414, 423-425, 448-449
  documentación electrónica, 414
  frente a iptables, 424
IPSec, 466
iptables, 133, 195, 255, 412-414, 424-427, 433-438, 448
Irix, 30
ISC, 439-440
  bind, 444-445, 458-459
  dhcpd, 439-441
ISO-9660, 364-365
  sistema de archivo iso9660, 383

J

J2EE
  ejecución con Tomcat de aplicaciones, 394-395
  JSP, 395
  servidor EJB, 395
  servidores alternativos, tabla de, 400
J2SDK, 321-333
  guión de configuración total, 330-331
  licencia SCSL, 322
  variable
    CLASSPATH, 327-329

JAVA_HOME, 329-330
PATH, 329-330
JAR
instalación de bibliotecas adicionales, 391
permisos, 394
uso de muchos archivos, 392-393
Java
archivo de configuración mejorado, 391-392
bibliotecas adicionales, 391
java.sh, archivo de ejecución de Java extendido, 391-392, 393
JBoss, servidor, 400
JBuilder, 390-392
JDK
advertencia sobre distintos, 330
distinción entre J2SDK y, 321-322
instalación de múltiples, 326
Joliet, estándar, 364
Journaling, 48-49, 120, 468
JSP, 395, 400
Juegos, 369-370
JVM, 321, 357, 392

K

Kaffe, 73
Kamera, 363
Kapital, 346
KDE, 75-79
ejemplos de temas, 77-78
iconos automáticos, 354
kdm, 165, 198-199
Kernel Trafic, sitio web, 466
kernel-headers, 125
kernel-source, 125
kickstart, programa, 121
Konqueror, 346-347, 349
extensión Java, 350
extensión Macromedia Flash, 349
gphoto y, 363
kpackage, 98
ksh, shell, 164
Kudzu, programa, 130-131
KXXfoo, patrón de nombrado SysV, 111

L

LAME, 366
LAN inalámbrica, 417-418
LDAP, servidor, 79, 116, 133, 248, 422
LDP (*Linux Documentation Project*), 126
Lector de tarjetas USB en Slackware Linux, 167
lex, 388
libstdc++, bibliotecas de desarrollo, 388

libtermcap, 420
Licencia
  aceptación, 208
  Apache, 274
  BSD, 52
  GPL, 51-52
  LGPL, 53
    bibliotecas y, 53
  MPL, 53
  vírica, 52
Licencias de código abierto, utilidad, 50
LILO, cargador de arranque, 123
Linux y GNU, 34-36
Listas de correo
  Bugtraq, 453
  de distribuciones, 467
  l-k, 466
  linux-kernel, 466
listen-on, línea de named.conf, 445
logresolve, 285-286
lokkit, comando, 132
losetup, 420
lpd, inhabilitar demonio de impresión, 162
ls, 37
LSB (*Linux Standards Base*), 80
lsmod, comando, 385
ltrace, herramienta de desarrollo, 388

M

MAC, dirección, 442-443, 451
MacOS, 79
make
  compilación, 215
  frente a ant, 389
  menuconfig, comando, 126
  utilidad, 215
makepkg, 142
Man-in-the-middle, 237
Mandrake, distribución, 134
MANPATH, variable de man, 291
Maximum RPM, 97, 224
md5, 94-95
Mesa, implementación de OpenGL, 361
mi-software, paquete de ejemplo
  construcción y instalación en Debian GNU/Linux, 226
  construcción y instalación en Slackware Linux, 224-226
MIME, estándar, 288
Minix, 41
mkdir
  creación del repositorio CVS con, 305, 310
  uso con J2SDK, 324

mkfontdir, comando, 131
mkinitrd, comando, 124, 126
mkisofs, 364-365
modprobe, comando, 359, 385
modules, directorio de Apache, 285
Módulo, elemento en reglas PAM, 253
Monousuario
  incapacidad de montar partición /usr/local, 112
  modo, 70, 102
  nivel de ejecución en
    Red Hat Linux, 103
    Slackware Linux, 145
  sistemas, 32
Montador automático, 384-386
mount, comando, 67, 352, 364, 383
Mozilla, 346-350
  extensión CrossOver, 368-369
  extensión Java, 350
  Netscape Communicator, relación con, 349
MP3
  estándar y archivos, 365-367
  reproductores, 354-358, 377
MPEG, 367-369
  MPEG-2 Layer 3, 358
mpg123, 366
mpg321, 366
MPM, 281-282
Multics, 30
Multiusuario
  instalación en sistemas, 212-213
  modo, 70
  nivel de ejecución en Red Hat Linux, 102-103
  nivel de ejecución en Slackware Linux, 145

