

Fundamentos de Sistemas Operativos

Versión 2.0

Francisco Medina López

Este libro fué escrito en Linux usando L^AT_EX por su autor.

©2006 Francisco Medina López

Todos los derechos reservados por su autor. Este libro puede ser reproducido parcial o totalmente siempre y cuando se mantenga los créditos a los autores.

Francisco Medina López

Centro Coapa, DGSCA

Facultad de Contaduría y Administración

Universidad Nacional Autónoma de México

Índice general

1. Teoría de Sistemas Operativos	1
1.1. Introducción	1
1.2. ¿Qué es un Sistema Operativo?	2
1.2.1. El sistema operativo como una maquina extendida.	2
1.2.2. El sistema operativo como controlador de recursos.	2
1.3. Historia de los Sistemas Operativos	2
1.3.1. Primera generación (1945-1955)	3
1.3.2. Segunda generación (1955-1965)	3
1.3.3. Tercera generación (1965-1980)	3
1.3.4. Cuarta generación (1980-1990)	3
1.4. Conceptos de Sistemas Operativos	4
1.4.1. Procesos	4
1.4.2. Llamadas al sistema	5
1.4.3. Archivos	5
1.5. Componentes de un Sistema Operativo	5
1.5.1. Administrador de Procesos	6
1.5.2. Administrador de E/S	7
1.5.3. Administrador de la Memoria	7
1.5.4. Manejo del Sistema de Archivos	7
1.6. Breve historia de UNIX	7
1.6.1. UNIX actuales.	8

2. Introducción a GNU/Linux	9
2.1. ¿Qué es GNU/Linux?	9
2.2. Características	9
2.3. Historia de GNU/Linux	10
2.4. Objetivos de GNU/Linux	11
2.5. Filosofía del sistema GNU/Linux	12
2.6. Distribuciones	12
2.6.1. Slackware	13
2.6.2. Debian	13
2.6.3. SuSE	13
2.6.4. Mandriva	13
2.6.5. Red Hat	14
2.6.6. Fedora Core	14
2.6.7. CentOS	14
3. Preparandonos para la instalación	15
3.1. Introducción	15
3.2. Requerimientos mínimos de Hardware	15
3.2.1. Unidad Central de Proceso	15
3.2.2. Memoria	16
3.2.3. Disco Duro	16
3.2.4. BIOS	17
3.3. Revisión Final	17
4. Instalación de CentOS	19
4.1. Introducción	19
4.2. Acerca de las consolas virtuales	20
4.3. Inicio del programa de instalación	20
4.3.1. Método de instalación	21
4.4. Verificar la integridad de los cd's	21

4.5. Bienvenido a CentOS	22
4.6. Selección del idioma	22
4.7. Configuración del teclado	22
4.8. Tipo de instalación	24
4.8.1. Tipo de instalación Personalizada	25
4.9. Particionando el Disco Duro	25
4.9.1. Particionamiento automático	26
4.9.2. Partionamiento manual	26
4.9.3. Esquema de particionamiento recomendado	28
4.9.4. Tipos de sistemas de archivos	29
4.10. Configuración del gestor de arranque	29
4.11. Configuración de la red	29
4.12. Configuración del firewall	31
4.13. Selección del soporte del idioma	32
4.14. Configuración del huso horario	32
4.15. Configuración de la contraseña de root	32
4.16. Selección de grupos de paquetes	34
4.17. Listo para la instalación	35
4.18. Instalación de paquetes	35
4.19. Fin de la instalación	37
5. Introducción al shell de GNU/Linux	39
5.1. Primeros pasos...	39
5.1.1. Entrar en sesión	39
5.1.2. Cuentas, grupos y passwords	41
5.1.3. ¿Cómo ejecutar comandos?	42
5.1.4. Directorios	42
5.1.5. Listado de archivos	42
5.1.6. Manual de UNIX	44
5.1.7. Contenido de un archivo	44

5.1.8. Crear y borrar un archivo	44
5.1.9. Usuarios en el sistema	46
5.1.10. Write	46
5.1.11. Talk	46
5.1.12. Cambio de contraseña	47
5.1.13. Fin de Sesión	47
5.2. El Sistema X Window	47
5.2.1. Entornos de escritorio y gestores de ventanas	48
5.2.2. GNOME	49
5.3. Interprete de comandos: Shell	50
5.4. Estructura de Comandos	51
5.5. Expansiones de la línea de comandos	52
5.6. Variables de ambiente	52
5.7. Entrecorillado de argumentos	53
5.8. PS1	54
5.9. Entrada estándar y salida estándar.	54
5.10. Fin de flujo.	55
5.11. Error estándar.	56
5.12. Interconexión de comandos (entubamiento).	56
5.13. Filtros.	57
5.14. Campos y delimitadores.	57
5.15. Valores de retorno de los comandos.	58
5.16. El operador grave.	59
5.17. Secuencias de comandos.	59
5.18. Redirección del shell.	59
5.19. Comandos en <i>background</i>	60
5.20. Variables de entorno	61
5.21. Alias	62
5.22. Reutilización de comandos	62

5.23. Archivos de <i>bash</i>	62
5.24. El Sistema de Archivos	63
5.24.1. Definiciones	63
5.24.2. El sistema de archivos en GNU/Linux	64
5.24.3. Implantación del sistema de archivos	64
5.24.4. Manejo del sistema de archivos	66
5.24.5. Redireccionamientos	70
5.24.6. Navegando por el sistema de archivos	71
5.24.7. Espacio en disco	77
5.24.8. Montaje	78
6. Editores	81
6.1. Editor de pantalla completa <i>vi</i>	81
6.2. Editor de línea <i>ed</i>	84
7. Introducción a la Administración de GNU/Linux	89
7.1. Objetivo del administrador de sistemas	89
7.2. ¿Porqué necesitan los sistemas ser administrados?	89
7.3. Tareas del Administrador del Sistema	90
7.4. Responsabilidad del administrador del sistema.	91
8. Administración de Usuarios	93
8.1. El archivo <i>/etc/passwd</i>	93
8.2. El archivo <i>/etc/shadow</i>	96
8.3. El archivo <i>/etc/group</i>	97
8.4. Creación de cuentas de usuarios	98
8.4.1. Edición de archivos.	98
8.4.2. Fijar contraseña inicial.	99
8.4.3. Crear directorio propio del usuario.	99
8.4.4. Copiar archivos de inicialización.	99
8.4.5. Editar <i>/etc/group</i>	101

8.4.6. Tabla de usuarios y guía telefónica.	101
8.4.7. Fijar cuotas.	101
8.4.8. Verificar la nueva cuenta.	101
8.5. Baja de usuarios	102
8.6. Inhabilitar cuentas	102
8.7. Caducidad de contraseñas.	102
8.8. Pseudo logins.	103
8.9. Herramientas para la administración de usuarios.	103
8.10. Herramientas para la administración de grupos	103
9. Inicio y Baja del sistema	105
9.1. Bootstrapping	105
9.2. Arranque automático y arranque manual.	105
9.2.1. Pasos del proceso de arranque.	106
9.2.2. Inicialización del núcleo (kernel).	106
9.2.3. Configuración del hardware.	106
9.2.4. Procesos del sistema.	107
9.2.5. Intervención del operador (solo en arranque manual).	107
9.2.6. Scripts de arranque (scripts rc).	108
9.2.7. Operación en multiusuario.	108
9.3. Scripts de inicialización de Linux	108
9.4. Scripts rc estilo System V.	109
9.5. Problemas de arranque.	111
9.5.1. Problemas de hardware.	111
9.5.2. Problemas con el gestor de arranque del sistema operativo.	112
9.5.3. Problemas en el sistema de archivos.	112
9.5.4. Problemas de configuración del núcleo (kernel).	112
9.5.5. Errores en los scripts de arranque.	113
9.6. Baja del sistema	113
9.6.1. ¿Cuándo apagar un equipo?	114

9.6.2. shutdown	114
9.6.3. halt	114
9.6.4. reboot	115
9.6.5. Cambiar nivel de ejecución (System V).	115
10. Administración de Software	117
10.1. Administración de paquetes con RPM	117
10.1.1. Instalación de Paquetes	117
10.1.2. Desinstalación de paquetes	119
10.1.3. Actualización de paquetes	119
10.1.4. Consulta de paquetes	120
10.1.5. Verificación de paquetes	121
10.2. Administración de paquetes con yum	122
10.2.1. Actualización del sistema con yum	122
10.2.2. Búsquedas de paquetes	122
10.2.3. Consulta de información	122
10.2.4. Instalación de paquetes	123
10.2.5. Desinstalación de paquetes	123
10.2.6. Listado de paquetes	123
10.2.7. Limpieza del sistema	123
11. Administración básica de los servicios de red	125
11.1. xinetd	125
11.1.1. Archivos de configuración xinetd	125
12. Resaldos	127
12.1. Introducción	127
12.2. Dispositivos y Medios	127
12.3. Régimen de respaldo incremental	129
12.3.1. Respaldo de un sistema de archivos	129
12.3.2. Niveles de un respaldo incremental	129

12.3.3. Respaldo en Solaris 8	130
12.3.4. Esquemas de respaldo	131
12.3.5. Elección de un esquema de respaldo	132
12.4. Recomendaciones generales	132
12.5. Recuperación	134
12.5.1. Restaurar archivos	134
12.6. Restaurar sistemas de archivos	136
12.6.1. Recuperación de datos en Solaris 8	137
12.7. Otros comandos de respaldo	137
12.8. Amanda	138
13. Monitoreo del Sistema	141
13.1. Monitoreo de recursos	141
13.1.1. free	141
13.1.2. top	142
13.1.3. vmstat	143
13.1.4. Sysstat	144
13.1.5. iostat	144
13.1.6. mpstat	145
A. Resumen de comandos básicos en Unix	147
A.1. Comandos de Linux	147

Índice de figuras

1.1. Componentes de un sistema de cómputo	1
1.2. Estados de un proceso	4
1.3. Componentes típicos del kernel de GNU/Linux	5
4.1. Pantalla de inicio de instalación	21
4.2. Verificación de la integridad del conjunto de cd's	22
4.3. Pantalla de bienvenida	23
4.4. Selección del idioma	23
4.5. Configuración del teclado	24
4.6. Tipo de instalación	25
4.7. Configuración del particionamiento del disco	26
4.8. Particionamiento automático	27
4.9. Particionamiento con Disk Druid en los sistemas x86, AMD64 y Intel® EM64T	27
4.10. Configuración del gestor de arranque	30
4.11. Configuración de la red	30
4.12. Configuración del firewall	31
4.13. Selección del soporte del idioma	33
4.14. Configuración del huso horario	33
4.15. Contraseña de root	34
4.16. Selección de grupos de paquetes	35
4.17. Listo para la instalación	36
4.18. Instalación de paquetes	36

4.19. Instalación de paquetes	37
5.1. Gnome Desktop	50
5.2. Organización del disco en los sistemas.	66

Índice de cuadros

1.1. Unix Actuales	8
1.2. Variantes de Unix	8
2.1. La historia de GNU/Linux.	12
3.1. Espacio en disco mínimo requerido	16
3.2. Tamaño mínimo requerido por partición	17
3.3. Tabla de requerimientos del sistema	18
4.1. Consolas, combinaciones de teclas y contenidos	20
5.1. Interpretes de comandos en Linux/Unix	51
5.2. Algunos Filtros en línea de comandos Linux/Unix	57
5.3. Variables de entorno más usuales	61
5.4. Archivos de <i>bash</i>	63
5.5. Estructra de directorios en GNU/Linux	65
5.6. Accesos a un archivo.	67
9.1. Procesos espontáneos en BSD	107
9.2. Procesos espontáneas en System V	107
9.3. Procesos espontaneos en Linux	107
A.1. Comandos Linux/Unix de manipulación de archivos y directorios	147
A.2. Comandos Linux/Unix más frecuentes	148
A.3. Equivalencia de comandos Linux/Unix y DOS	148

Capítulo 1

Teoría de Sistemas Operativos

1.1. Introducción

Una computadora sin software es un montón de fierros que no nos es de gran ayuda, necesita de programas para solucionar nuestros problemas. Estos programas pueden clasificarse en dos grandes grupos:

- *programas de sistema* (software de sistema): controlan la operación de la computadora. Los más representativos son los compiladores y el sistema operativo.
- *programas de aplicación* (software de aplicación): resuelven problemas para los usuarios. Algunos ejemplos pueden ser: procesadores de palabras, hojas de cálculo, manejadores de bases de datos, juegos, etc.

Como ya se mencionó el sistema operativo es el programa fundamental entre los programas de sistema; controla todos los recursos de la computadora y proporciona la base sobre la que pueden escribirse los programas de aplicación. Es un nivel de software por encima del hardware que controla todas las partes del sistema y presenta al usuario una interfaz o **máquina virtual**. En un sistema de cómputo se reconocen los siguientes niveles, analizando estos niveles desde el usuario hacia la máquina física encontramos: (ver figura 1.1).

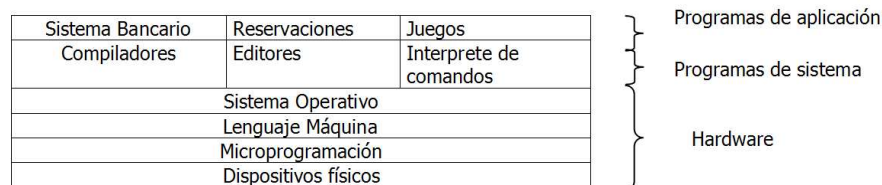


Figura 1.1: Componentes de un sistema de cómputo

- *Programas de aplicación*: Sistema bancario, Reservaciones aéreas, Juegos.
- *Programas de sistema*: Compiladores, Editores, Intérprete de comandos.
- *Sistema operativo*: Descrito en la sección 1.2
- *Lenguaje Máquina*: Conjunto de instrucciones que interpreta el microprograma.(50 a 300 instrucciones las cuales sirven para desplazar datos a través de la máquina, hacer operaciones aritméticas y comparar valores).
- *Microprogramación*: Software primitivo que controla en forma directa los dispositivos y proporciona una interfaz más amplia con la siguiente capa.
- *Dispositivos físicos*: son chips, cables, fuente de poder tubos de rayos catódicos y otros dispositivos físicos similares.
- *Arquitectura*: conjunto de instrucciones, organización de la memoria, E/S y estructura del bus.

1.2. ¿Qué es un Sistema Operativo?

Antes de iniciar describiendo que es GNU/Linux, sus ventajas, principales características, etc., necesitamos saber que es un sistema operativo. Este puede ser definido de acuerdo a la función que realiza, ya sea como *administrador de recursos*(sección 1.2.2) o como una *máquina virtual*(sección 1.2.1).

1.2.1. El sistema operativo como una maquina extendida.

Desde este punto de vista el sistema operativo oculta la verdad acerca del hardware al programador y le presenta una agradable y sencilla visión; esta abstracción que se le presenta al usuario es mucho más fácil de utilizar que el hardware subyacente, en otras palabras: "*Presentar al usuario el equivalente de una máquina extendida o máquina virtual que sea más fácil de programar que el Hw subyacente.*"

1.2.2. El sistema operativo como controlador de recursos.

Desde otro enfoque la función del sistema operativo es la de proporcionar una asignación ordenada y controlada de los recursos para los varios programas que compiten con ellos. Esto lo logra llevando un registro de la utilización de los recursos, dando paso a las solicitudes de recursos, llevando cuenta de su uso y mediar entre las solicitudes en conflicto de los distintos programas y usuarios.

1.3. Historia de los Sistemas Operativos

Al igual que las generaciones de computadoras¹ los sistemas operativos han tenido una constante evolución desde sus inicios. Este burdo resumen intenta describir esta corta pero importante evolución. Corta en el hecho de que apenas ha transcurrido medio siglo desde el incipiente comienzo.

¹La historia de la computación típicamente se divide en cinco generaciones marcadas por avances significativos en el hardware.

1.3.1. Primera generación (1945-1955)

- bulbos y conexiones.
- Programación en lenguaje de máquina absoluto, o realizando directamente las conexiones eléctricas.
- Alrededor de 1950 se introducen las tarjetas perforadas.

1.3.2. Segunda generación (1955-1965)

- transistores
- sistemas de procesamiento por lotes.
- FMS(Fortran Monitor System), IBSYS
- Los programas y datos se entregaban en tarjetas, se acumulaban y luego eran procesados todos juntos por la máquina, buscando minimizar los tiempos muertos.

1.3.3. Tercera generación (1965-1980)

- Circuitos integrados
- El sistema 360 de IBM unifica computadoras comerciales y científicas en una sola línea de máquinas con software compatible.
- Se introduce la multiprogramación, que divide la memoria en partes y ejecuta un programa distinto en cada una.
- El *spooling* permite la operación simultánea y en línea de periféricos.
- El *tiempo compartido* (Timesharing) es una variante de multiprogramación que habilita a cada usuario una terminal en línea.
- MULTICS (MULTiplexed Informartion and Computing Service) , un gigantesco sistema operativo, fracasa en su construcción pero aporta muchas ideas que hacen surgir UNIX.
- Ken Thompson desarrolla UNIX en una PDP-7.

1.3.4. Cuarta generación (1980-1990)

- estaciones de trabajo y computadoras personales.
- Sistemas operativos DOS y UNIX.
- Software "amigable con el usuario".
- Sistemas operativos de red, con varias computadoras interconectadas que pueden ser accedidas por un mismo usuario.
- Sistemas operativos distribuidos, compuestos por varios procesadores que se presentan al usuario como un sistema único.

1.4. Conceptos de Sistemas Operativos

Cuando hablamos de sistemas operativos tenemos que tener bien en claro los siguientes conceptos, los cuales son la base de cualquier sistema operativo:

- Procesos
- Llamadas al sistema
- Archivos

1.4.1. Procesos

Todo el software ejecutable de la computadora se organiza en varios procesos. *Un proceso es tan solo un programa en ejecución*, consta del programa ejecutable, sus datos y pila. Los estados en los que un proceso se puede encontrar son:

- En ejecución
- Listo (no existe cpu disponible para el)
- Bloqueo

Las transiciones entre un estado y otro se detallan en la figura 1.2.

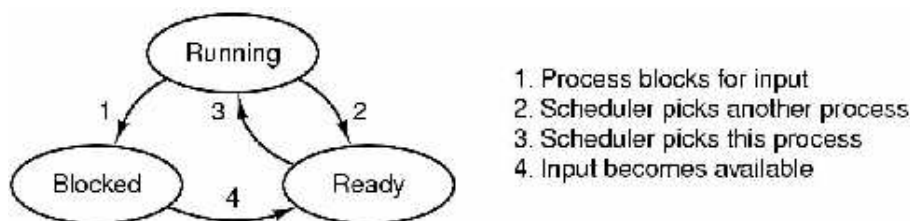


Figura 1.2: Estados de un proceso

Comunicación entre procesos

Los procesos durante su ejecución, rara vez hacen las cosas solos, necesitan comunicarse con otros procesos, alguno amigables, otros no tanto. Cuando hay problemas de comunicación entre ellos se generan condiciones de competencia.

Condición de competencia

Es cuando dos o más procesos leen o escriben en ciertos datos compartidos y el resultado final depende de quién ejecuta qué y en qué momento.

1.4.2. Llamadas al sistema

1.4.3. Archivos

1.5. Componentes de un Sistema Operativo

Un sistema operativo está formado por varios programas que en conjunto presentan al usuario una vista integrada del sistema, los componentes principales de un sistema operativo son los siguientes módulos:

- Administrador de procesos. (*scheduler*)
- Administrador de E/S.
- Administrador de la Memoria.
- Manejo del Sistema de Archivos.

La figura 1.3 muestra los componentes típicos del kernel de GNU/Linux.

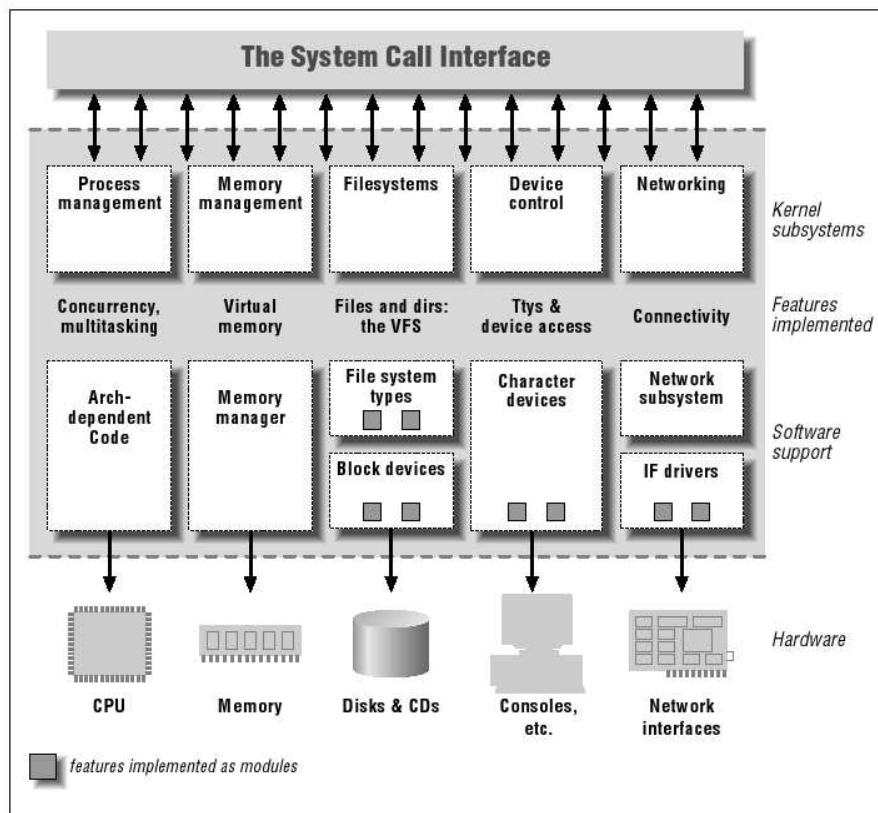


Figura 1.3: Componentes típicos del kernel de GNU/Linux

1.5.1. Administrador de Procesos

El planificador de procesos es el proceso² encargado de decidir cual proceso se ejecuta primero. En otras palabras "Es la parte del sistema operativo que decide que proceso se ejecuta en que momento, y por cuanto tiempo (quantum)."