N

NAT, 412-414, 435, 446
  DNAT, 412-414
  documentación electrónica, 462
  NAT
    de destino, 412
    de fuente, 413, 435
  SNAT (*Source Nat*), 435
NCP, protocolo, 385
NCSA, 275
netconfig, programa, 379, 417
netfs, servicio, 385
NFS NSM, servicio, 384
NFS, uso, 382-386
nfs-utils, paquete, 384
nfslock, servicio, 384
nice, parámetro de configuración de xinetd, 315

NIS
NISDOMAIN, variable de, 381-382
relación con NFS de, 382-386
uso, 380-382
yellow pages, 381
Niveles de ejecución,
comprensión de, 102-103
cortafuegos y, 427
Debian GNU/Linux y, 189-192
guiones de arranque y, 69-71
por defecto en Red Hat Linux, 109
Red Hat Linux y, 103, 108, 112-114
Slackware Linux y, 145-146, 148
tabla de, 70
nmap, 421-422
NSA, 468
Núcleo Linux, 61-63, 123-127
árboles, 61-62
construcción a partir del código fuente, 126-127
documentación electrónica, 126
en Slackware Linux, 139
enlace simbólico, 125-126
fragmentación, 63
lista de correo sobre, 466
mejoras, 468
paquetes necesarios para construcción, 125
pasos de instalación en Red Hat Linux, 124
visión, 63

O

Ogg, 365-367
compañía Xiphophorous, 365-366
paquetes Vorbis, 365-367
oggenc, programa codificador, 366
OHCI, 355
ONBOOT, variable de DHCP, 379
OpenGL, 361
OpenOffice, 132, 218, 343-344, 377, 390, 396-397
OpenSSH, 231-244
acceso remoto mediante, 370-371, 386-387
actualización, 235-236
autenticación, 233-234
claves, 235, 238239
cliente, 236
OpenBSD, 231
provisión de otros servicios, 371-372
reenvío por agente, 236-237
servidor, 239
variable X11Forwarding, 240
OpenSSL, 176, 219, 232, 420
Opera, navegador web, 348

optional, valor del elemento control, 252
Orion, servidor, 400
OSI, 51, 54
other, servicio por defecto de PAM, 256-257
OUTPUT, cadena en reglas de cortafuego, 433

P

PAM, módulos de autenticación enlazables, 247-258
  archivo de ejemplo, 255-257
  configuración en Red Hat Linux, 133-134
  credenciales de usuario, 248
  instalación, porque no se detalla, 248-249
  pila por defecto, 257
  pilas, 253-255
  reglas, 250-253
Parámetros
  de red para SOCKS, 267
  en línea de comandos, 216
    -- , sintaxis de palabras completas, 216
Particiones, 68-69
PCI, buses, 356
PDA, 36, 59, 73, 469
  basado en Linux, 469
  PocketLinux y Kaffe, 36, 73
Peanut Linux, distribución, 167-168
Per Cederqvist, 300
Perl, 401
Personal, edición de Red Hat Linux, 121
PHP, 401
pidof, 104-105
Pilas PAM, 253-255
pkgtool, 140-141
Planos, directorios o archivos, 218
plaympeg, programa, 368
portmap
  inhabilitar servicio, 161-162
  instalación del servicio, 380
Posix, especificación, 30, 66, 72, 102
PowerTools, CD, 194
ppa, controlador, 352
prefdm, programa, 110
Professional, edición de Red Hat Linux, 121
python-xmlrpc, 420

Q

Quake, 369

R

RADIUS, servidor, 116, 133
Raíz, directorio, 67

Raymond, Eric y el modelo de código abierto, 47
rc.modules, 145, 147, 151
    función, 149
rc.netdevice, 149, 151
RCS, 299
Read-Only Linux, sitio web, 459
Readme, archivo, 208
Red Hat Linux
    bibliotecas centrales, tabla de, 116
    calidad, 120
    cambios al núcleo estándar en, 85
    cargador de arranque, 122-123
    CD de fuentes y documentación, 121
    directorios de configuración, 129-130
    distribución central, 120-121
    ediciones, 121-122
    entorno de usuario, 127-128
    FHS, relación con, 100-101
    gestión de software similar en, 123
    herramientas SysV propias, 114-115
    initscripts, paquete, 105-107
    instalación de fuentes TrueType, 131-132
    kudzu, 130-131
    núcleo estándar, 123-127
    objetivo de la distribución, 84
    proteger, 132-134
    SysVinit, paquete, 104-105
    up2date, 98, 177
    Xconfigurator, 118, 154, 184, 202
    XF86Config, 118
    xinetd, 128-129
Red Hat, negocio, 84
Red local, comprensión, 378-379
    direcciones IP, rangos, 431
redir_ports, significado de la variable, 429, 431
Reinos de Apache, 287
ReiserFS, 120
REJECT, 452
removepkg, 142
Repositorios CVS críticos, 306
    permisos, 307-311
required, valor del elemento control, 252
requisite, valor del elemento control, 252
resolveprotocol, opción de biblioteca SOCKS Dante, 268
Retroadaptación, 62
Rio 500, proyecto, 358
Rock Ridge, estándar, 364
root, explicación del directorio, 67
root, usuario, 158-159, 212-213
    shell bash, 116, 153, 192
ROOT, variable para instalación de paquetes en Slackware Linux, 145
rotatelogs, 285-287