Objetivos

- *Equidad*: garantizar que el proceso obtiene su proporción justa de CPU.
- *Eficacia*: mantener ocupado el CPU el 100
- *Tiempo de Respuesta*: minimizar el tiempo de respuesta para los usuarios interactivos.
- *Tiempo de regreso*: minimizar el tiempo que deben esperar los usuarios por lotes para obtener resultados.
- *Rendimiento*: maximizar el número de tareas procesadas por hora.

Tipos de estrategias

Para lograr sus objetivos, el planificador de procesos cuenta con dos tipos de estrategias:

- *Planificación apropiativa*: permite que procesos ejecutables seas suspendidos en forma temporal.
- *Planificación no apropiativa*: ejecución hasta terminar (primeros sistemas operativos por lotes)

Algoritmos de planificación de procesos

- *Round robin* todos los procesos tienen la misma prioridad. Cada proceso tiene asignado un intervalo de tiempo de ejecución (quantum). Si el proceso continúa en ejecución al final de su quantum, otro proceso se apropia de la CPU. Si el proceso está bloqueado o ha terminado antes de consumir su quantum, se alterna el uso de la CPU. La alternancia entre un proceso y otro necesita cierta cantidad de tiempo para administración (resguardo y carga de registros y mapas de memoria, actualización de varias tablas y listas, etc). Si el quantum es muy corto, se alternan demasiados procesos, lo que reduce la eficacia de la CPU; pero si es muy largo puede causar una respuesta lenta a las solicitudes interactivas breves.
- *Por prioridad* hay dos clases de prioridades: *estática* o *dinámica*. Si las prioridades no se ajustan de vez en cuando, las clases de prioridad mínima podrían morir de inanición eterna.
- *Colas múltiples*.
- *Primero el trabajo mas corto*.

²también conocido como scheduler

1.5.2. Administrador de E/S

1.5.3. Administrador de la Memoria

1.5.4. Manejo del Sistema de Archivos

1.6. Breve historia de UNIX

Ahora que ya conocemos la historia de los sistemas operativos, podemos ubicar a UNIX en un contexto histórico. Pero, ¿qué cosa es UNIX y porque es tan importante a la hora de hablar de GNU/Linux? Estamos por descubrirlo.

El sistema operativo UNIX comienza en 1969 como un proyecto de investigación de AT&T Bell Labs. Es descendiente de *MULTICS*, un proyecto de sistema operativo sumamente ambicioso iniciado por un conjunto de empresas que nunca vió la luz. En 1976 aparece la Versión 6 (V6), disponible en forma gratuita para las Universidades. La versión 7, ampliamente distribuida, aparece en 1979; es la base de la mayoría de las versiones de UNIX. Costaba \$100 dls. para universidades y \$21000 dls. para otros. Luego de esta versión AT&T creó el grupo USG (UNIX Support Group), luego este grupo se separó de AT&T con el nombre USL (UNIX System Laboratories) para desarrollar UNIX como producto comercial. Los grupos Bell Labs y USG continuaron desarrollando UNIX en direcciones divergentes. USL sacó al mercado las versiones System III y System V, ampliamente difundidas. En 1992 AT&T vendió sus operaciones en UNIX a Novell quien sacó un producto llamado UnixWare, el cual no resultó muy popular.

El UNIX de Berkeley comenzó en 1977, cuando el CSRG (*Computer Systems Research Group*) de la Universidad de California en Berkeley licenció el código de la versión V6 de AT&T. Las versiones de Berkeley llamadas BSD (*Berkeley Software Distribution*) comenzaron en 1977 con 1BSD (para PDP-11) y culminaron en 1993 con la versión 4.4BSD. El propósito último de Berkeley era eliminar todo el código original de AT&T, pero perdió sus fuentes de financiamiento antes de culminar su propósito. Al disolverse, el CSRG lanzó su última versión de código propio, llamada 4.4BSD-Lite (liviana). La mayoría de los UNIX de la rama BSD tienen como antecedente esta versión.

Superadas varias querellas de licencias entre BSD y AT&T, desde hace unos años los fabricantes pueden tomar el código fuente público de AT&T, BSD o ambos para comenzar a partir de ellos sus propios desarrollos. Esto ha originado un sin número de variantes. La clasificación por el origen, AT&T o BSD, se va perdiendo al combinar los fabricantes lo mejor de ambas. Si bien esto ha originado bastante confusión, en el correr de los años la variedad de líneas de investigación ha permitido decantar soluciones eficientes, robustas y elegantes, a más de asegurar la permanencia del sistema operativo sin riesgo alguno de monopolio o propiedad.

El desarrollo más reciente ha sido el advenimiento de Linux, a partir de un núcleo (kernel) diseñado por Linus Torvalds, un estudiante graduado de la Universidad de Helsinki, Finlandia, en 1991. A este núcleo se agregan una cantidad de productos complementarios, fundamentalmente de GNU, un proyecto de desarrollo de UNIX libre de costo. Sostenido por una extensa comunidad de empresas, desarrolladores, voluntarios y entusiastas, Linux es hoy un sistema operativo completo, de calidad profesional, usado en producción y soportado como sistema operativo primario por muchos proveedores. Muchas empresas grandes como Oracle han portado sus productos a Linux.

1.6.1. UNIX actuales.

En cuanto a popularidad, las variantes de UNIX más difundidas al día de hoy son las mostradas en la tabla 1.1, sin un orden predeterminado.

Producto	Fabricante	Características
Solaris	Sun Microsystems	Basado en AT&T, con muchas extensiones. Arquitectura Sparc y x86.
HP-UX	Hewlett-Packard	Híbrido AT&T y BSD, con particularidades propias. Arquitectura propietaria
Linux	Público	BSD (SunOS) en lo interno, AT&T en la administración. Arquitectura Intel x86, sparc, alpha, y otras. Existen múltiples distribuciones; entre las más conocidas pueden citarse Red-Hat, S.u.s.e., Slackware, Debian, Corel, Caldera, Mandrake. FreeBSD Público Basado en BSD. Arquitectura Intel x86.

Cuadro 1.1: Unix Actuales

Otras variantes con difusión en el mercado son las mostradas en la tabla 1.2:

Producto	Fabricante	Características
SunOS (Solaris 1)	Sun Microsystems	Basado en BSD. Arquitectura Sparc.
DEC OSF/1	Digital Research	Basado en BSD e intento de standard Open Software Foundation. Actualmente DEC abandonó el OSF pues no resultó un standard y comercializa Digital UNIX (DU). Arquitectura Alpha.
AIX	IBM	Es bastante diferente de ATT y BSD especialmente en las herramientas administrativas. Arquitectura RS6000.
IRIX	Silicon Graphics	Similar a AT&T. Arquitectura propietaria.
SCO	The Santa Cruz Operation	Basado en AT&T pero con muchos agregados. Arquitectura Intel x86.
OpenBSD	Público	Basado en BSD. Arquitectura Intel x86 y plataformas adicionales (alpha, sparc, hp, amiga)

Cuadro 1.2: Variantes de Unix

Capítulo 2

Introducción a GNU/Linux

2.1. ¿Qué es GNU/Linux?

Ahora que ya conocemos la historia de Unix, estamos preparados para definir a GNU/Linux.

GNU/Linux es un *clon* del sistema operativo Unix, escrito desde cero por el finlandés *Linus Torvalds* con la asistencia de un pequeño grupo de hackers esparcidos por la red¹. Entre sus principales características se encuentra la implantación del estandar POSIX.

Estrictamente, GNU/Linux se refiere al *núcleo* o kernel. En un sentido más amplio, comprende el núcleo del sistema operativo más un conjunto de programas que permiten compilar lenguajes de programación, editar texto, interpretar comandos, manejar archivos y discos, acceder a otras máquinas, establecer comunicaciones telefónicas, enviar y recibir correo electrónico, manejar las colas de impresión y un sinnúmero de tareas más. Algunos de estos programas pueden haber sido desarrollados por los propios usuarios.

Cuando hablemos de GNU/Linux nos estaremos refiriendo al sentido amplio.

2.2. Características

GNU/Linux posee las siguientes características:

- *portable*: el mismo sistema operativo corre en un espectro de máquinas que van desde notebooks a supercomputadoras. Es el único sistema operativo con estas características.
- *flexible*: se adapta a muchas aplicaciones diferentes.
- *potente*: dispone de muchos comandos y servicios ya incorporados.
- *multiusuario*: atiende a muchas personas simultáneamente.

¹Así es como lo dice la página oficial del desarrollo del kernel <http://www.kernel.org/>, sin embargo algunos pensamos que se fusiló código de MINIX, sistema operativo desarrollado por el profesor Andrew Tannenbaum.

- *multitarea*: hace muchas cosas a la vez.
- *elegante*: sus comandos son breves, coherentes, específicos para cada tarea y muy eficientes.
- *orientado a redes* desde el comienzo.
- Dispone de un estándar (*POSIX*) que debe cumplir todo sistema operativo que pretenda ser Unix, lo que asegura una evolución predecible y compatibilidad con otros Unix.

2.3. Historia de GNU/Linux

Lo que en un principio no era más que un proyecto personal de un joven que se creía el mejor programador del mundo², terminó siendo uno de los mejores sistemas operativos; usado ampliamente en todo el mundo, desde instituciones educativas hasta comerciales, pasando por gubernamentales.

Fué en Julio de 1991 cuando Linux aún siendo estudiante de Computer Science en Finlandia, envió su primer mensaje al grupo de noticias `comp.os.minix`, respecto a un proyecto personal sobre el sistema operativo Minix³.

Este es el primer mensaje:

```
From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroup: comp.os.minix
Subject: GCC-1.40 and a posix question
Message-ID: 1991Jul13, 100050.9886@klaava.Helsinki.FI
Date: 3 Jul 91 10:00:50 GMT
```

```
Hello netlanders,
Due a project I'm working on (in minix), I'm interested in the posix standard
definition. Could somebody please point me to a (preferably) machine-readable
format of the latest posix rules? Ftp-sites would be nice.
....
Linux Torvalds torvalds@kruuna.helsinki.fi
```

Al que le siguió este mensaje, que muchos consideran el verdadero inicio del Linux.

```
From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroup: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: 1991 Aug 25, 20578.9541@klaava.Helsinki.FI
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki.
```

²A sus 21 años

³Minix es un clon del sistema operativo Unix distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987. La última versión oficial de Minix es la 3.0 y data de octubre del 2005.

Hello everybody out there using minix- I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix; as my OS resembles it somewhat (same physical layout of the file-sytem due to practical reasons) among other things.

I've currently ported bash (1.08) an gcc (1.40), and things seem to work. This implies that i'll get something practical within a few months, and I'd like to know what features most people want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linux Torvalds torvalds@kruuna.helsinki.fi

Respecto al inicio, Linux escribiría años después lo siguiente:

"Estaba finalizando de escribir mi tesis - Fue el unico credito que en tenido en mi vida academica, cuando escribi Linux. Yo tenia 21 anos, y la verdad no tenia idea de lo que estaba haciendo. Me creia el mejor programador del mundo, todo programador de 21 años lo cree, .. Me dije: Cuan difícil puede ser, es solo un sistema operativo..?"

Las versiones iniciales fueron distribuidas en código fuente por el propio Linus, para que otras personas puedan entender su proyecto y sobretodo para que lo ayudaran.

Linus trabajó activamente hasta la versión 0.96, pues tras ello, se sumaron al proyecto más programadores y se formó un grupo de desarrollo amplio (Linux Developers) que continúa siendo dirigido por él; pero como el mismo lo reconoce, su labor es más la de un "router"⁴ del grupo que la del desarrollo en sí.

La historia cronológica se encuentra resumida en la tabla 2.1.

2.4. Objetivos de GNU/Linux

GNU/Linux fue diseñado teniendo en mente los siguientes objetivos::

- crear un sistema interactivo de tiempo compartido diseñado por programadores y para programadores, destinado a usuarios calificados.
- que fuera *sencillo, elegante, escueto y consistente*.
- que permitiera resolver problemas complejos combinando un número reducido de comandos básicos.

⁴Como la de todo buen lider, tu has esto tu lo otro

Versión	Año	Usuarios Estimados	Tamaño del kernel (KBytes)	Hitos
0.01	1991	100	63	Linus Torvalds escribe el kernel de Linux.
0.99	1992	1000	431	Software GNU es integrado al kernel dando como resultado un sistema totalmente funcional.
0.99	1993	20,000	938	Dada la cantidad de contribuciones al código, Linus se ve obligado a delegar la responsabilidad de revisión del kernel.
1.0	1994	100,000	1,017	Se libera la primera versión estable.
1.2	1995	500,000	1,850	Compatibilidad con procesadores non-intel.
2.0	1996	1,500,000	4,718	Linux soporta multiples procesadores, IP masquerading y Java.
2.2	1999	7,500,000	10,593	El crecimiento de Linux excede a Windows NT.
2.4	2001	10,000,000	19,789	Linux invade las empresas privadas.
2.6	2003	20-50,000,000	32,476	Linux mejora para cumplir con requerimientos de cualquier sistema operativo comercial, como la mejora en el manejo de hilos (<i>threads</i>).

Cuadro 2.1: La historia de GNU/Linux.

2.5. Filosofía del sistema GNU/Linux

Los objetivos con que se creó determinaron una “filosofía” caracterizada por:

- comandos cortos, simples, específicos y muy eficientes, que “hacen una sola cosa pero la hacen muy bien”.
- entrada y salida estandarizadas que permiten la interconexión de comandos. Esto se llama entubamiento (“pipelining”): la salida de un comando es tomada por el siguiente como entrada.
- todo es un *archivo*.

2.6. Distribuciones

Como ya mencionamos, GNU/Linux es sólo el núcleo del sistema operativo, pero necesita aplicaciones y programas para *hacer algo*. Muchos han sido portados a Linux, otros han sido creados específicamente para GNU/Linux, todos ellos se encuentran en Internet dispuestos a que cualquiera los baje y los instale en su sistema.

Como esto es una ardua tarea no tardan en surgir compañías dedicadas a reunir todos esos programas facilitando la tarea de crear un sistema Linux funcional. En la actualidad existe un sinnúmero de estas compañías pero solo se mencionarán las mas importantes dentro del contexto mundial.

2.6.1. Slackware

Una de las primeras distribuciones que surge fué *Slackware* (<http://www.slackware.com>), diseñada por Patrick Volkerding a partir de *SLS Linux*. Esta tuvo una gran aceptación al principio hasta llegar a ser la distribución mas popular del mercado. Actualmente ha perdido terreno a favor de distribuciones mas modernas, siendo relegada a aplicaciones especializadas.

Una de las debilidades de *Slackware* se encuentra en el sistema de instalación de paquetes, el cual no tiene control de versiones ni dependencias. Las nuevas distribuciones han implementado y adoptado dos sistemas de instalación de archivos RPM (RedHat Package Manager) y DEB (Debian Package Manager). Cada programa distribuido de esta forma es un archivo comprimido, que se identifica por la extensión *rpm* o *deb* y proporciona una descripción de lo que contiene, la versión del programa, su ubicación en el sistema, validez de la firma electrónica y las dependencias con otros programas o bibliotecas; por ejemplo, un determinado paquete puede necesitar de otro para su correcto funcionamiento, por lo que se dice que es *dependiente* del otro. De esta forma se garantiza el éxito del proceso de instalación de una aplicación y la estabilidad a largo plazo del sistema.

2.6.2. Debian

Debian (<http://www.debian.org>) es una distribución bastante popular que no está desarrollada por ninguna compañía comercial sino que es fruto del trabajo de diversos voluntarios en toda la comunidad de Internet. Es, por lo tanto, una distribución completamente gratis, sin restricción de licencias en donde todo el software es GNU/GPL⁵ y no incluye software comercial. Además es bastante completa y estable gracias a su sistema de instalación de paquetes DEB. Sin embargo, tal vez sea algo difícil para alguien que empieza por primera vez con Linux. Esto no quiere decir que si es la primera vez que se va a instalar Linux y se tiene una *Debian* a mano vaya a ser casi imposible instalarla. Es importante mencionar que *Debian* también ofrece una versión de su distribución basado en otro kernel diferente a Linux: GNU Hurd.

2.6.3. SuSE

SuSE (<http://www.suse.de>) es una distribución de una compañía alemana la cual combina el sistema de paquetes de Red Hat (RPM) con una organización derivada de Slackware. Esta distribución es la mas popular en Europa y tiene un gran soporte para diferentes lenguas incluido el Español. Es una de las más fáciles de instalar y configurar, además viene con una gran cantidad de paquetes.

2.6.4. Mandriva

Mandriva Linux(<http://www.mandriva.com/community/>) antes Mandrake Linux es una distribución Linux que hizo su aparición en julio de 1998 propiedad de Mandriva, enfocada a principiantes o usuarios medios.

Apoya también totalmente la licencia GNU GPL, ofrece públicamente su distribución en formato ISO, sus asistentes o sus repositorios.

⁵GNU Licencia publica General

2.6.5. Red Hat

Red Hat (<http://www.redhat.com>) es la distribución mas popular del mercado hoy en día, siendo emulada por muchas otras. Muy sencilla de instalar, excelente auto-detección de dispositivos, instalador gráfico y un excelente conjunto de aplicaciones comerciales en su distribución oficial.

Esta guía se basará principalmente en la instalación de Red Hat por ser la más difundida. Sin embargo, comprendiendo los conceptos del proceso de instalación de estas se pueden aplicar a cualquier otra distribución.

2.6.6. Fedora Core

2.6.7. CentOS

CentOS (acrónimo de Community ENTerprise Operating System) es un clon a nivel binario de la distribución Red Hat Enterprise Linux, compilado por voluntarios a partir del código fuente liberado por Red Hat, empresa desarrolladora de RHEL.

Esta distribución a comenzado a ganar terreno dentro de las más usadas a nivel mmundial. Excelente opción para aquellos que quieran la misma estabilidad de RHEL pero si el costo de la licencia.

Capítulo 3

Preparandonos para la instalación

3.1. Introducción

El proceso de instalación consiste en copiar los archivos del sistema operativo al disco duro de nuestro equipo. Este proceso puede hacerse de un conjunto (set) de cdrom o via red.

Los procedimientos aquí descritos, así como los requerimientos y restricciones aplican para la distribución de CentOS y para RedHat.

Antes de pensar en instalar cualquier distribución Linux, debemos conocer el hardware donde queremos instalarlo, esta información puede ser obtenida, ya sea de la especificaciones del fabricante o si el equipo ya cuenta con un Sistema Operativo instalado, usando las herramientas de dicho sistema operativo. Esta sección pretende presentar información que es necesaria para poder instalar CentOS.

3.2. Requerimientos mínimos de Hardware

CentOS soporta una gran variedad de hardware, sin embargo no todos los dispositivos están soportados. El equipo a instalar **debe** cumplir ciertos requerimientos mínimos. Una lista bastante extensa que nos puede ayudar a indentificar si nuestro Hardware es compatible y/o certificado esta disponible en la página <https://hardware.redhat.com/hwcert/index.cgi>. (¿Porque en la página de RedHat? porque centos es un clon de RedHat)

3.2.1. Unidad Central de Proceso

CentOS no soporta procesadores Intel i386 o inferiores. Soporta procesadores Intel i486, Celeron, Pentium, Pentium Pro, Pentium II, Pentium III, y Pentium IV y procesadores compatibles manufacturados por otros fabricantes, como el procesador AMD's Athlon, Athlon XP, and Athlon MP. Estos procesadores son miembros de la familia conocida como *x86*. CentOS también soporta procesadores

que no sean intel (*non-Intel* como AMD's AMD64, IBM's PowerPC y algunos procesadores usados en mainframes IBM. Tambien soporta los nuevos procesadores Intel Itanium.

Para poder usar CentOS en el *escritorio* la velocidad mínima del procesador debe ser de 400 MHz con un procesador Pentium II. En modo texto se requiere al menos un procesador de 200MHz.

Para cada tipo de procesador existe un *set* o conjunto de cd's distintos. A continuación se describen los diferentes set de cd's de instalación disponibles:

i386 Procesadores AMD (K6, K7, Thunderbird, Athlon, Athlon XP, Sempron), Pentium (Classic, Pro, II, III, 4, Celeron, M, Xeon), VIA (C3, Eden, Luke, C7).

x86.64 Procesadores AMD (Athlon 64, Opteron) and Intel Pentium (Xeon EM64T) processors.

ia64 Procesadores Intel Itanium2

s390 Procesadores IBM S/390

s390x Procesadores para servidores IBM Z-Series

alpha Procesador DEC Alpha

3.2.2. Memoria

Para un desempeño óptimo se necesita por lo menos 256 MB de RAM. Esta es la recomendación de RedHat, mas sin embargo la mayoría de la gente instala GNU/Linux en sistemas que van desde 192 hasta 256 MB de RAM. Se puede instalar con 64 MB de RAM pero sin ambiente gráfico(no GUI).

Gente muy especializada ha instalado GNU/Linux en equipos con 4 MB de RAM, usando sus propios metodos de instalación y configuración.

3.2.3. Disco Duro

La cantidad de espacio necesaria para la instalación varia de distribución a distribución, la tabla 3.1 muestra los requerimientos mínimos para la instalación de CentOS.

Tipo de Instalación	Espacio mínimo requerido
Personal Desktop	2.3 Gb
Workstation	3.0 Gb
Server	1.1-1.5 Gb(sin ambiente gráfico 6.9 gigabytes (todo incluido)
Custom	0.62-6.9 Gb

Cuadro 3.1: Espacio en disco mínimo requerido

CentOS como cualquier otro GNU/Linux necesita por lo menos **dos** particiones para trabajar, / (*root*), y *swap*. Aunque las recomendadas son: */boot*, */var*, */usr*, */tmp* y */home*.