RPC, 162, 380, 384
rpm -U frente a rpm -u, 93
RPM, 63-64, 85-101
  actualización, 93
  arquitecturas, tabla de, 87-88
  comando útiles, tabla de, 99
  comandos combinados, tabla de, 99
  convención de nombrado, 86-87
  dependencia, 90
  desinstalación paquetes, 96
  estándar de facto, 65
  firmados criptográficamente, 97
  formato del archivo, 88-89
  fuente, 64, 86, 95, 212, 222-223
  fuente, equivalencia en Debian GNU/Linux, 227
  gnoRPM, 98
  información paquetes instalados, 93-94
  información paquetes no instalados, 90-92
  instalación paquetes, 92-93
  kpackage, 98
  manual, 97, 224
  permisos, 89-90
  precaución con las mayúsculas, 93
  prestaciones adicionales de, 97
  reconstrucción paquetes, 95-96
  verificación paquetes, 94-95
rpm2cpio, 88
rpmfind, 122, 271
RPMS, subdirectorio de /usr/src/redhat, 224
rules, archivo de paquete Debian, 227

S

Saludo de los tres dedos, 106
SAMBA, 119
Sawfish, 342-343
SCCS, 299-300
scp, 231, 236, 371
SCSI, nomenclatura, 351-352
sd_mod, controlador, 352
SDL (*Simple DirectMedia Layer*), API, 370
SE-Linux (Linux de seguridad mejorada), 468
Seguimiento de correcciones en distribuciones, sitios web, 454
Seguridad
  avanzada, 455
  en profundidad, 407, 454-455
Sendmail
  inhabilitar servidor SMTP, 163
  inicio en Slackware Linux, 150
Serie de núcleos, 61
server, parámetro de configuración de xinetd, 314
server_args, parámetro de configuración de xinetd, 314

service, 114
Servicio, elemento en reglas PAM, 250-251
Servidor, 79
session, pila PAM, 256
sftp, 231, 236, 371
Shell de comandos, utilidad, 69-70
shlib, 262-263
shutdown, comando, 71, 115
Sintaxis de shell, 127-128, 330
Sistema
  de anotaciones remoto, 447
    configuración de anotador, 449-450
    configuración servidor interno, 447-448
    entrada típica, 451
    perforación cortafuego local por parte de, 448-449
  operativo
    bibliotecas de sistema, 33-34
    naturaleza, 32-34
    núcleo, 33
    recursos, 32-33
Slackware Linux
  añadir hardware nuevo, 166-167
  cambiar entorno de escritorio, 165-166
  entorno de usuario, 164-165
  filosofía de, 138
  formato *tarball* de, 140
  guión Xsession, 165-166
  guiones de arranque de, 144-152
  herramientas de instalación de paquetes, 140-143
  programa xf86config, 154
  proteger, 159-164
  sustituto de kernel-headers, 139
slang, 420
slocate, 420
SM (*SmartMedia*), tarjetas, 357
SMB, 119, 385
smpeg, paquete, 368
  plaympeg, programa, 368
SNMP, 443, 457
socket_type, parámetro de configuración de xinetd, 314
SOCKS Dante, biblioteca, 261-272
  archivo de ejemplo, 270-271
  archivos de configuración, 266
  bloque de ruta, 269
  formato de archivo, 268
  necesidad, 261-263
  parámetros de red, 267
  ruta de configuración, 264
socks.conf, archivo de configuración de biblioteca SOCKS Dante, 269
  ejemplo, 270-271
Software
  de codigo abierto, 43-54, 59, 83-84, 172-173, 208-209