La tabla 3.2 describe los tamaños mínimos recomendados para estas particiones.

Partición	Tamaño
/boot	100 megabytes
/(root)	500 megabytes
(swap)	Dos veces la cantidad de memoria RAM en sistemas con un mínimo de 500 megabytes
/home	Tan grande como sea necesario; depende del número de usuarios y el tipo de trabajo que realicen
/tmp	Mínimo de 500 megabytes
/usr	Mínimo de 1.7–5.5 gigabytes, dependiendo de la versión de CentOS y la cantidad de paquetes a instalar
/var	Mínimo de 500 megabytes

Cuadro 3.2: Tamaño mínimo requerido por partición

3.2.4. BIOS

Si se desea realizar la instalación desde la unidad de cdrom, se debe verificar que el bios de nuestro equipo este configurado para que el arranque (*boot*) sea desde la unidad de cdrom.

3.3. Revisión Final

Como parte de las buenas prácticas de administración, es recomendable tener documentado las opciones de instalación y configuración final de cada equipo o servidor.

La tabla 3.3 pretende ayudar a mantener un registro de sus configuraciones y requerimientos de sistema actuales.

Una vez que llenamos esta tabla, estamos listos para comenzar con la instalación de CentOS.

<i>disco(s) duros</i> : tipo, etiqueta, tamaño; ej: IDE hda=40 GB	
<i>particiones</i> : mapa de las particiones y sus puntos de montaje, ej: /dev/hda1=/home, /dev/hda2=/ (llene esta parte cuando sepa dónde se ubicarán)	
<i>memoria</i> : cantidad de memoria RAM instalada en su sistema, ej: 128 MB, 512 MB	
<i>CD-ROM</i> : tipo de interfaz; ej: SCSI, IDE (ATAPI)	
<i>adaptador SCSI</i> : si está presente, el fabricante y número de modelo; ej: BusLogic SCSI Adapter, Adaptec 2940UW	
<i>tarjeta de red</i> : si está presente, fabricante y número de modelo; ej: Tulip, 3COM 3C590	
<i>ratón</i> : tipo, protocolo y número de botones; ej: ratón genérico de 3 botones PS/2, ratón serial MouseMan de 2 botones	
<i>monitor</i> : fabricante, modelo y especificaciones del fabricante; ej: Optiquess Q53, ViewSonic G773	
<i>tarjeta de vídeo</i> : fabricante, número de modelo y tamaño de VRAM; ej: Creative Labs Graphics Blaster 3D, 8MB	
<i>tarjeta de sonido</i> : fabricante, chipset y número de modelo; ej: S3 SonicVibes, Sound Blaster 32/64 AWE	
<i>IP, DHCP y direcciones BOOTP</i> : cuatro números, separados por puntos; ej: 10.0.2.15	
<i>direcciones IP del gateway</i> : cuatro números, separados por puntos; ej: 10.0.2.245	
<i>una o más direcciones IP de servidores de nombres (DNS)</i> : uno o más conjuntos de números separados por puntos; ej: 10.0.2.1	
<i>nombre de dominio</i> : el nombre dado a su organización, ej: ejemplo.com	
<i>nombre de la máquina</i> : el nombre del equipo, su selección personal de nombres, ej: titan, homero	

Cuadro 3.3: Tabla de requerimientos del sistema

Capítulo 4

Instalación de CentOS

4.1. Introducción

CentOS (acrónimo de Community ENTerprise Operating System) es un clon a nivel binario de la distribución Red Hat Enterprise Linux, compilado por voluntarios a partir del código fuente liberado por Red Hat, empresa desarrolladora de CentOS.

El siguiente procedimiento detalla la instalación del sistema operativo CentOS, la cual es exactamente igual a la de CentOS (RedHat Enterprise Linux), desde un CDROM usando Anaconda. La instalación desde red es similar pero con sutiles diferencias.

Este procedimiento tambien puede aplicarse a los distintos clones de Red Hat Enterprise Linux.

El proceso de instalación se puede resumir en los siguientes pasos:

1. Insertar el primer CD en el equipo y reiniciarla.
2. Despues de reconocer el hardware del equipo, se despliega la pantalla inicial de instalación con el prompt **boot** en la parte inferior de la pantalla.
3. Presionar **ENTER** o dejar pasar uno segundo para que inicie la instalación.
4. Como parte del proceso de inicio, CentOS genera un disco RAM que es usado en lugar de un disco duro. Las herramientas usadas para la instalación, son copiadas al disco RAM. El uso de este disco RAM permite establecer una configuración sin borrar o dañar la información contenida en el disco duro. Una ves validada la configuración se incia la copia de archivos al disco duro.
5. Si se desea, se puede revisar la integridad de los CD de instalación.
6. Anaconda inicia y prueba el hardware para iniciar el modo gráfico.
7. Anaconda colecta información acerca de como se desea realizar la instalación.
8. Cuando el proceso de Anaconda termina de recolectar información, **advierte** sobre el riesgo de perder información del disco duro.

9. Cuando el sistema reinicia, el script de instalación realiza una serie de preguntas antes de completar la instalación.
10. CentOS está listo para usarse.

4.2. Acerca de las consolas virtuales

El programa de instalación de CentOS ofrece terminales adicionales a las ventanas de diálogo del proceso de instalación. Además de darle la posibilidad de insertar comandos desde el intérprete de comandos de la shell, le muestra muchos mensajes de diagnóstico. El programa de instalación despliega estos mensajes en cinco consolas virtuales, entre las que puede cambiarse usando una combinación de teclas.

Estas consolas virtuales pueden ayudarle en el caso de que encuentre problemas durante la fase de instalación de CentOS. Los mensajes visualizados durante la instalación o en las consolas del sistema, pueden ayudarle a señalar un problema. Consulte la tabla 4.1 para ver la lista de las consolas virtuales, las combinaciones de teclas para cambiarse entre ellas y sus contenidos.

En general, no hay ninguna razón para dejar la consola predeterminada (consola virtual #7 para las instalaciones gráficas) a no ser que tenga intenciones de detectar problemas de instalación.

Consola	Combinación de teclas	Contenido
1	Ctrl-Alt-F1	diálogo de instalación
2	Ctrl-Alt-F2	intérprete de comandos
3	Ctrl-Alt-F3	registro de instalación (mensajes del programa de instalación)
4	Ctrl-Alt-F4	mensajes del sistema
5	Ctrl-Alt-F5	otros mensajes
7	Ctrl-Alt-F7	pantalla gráfica de X

Cuadro 4.1: Consolas, combinaciones de teclas y contenidos

4.3. Inicio del programa de instalación

Para arrancar el programa de instalación, se puede usar cualquiera de los siguientes medios (en función del medio compatible con nuestro sistema):

- *CD-ROM* ¿ Si nuestro equipo soporta una unidad de CD-ROM de arranque y contamos con el conjunto de CD-ROMs de CentOS completo.
- *CD-ROM de arranque* ¿ Si nuestro equipo soporta una unidad de CD-ROM de arranque y queremos realizar una instalación de red o disco duro.
- *módulo de memoria USB* ¿ Nuestro equipo soporta el arranque desde un dispositivo USB.

En cualquier caso la figura 4.1 muestra la pantalla que veremos al iniciar desde cdrom o memoria usb.



Figura 4.1: Pantalla de inicio de instalación

4.3.1. Método de instalación

Están disponibles los siguientes métodos de instalación:

CD-ROM Se necesita una unidad de CD-ROM y los CD-ROMs CentOS.

Disco duro Si descargamos los isos's de la red y los copiamos a nuestro disco duro, los podemos utilizar para la instalación.

NFS Si estamos realizando la instalación desde un servidor NFS utilizando imágenes ISO o una copia, podemos utilizar este método. Aun así necesitamos un CD-ROM de arranque y usar la opción de arranque *linux askmethod*.

FTP Si estamos realizando la instalación directamente desde un servidor FTP, utilizamos este método. Aun así necesitamos un CD-ROM de arranque y usar la opción de arranque *linux askmethod*.

HTTP Si estamos realizando la instalación directamente desde un servidor Web HTTP, utilizamos este método. Aun así necesitamos un CD-ROM de arranque y usar la opción de arranque *linux askmethod*.

4.4. Verificar la integridad de los cd's

En caso que hayamos elegido la opción de instalación via los set's de cd's, el primer paso es verificar la integridad de los cd's(figura 4.2).Esto es únicamente necesario cuando nos queremos asegurar que

nuestros discos funcionan sin problema. Este proceso puede tardar hasta 40 minutos. Para elegir una opción apretamos tabulador y luego enter.

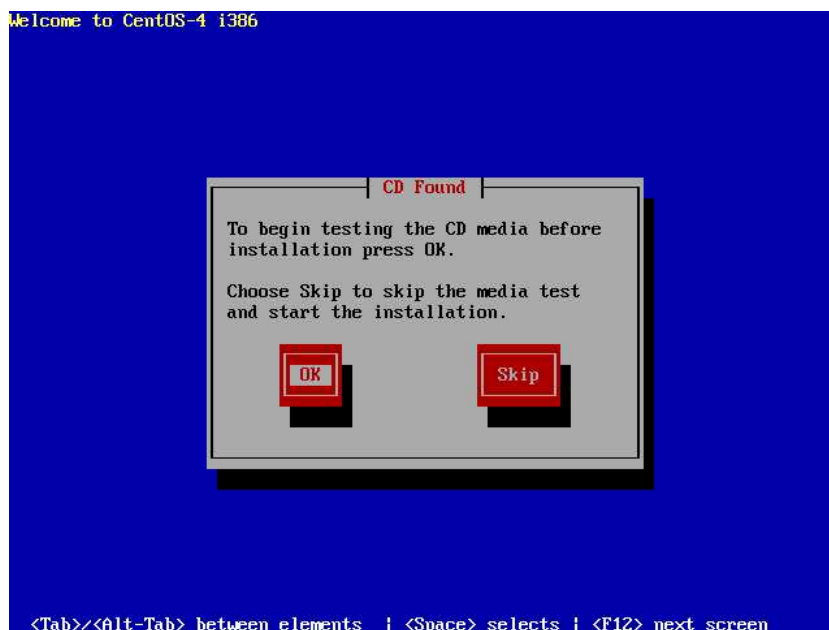


Figura 4.2: Verficiación de la integridad del conjunto de cd's

4.5. Bienvenido a CentOS

El instalar anaconda prueba nuestra tarjeta de video y despues de reconocerla, y comprobar que el ambiente gráfico puede iniciar sin problemas, la pantalla de Bienvenida es desplegada (figure 4.3); no pide ninguna información, solo damos click en el botón Siguiente para continuar.