frente a código libre, 172-173
desarrollo cooperativo, 48-49
libre, 36, 41, 43-47
licencias, 46-46
de código abierto, 49-54
naturaleza del software, 43-45
portable, 222
software horizontal, 45
SOHO, 408
Solaris, 30
SOURCES, subdirectorio de /usr/src/redhat, 224
SPECS, subdirectorio de /usr/src/redhat, 224
SPT=, valor de anotaciones, 451
SRC=, valor de anotaciones, 451
SRPMS, subdirectorio de /usr/src/redhat, 224
SSH, 231
cliente, 236
servidor, 239
ssh-add, programa, 236-237
ssh-agent, programa, 236-237
Stallman, Richard, 34, 45-46
strace, herramienta de desarrollo, 388
Subcomandos, 216
sufficient, valor del elemento control, 252
Suplantación, ataque de, 196
SuSE Linux, distribución, 167
switchdesk, 118, 165-166
sync, comando, 103
syslog, 148, 447-450
paquete sysklogd, 447
SysV, modelo de guiones de arranque
cierre de servicios en Debian, 199-200
configuración manual en Red Hat Linux, 112-113
ejecución en Red Hat Linux, 107-112
enlaces simbólicos, 113
patrones de nombrado de guiones, 111
Red Hat Linux y, 101
SYYbar, patrón de nombrado SysV, 111

T

Tanenbaum, Richard, 41
tarball, formato, 140
archivos en Slackware, 224
Tarjeta gráfica 3D, 360-361
tasksel, uso del programa, 178-179, 183
TCP y UDP, 157-158
tcpip1, paquete, 156
tcsh, implementación de csh, 127, 164, 331
telinit, comando, 71
Temas en GNOME y KDE, 77
TeX, 132

tinyproxy, servidor, 371
Tipo, elemento en reglas PAM, 251
Torvalds, Linus, 23, 35, 41, 48, 62
Transparencia de red
  instalación de paquetes y, 177
  X Windows y, 73-74
Tripwire, sistema de detección de intrusiones, 95, 420, 455-457
Troya, caballo de, 97
TTL=, valor de anotaciones, 451
ttmkfdir, comando, 131
tty
  combinaciones de teclas para conmutar, 108
  definición en Red Hat Linux, 108-109
  definición en Slackware Linux, 146
TurboLinux, distribución, 135
twm, 76

U

uCLinux, 73
UHCI, 355
Unix
  primera encarnación, 30
  propietario de la marca, Open Group, 32
  System V, 30
  Unics, 30
  versiones libres, 30
UnixWare, 32
up2date, 98, 122
  apt-get, relación con, 177, 182
  configuración de PAM, 255
  pila PAM de, 256
update-inetd, uso del programa, 200-201
update-rc.d
  tabla de uso de, 190
  uso de, 190-191
upgradepkg, 141-142
UPS, 108
USB
  cámara digital, 356-357
  conexión en caliente, 356
  controlador de la controladora, 355-356
  explicación, 354
  funcionamiento, 355-356
  lector de tarjetas, 357
  otros dispositivos, 358-359
  reproductor MP3, 358
  uso de dispositivos, 354-359
usb-core, controlador, 357
usb-storage, controlador, 357
user, parámetro de configuración de xinetd, 314
useradd, 308

V

Veillard, Daniel, 122
VFS, 48, 383
vim, 38
  establecer como alternativa de vi, 197-198
vmlinuz, archivo, 66
VMware, 396-398
Volkerding, Patrick, 137
VPN, red privada virtual, 459-461, 466
  proceso pppd, 460

W

W3C, 122
wait,
  parámetro de configuración de xinetd, 314
  programa, 104
WAP, 408, 417-418
  acceso a red Ethernet, 457-458
  configuración de dhcpd, 440, 443
  problemas, 458
WEP, esquema de privacidad, 457-458
wuftpd, 90

X

X Windows, 73-78, 116-118
  cambiar entorno en Slackware Linux, 165
  conjunto de componentes, 74-75
  consolas de texto alternativas en Red Hat Linux, 108
  gestores, 75-76
  modelo de capas, 74
  nivel de ejecución 4 en Slackware Linux, 145
  nivel de ejecución 5 en Red Hat Linux, 70
X.Y.Z, esquema de numeración del núcleo Linux, 61
xcdroast, 360, 364-365
xdpyinfo, 118
xemacs, 119
Xenix, 30
xev, 118
XFree86, 116-118
  extensiones de representación, 117
  herramientas relacionadas con, 118
xfs, 132
Xine, 369
xinetd, 128-129
  administración, 316
  motivo para el cambio, 128-129
  parámetros del archivo de configuración, 314-315
xinit, 147, 154, 165-166
XML, 300, 389
XMMS, 366-367

xserver-common, 184
xterm, 239
xvidtune, 118

Y

yacc, 197, 388
  bison, 197, 388
  byacc, 197
Young, Bob, 83
ypbind, 381

Z

Zaurus, 469
zcat, 38
Zen, 39, 66, 168
Ziegler, 405, 433, 438, 452, 462
ZIP
  disquete, 155
  formato de archivo, 88, 212, 325-328,
  formato de compresión, 140
  gzip, 89
  programa, 89
ZipSlack, distribución, 155, 167
zlib, 420
zsh, 97, 164
zsh, shell, 164
Zukowski, John, 321