4.6. Selección del idioma

Utilizando el ratón, seleccionamos el idioma a utilizar durante el proceso de instalación. (figura 4.4)

4.7. Configuración del teclado

Con el ratón, seleccionamos el tipo de configuración correcta para el teclado (por ejemplo, U.S. English) que se quiera usar durante el proceso de instalación y como el predeterminado para el sistema. (figura 4.5)



Figura 4.3: Pantalla de bienvenida

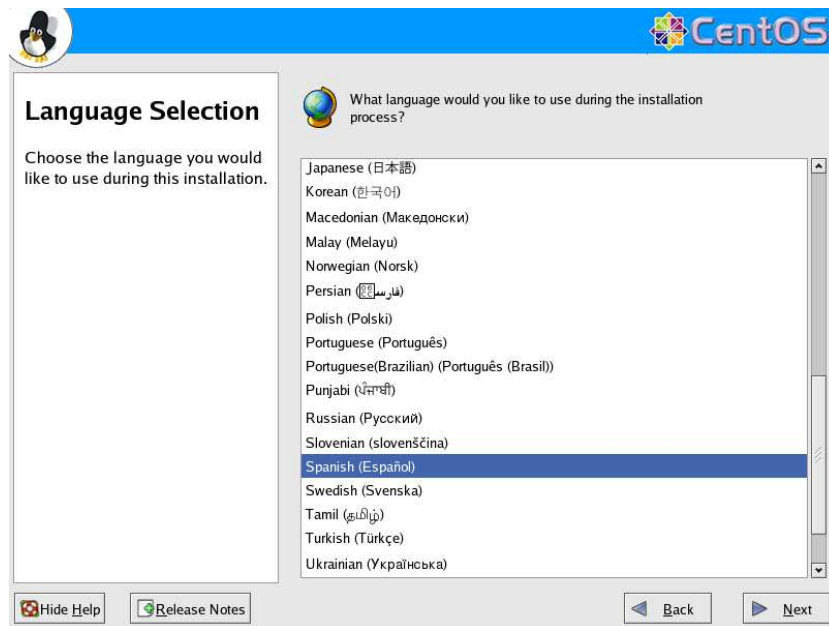


Figura 4.4: Selección del idioma

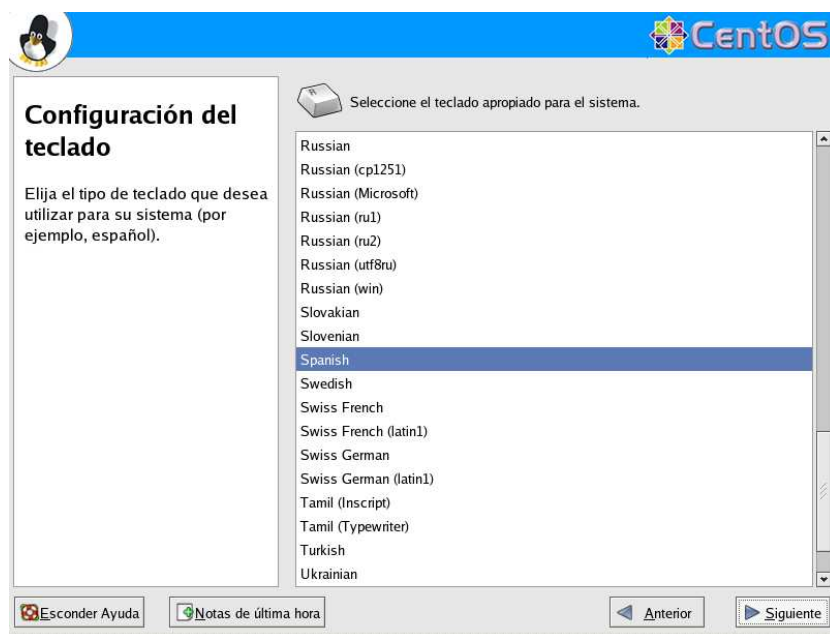


Figura 4.5: Configuración del teclado

Para cambiar la configuración del teclado después de haber completado la instalación, se usa el comando `system-config-keyboard`. Si no es root¹, se pedirá que introduzca la contraseña de root para continuar.

4.8. Tipo de instalación

En esta pantalla figura 4.6 se define el tipo de instalación que queremos, dependiendo de la opción que seleccionemos se definirán los paquetes² a instalar en nuestro sistema.

Los tipo de instalación disponibles son:

Escritorio personal Ideal para computadoras caseras o laptop's, se instala un entorno de escritorio gráfico con toda la paqueteria necesaria para jugar, hacer documentos, hojas de cálculo, presentaciones, editar imagenes.

Estación de trabajo Ideal para programadores ya que se instalan las herramientas necesarias para el desarrollo de software y administración del sistema.

Servidor Si se necesita un servidor web, de archivos, de impresión, de base de datos o de infraestructura(dhcp,dns), esta opción es la ideal. Con esta opción no es instala en ambiente gráfico.

Personalizada Si deseamos seleccionar paquete por paquete esta opción es la indicada.

¹root es el administrador de un equipo Linux, puede hacer y deshacerlo a su antojo.

²Por paquete entenderemos un conjunto de programas comprimidos, se instala un paquete se descomprimen los programas ejecutables.

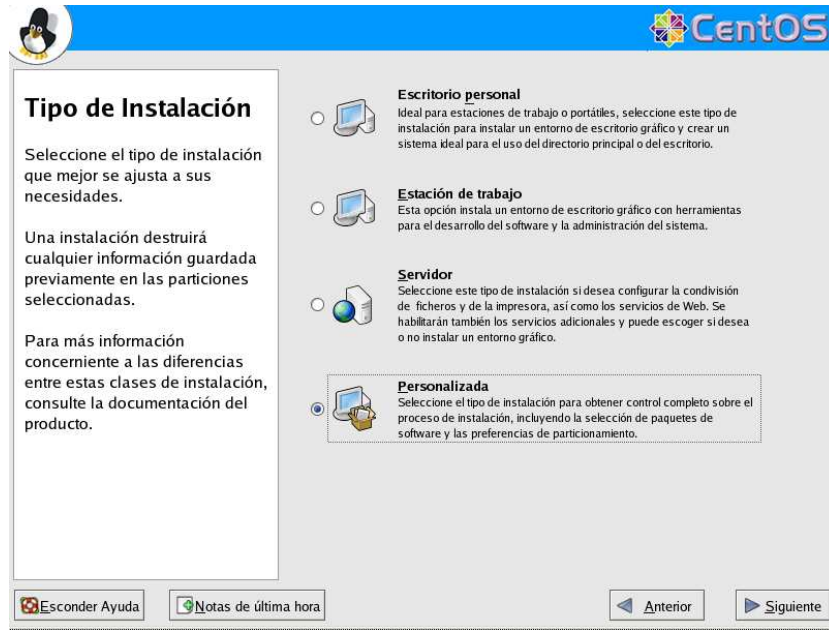


Figura 4.6: Tipo de instalación

4.8.1. Tipo de instalación Personalizada

Si elegimos esta opción tenemos que elegir los paquetes a instalar. Los paquetes están agrupados. Cada grupo tiene paquetes que se instalan de forma obligatoria y otros opcionales. Para poder elegir los paquetes a instalar tenemos que dar click en el botón Detalles de cada grupo de paquetes.

4.9. Particionando el Disco Duro

El particionamiento permite dividir el disco duro en secciones aisladas, donde cada sección se comporta como un propio disco duro. El particionamiento es especialmente útil si ejecuta más de un sistema operativo.

En esta pantalla (figura 4.7), podemos elegir entre realizar un particionamiento automático o un particionamiento manual con Disk Druid.

El particionamiento automático permite realizar una instalación sin tener que particionar los discos. Mientras que para particionar de forma manual, se debe elegir la herramienta de particionamiento **Disk Druid**.

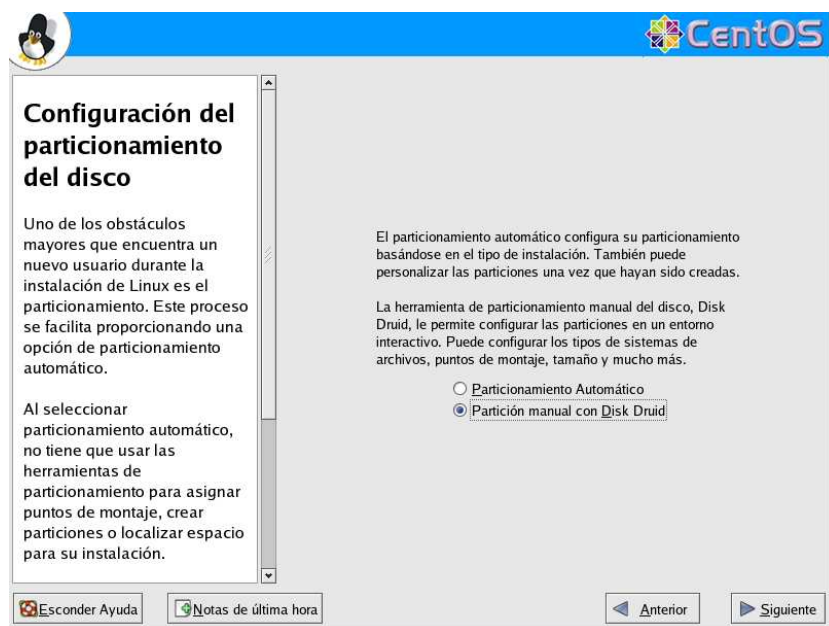


Figura 4.7: Configuración del particionamiento del disco

4.9.1. Particionamiento automático

El particionamiento automático es útil si se cuenta con espacio libre en nuestro disco duro. Tiene las siguientes opciones (figura 4.8):

- **Eliminar todas las particiones Linux del sistema** esta opción elimina sólo las particiones Linux (particiones creadas en una instalación Linux previa). No borrará el resto de particiones del disco(s) duro(s) (tal como VFAT o particiones FAT32).
- **Eliminar todas las particiones del sistema** esta opción elimina todas las particiones del disco duro (esto incluye las particiones creadas por otros sistemas operativos tales como particiones Windows VFAT o NTFS).
- **Mantener todas las particiones y usar el espacio libre existente** esta opción conserva los datos y las particiones actuales, asumiendo que se tiene suficiente espacio disponible en los discos duros.

Siempre es una buena idea respaldar los datos que se tenga. Por ejemplo, si está actualizando o creando un sistema de arranque dual, debería respaldar los datos que desea conservar en su(s) disco(s) duro(s). Los errores sí ocurren y pueden resultar en la pérdida de todos sus datos.

4.9.2. Particionamiento manual

Si se elegimos el particionamiento manual (figura 4.9), se debe indicar al programa de instalación dónde instalar CentOS. Esto se hace mediante la definición de los puntos de montaje para una o más

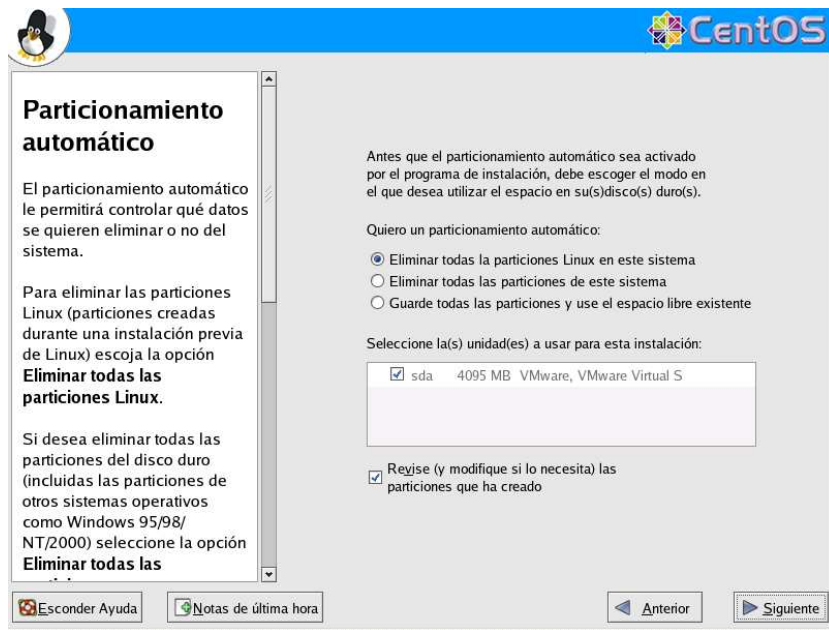


Figura 4.8: Particionamiento automático

particiones de disco. Es necesario también crear y/o eliminar particiones en este punto.

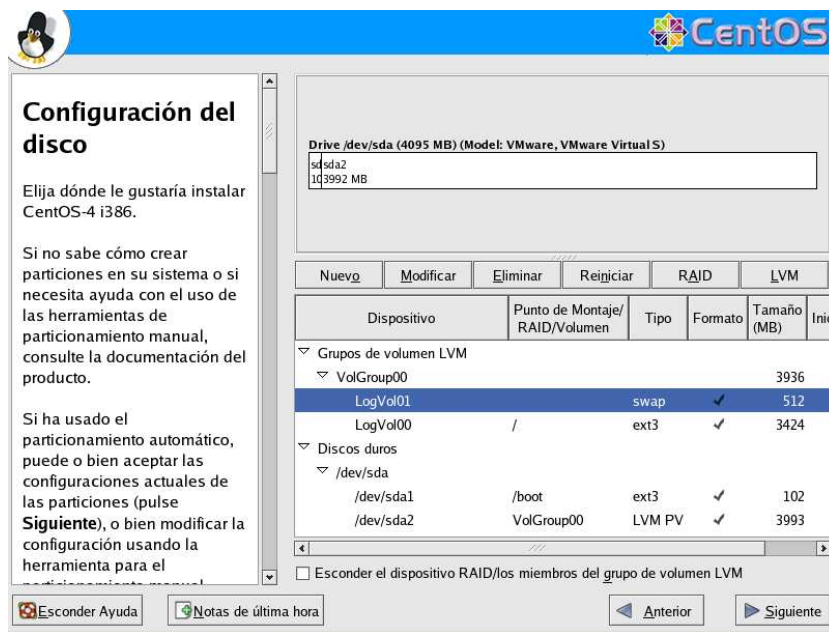


Figura 4.9: Particionamiento con Disk Druid en los sistemas x86, AMD64 y Intel® EM64T

La herramienta de particionamiento usada en CentOS será el Disk Druid. Con la excepción de ciertas situaciones “esotéricas”, Disk Druid normalmente mantiene los requisitos de particionamiento de una instalación normal de CentOS. Disk Druid ofrece una representación gráfica de nuestro(s) disco(s) duro(s).

4.9.3. Esquema de particionamiento recomendado

A menos que tenga una buena razón para hacer lo contrario, se recomienda que cree las siguientes particiones para los sistemas basados en x86.

Una partición swap (de al menos 256 MB) las particiones swap (de espacio de intercambio) son utilizadas para apoyar a la memoria virtual. En otras palabras, los datos son escritos a una partición swap cuando no hay suficiente memoria RAM para almacenar los datos que su sistema esta procesando.

Si no está seguro sobre el tamaño de la partición swap a crear, asigne el doble de la cantidad de RAM disponible en su máquina (pero no más de 2 GB). La partición debe ser de tipo swap.

La selección de la cantidad correcta de espacio de intercambio va a depender de varios factores, incluyendo los siguientes (en orden descendente de importancia):

- Las aplicaciones ejecutándose en la máquina.
- La cantidad de RAM física instalada en la máquina.
- La versión del sistema operativo.

El espacio de intercambio debe ser igual a 2X la RAM física, hasta un máximo de 2 GB de RAM física, y luego 1 X la RAM física para cualquier cantidad sobre los 2 GB, pero nunca menos de 32 MB.

Usando esta fórmula, un sistema con 2 GB de RAM física debería tener 4 GB de swap, mientras que un sistema con 3 GB de RAM tendría 5 GB de swap. La creación de una partición swap grande puede ser de gran ayuda si planea actualizar su RAM posteriormente.

Si el esquema de particionamiento requiere una partición swap más grande que 2 GB, debería crear una partición swap adicional. Por ejemplo, si necesita 4 GB de swap, es recomendable crear dos particiones swap de 2 GB. Si tiene 4 GB de RAM, debería de crear tres particiones swap de 2 GB. Red Hat Enterprise Linux permite hasta 32 archivos swap.

Una partición /boot/ (100 MB) La partición montada en /boot/ contiene el kernel del sistema operativo (que permite que su sistema inicie Red Hat Enterprise Linux), así como los archivos utilizados durante el proceso de arranque. Debido a las limitaciones de muchas BIOSes de computadores, es una buena idea la creación de una pequeña partición para guardar estos archivos. Para la mayoría de los usuarios es suficiente una partición boot de 100 MB.

Una partición root (500 MB a 5.0 GB) aquí es donde se localiza / (el directorio raíz). En esta configuración, todos los archivos (excepto aquellos almacenados en /boot) están en la partición raíz. Una partición raíz de 500 MB le permite el equivalente de una instalación mínima, mientras que una partición raíz de 5.0 GB le permitirá realizar una instalación completa, seleccionando todos los grupos de paquetes.

4.9.4. Tipos de sistemas de archivos

CentOS nos permite crear diferentes tipos de particiones las cuales se describen a continuación:

ext2 Un sistema de archivos ext2 soporta tipos de archivo estándar Unix (archivos regulares, directorios, enlaces simbólicos, etc). Proporciona la habilidad de asignar nombres de archivos largos, hasta 255 caracteres.

ext3 El sistema de archivos ext3 está basado en el sistema de archivos ext2 y tiene una ventaja principal, el *journaling*. El uso de un sistema de archivos *journaling* reduce el tiempo de recuperación tras una caída, ya que no es necesario hacer *fsck* al sistema de archivos. El sistema de archivos ext3 está seleccionado por defecto y su uso es bien recomendado.

volúmen físico (LVM) Mediante la creación de una o más particiones LVM físicas nos permite crear un volúmen lógico LVM. LVM puede mejorar el rendimiento cuando se utilizan discos físicos.

software RAID La creación de dos o más particiones de software RAID nos permite crear un dispositivo RAID (Redundant Array of Independent Disks).

swap Las particiones swap se usan para soportar la memoria virtual. En otras palabras, los datos se escriben en esta partición cuando no hay suficiente RAM para guardar los datos que el sistema está procesando.

vfat El sistema de archivos VFAT es un sistema de archivos Linux que es compatible con los nombres largos de archivos de Microsoft Windows en el sistema de archivos FAT.

4.10. Configuración del gestor de arranque

El siguiente paso en el proceso de instalación es la configuración del gestor de arranque (figura 4.10). El gestor de arranque es el primer software que se ejecuta cuando arranca la computadora. Es responsable de la carga y de la transferencia del control al kernel. El kernel, por otro lado, inicializa el resto del sistema operativo.

GRUB (*GRand Unified Bootloader*), que se instala por default, es un gestor de arranque muy potente ya que puede cargar una gran variedad de sistemas operativos.

Todas las particiones que se pueden arrancar aparecen en una lista, incluso las particiones que usan otros sistemas operativos. La partición que contiene el sistema de ficheros root del sistema tiene la Etiqueta de CentOS-4-i386 (para GRUB) o Linux. Las otras particiones puede que también tengan etiquetas de arranque.

De forma opcional podemos asignar una contraseña al gestor de arranque, esto es útil cuando no hay un buen mecanismo de control de acceso físico a nuestro equipo.

4.11. Configuración de la red

Si no se tiene un dispositivo de red esta pantalla no se mostrará durante la instalación. (figura 4.11)

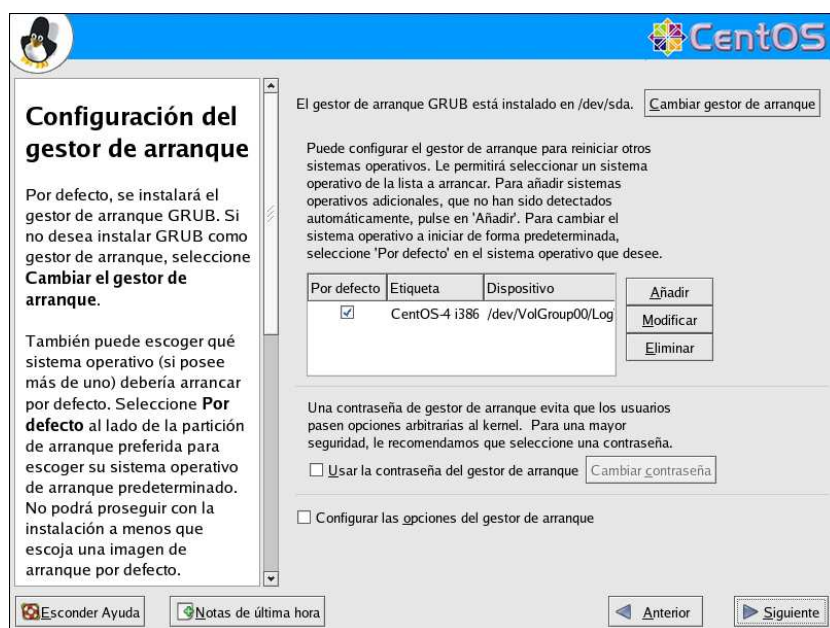


Figura 4.10: Configuración del gestor de arranque

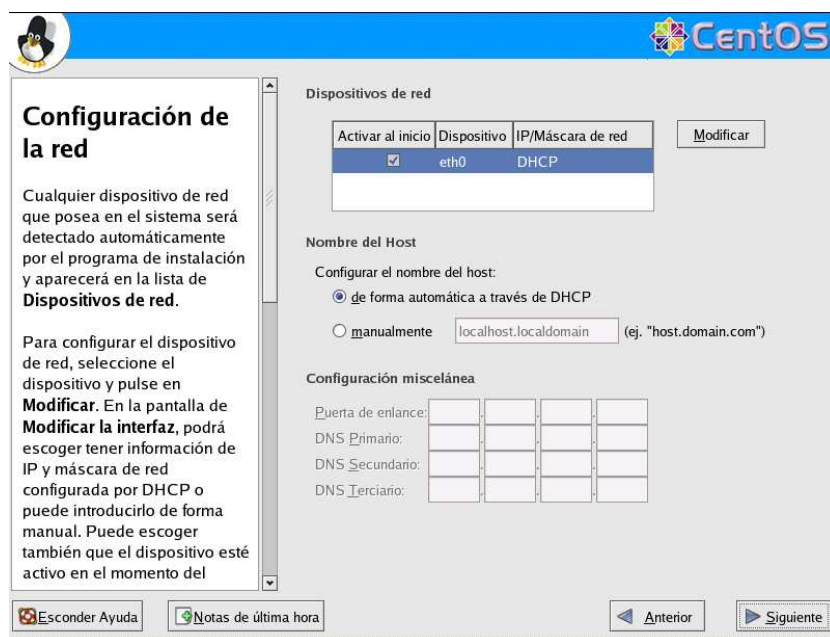


Figura 4.11: Configuración de la red

El programa de instalación automáticamente detecta los dispositivos de red que se tiene y los muestra en la lista **Dispositivos de red**.

Una vez que haya seleccionado el dispositivo de red, haga click en **Modificar**. En la pantalla emergente **Modificar interfaz** se puede elegir configurar la dirección IP y la Máscara de red del dispositivo a través de DHCP (o manualmente si no hemos seleccionado DHCP) y se puede también seleccionar activar el dispositivo en el momento de arranque. Si seleccionamos **Activar al inicio**, el dispositivo de red arrancará cuando el sistema se inicie.

Para cambiar la configuración de red después de la instalación, usar el comando `system-config-network`.

4.12. Configuración del firewall

CentOS ofrece la protección vía firewall³ para una seguridad mejorada del sistema. Un firewall bien configurado puede aumentar significativamente la seguridad del sistema.(figura 4.12)

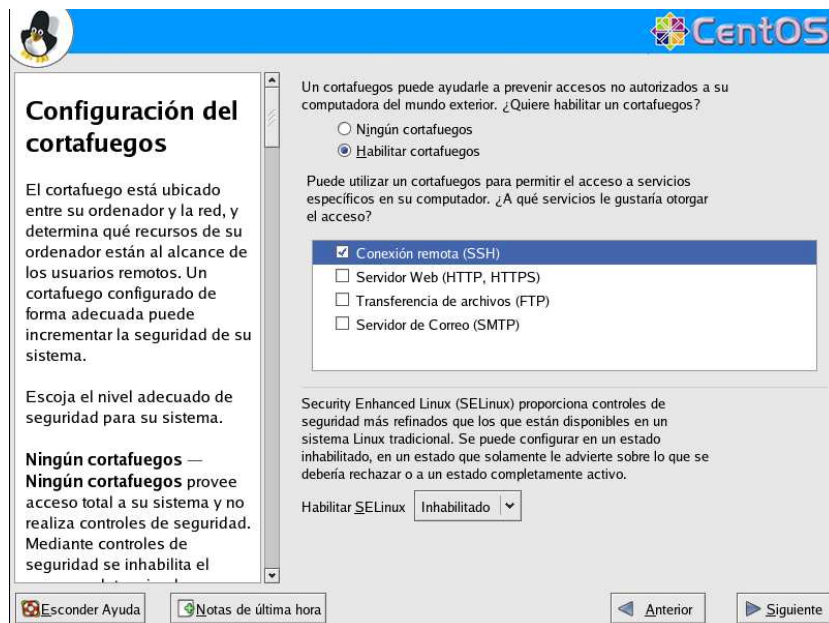


Figura 4.12: Configuración del firewall

Para cambiar la configuración después de la instalación, tenemos disponible la herramienta de administración `system-config-securitylevel`.

Adicionalmente, podemos configurar *SELinux* (Security Enhanced Linux) durante la instalación.

³Firewall es un mecanismo de control de acceso entre el equipo y la red, determina que recursos de nuestro equipo estan accesibles para los usuarios remotos.

SELinux permite proporcionar permisos de forma granulada para todos los sujetos (usuarios, programas y procesos) y objetos (archivos y dispositivos). De forma segura puede otorgar solamente los permisos que una aplicación necesita para funcionar.

La implementación de SELinux en CentOS está diseñada para mejorar la seguridad de varios demonios de servidores a la vez que se minimiza el impacto de las operaciones cotidianas de su sistema.

Están disponibles tres estados para seleccionar durante el proceso de instalación:

Inhabilitado si no desea activar los controles de SELinux en este sistema. La configuración Inhabilitado hace que la máquina no utilice las políticas de seguridad.

Atención para que se le notifique de cualquier detalle. El estado Atención asigna etiquetas a los datos y programas, y los registra, pero no hace cumplir ninguna política. El estado Atención es un buen estado de comienzo para los usuarios que eventualmente desean activar completamente SELinux, pero primero quieren ver los efectos que tendrán las políticas en su operación general del sistema.

Activo si se desea que SELinux actúe en un estado completamente activo. El estado Activo hace cumplir todas las políticas, tal como negar el acceso a los usuarios no autorizados para ciertos archivos y programas, para protección adicional del sistema.

4.13. Selección del soporte del idioma

Se pueden instalar y soportar múltiples idiomas para usarlos en nuestro sistema. Debemos seleccionar un idioma para usarlo como idioma por defecto. El idioma por defecto es aquel que será utilizado por el sistema una vez que la instalación se haya completado. Típicamente, el idioma por defecto es el mismo que seleccionó durante la instalación. (figura 4.13)

Si selecciona tan sólo un idioma, sólo podrá utilizar el idioma especificado una vez finalizada la instalación.

Para cambiar la configuración del idioma una vez finalizada la instalación, usar el comando `system-config-language`.

4.14. Configuración del huso horario

Es importante seleccionar el huso horario que corresponda a la ciudad más cercana a la ubicación física de nuestro equipo. (figura 4.14) Para cambiar la configuración del huso horario una vez finalizada la instalación, usar el comando `system-config-date`.

4.15. Configuración de la contraseña de root

La configuración de la cuenta y la contraseña de `root` es uno de los pasos más importantes durante la instalación. La cuenta de `root` es similar a la cuenta del administrador usada en las máquinas Windows

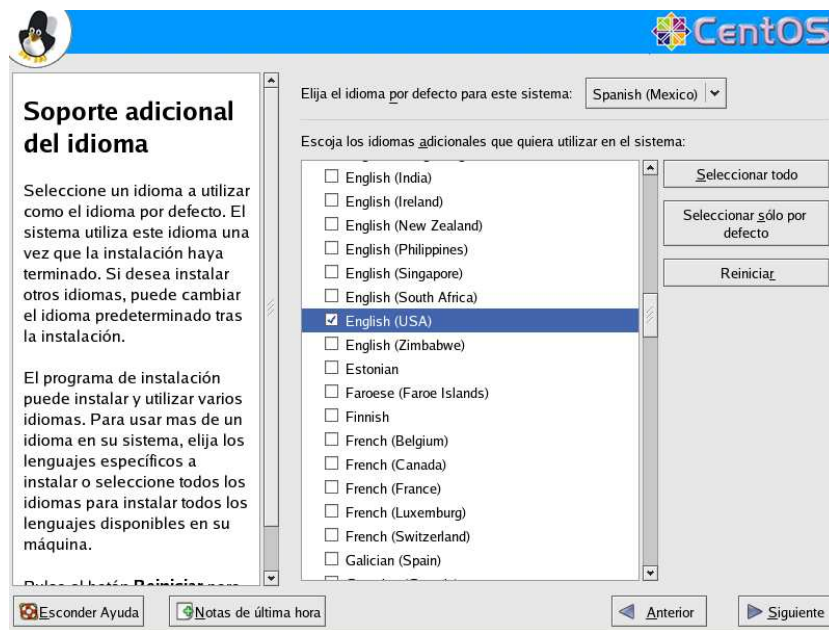


Figura 4.13: Selección del soporte del idioma



Figura 4.14: Configuración del huso horario

2000. La cuenta *root* es usada para instalar paquetes, actualizar RPMs y realizar la mayoría de las tareas de mantenimiento del sistema. (figura 4.15)

El usuario *root* (también conocido como *superusuario*) posee acceso completo al sistema; por este motivo, firmarse al sistema como usuario *root* es aconsejable hacerlo sólo para ejecutar el mantenimiento o administración del sistema.

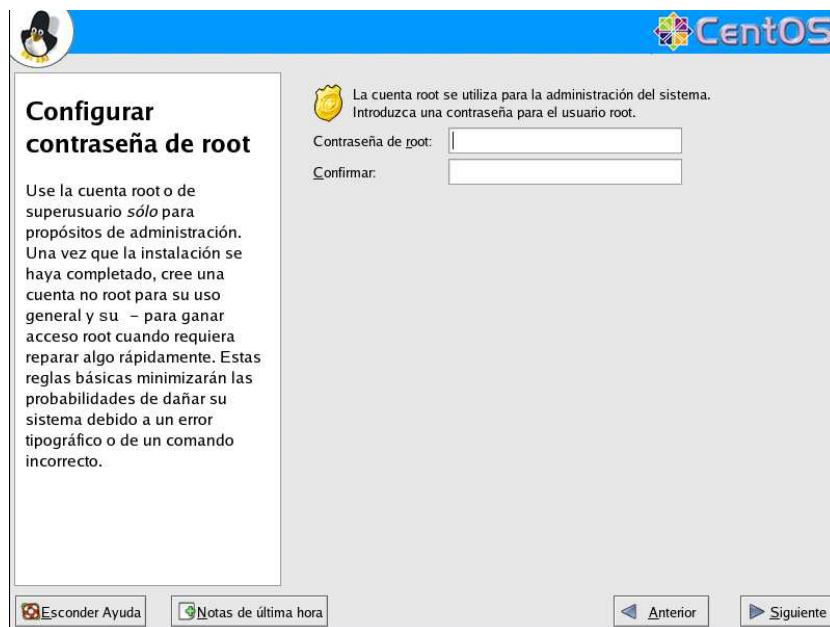


Figura 4.15: Contraseña de root

La contraseña de root debe de tener al menos ocho caracteres y no aparecerá en la pantalla cuando la teclee. Deberá introducirla dos veces; si las dos contraseñas no coinciden, el programa de instalación nos pedirá que la tecleemos de nuevo.

Deberíamos escribir una contraseña de *root* fácil de recordar, pero que no sea obvia o fácil de adivinar. Su nombre, su número de teléfono, qwerty, contraseña, root, 123456 y anteayer serían ejemplos de malas contraseñas. Las contraseñas mejores son aquellas que mezclan números con letras mayúsculas y minúsculas que no formen palabras contenidas en diccionarios, como por ejemplo : Aard387vark o 420BMttNT⁴. Recordemos que la contraseña es sensible a las mayúsculas y minúsculas. Se recomienda que nunca escriba su contraseña pero, si la escribe en un papel, guárdelo en un lugar seguro.

4.16. Selección de grupos de paquetes

Primero, aparecerá la pantalla **Selección de grupos de paquetes** que detalla el conjunto de paquetes que por default se seleccionan de acuerdo al tipo de instalación seleccionada con anterioridad. (figura 4.16)

⁴<http://www.bergen.org/ATC/Course/InfoTech/passwords.html>

Podemos seleccionar grupos de paquetes, los cuales agrupan componentes de acuerdo a una función (por ejemplo, Sistema X Window y Editores), paquetes individuales, o una combinación de los dos.

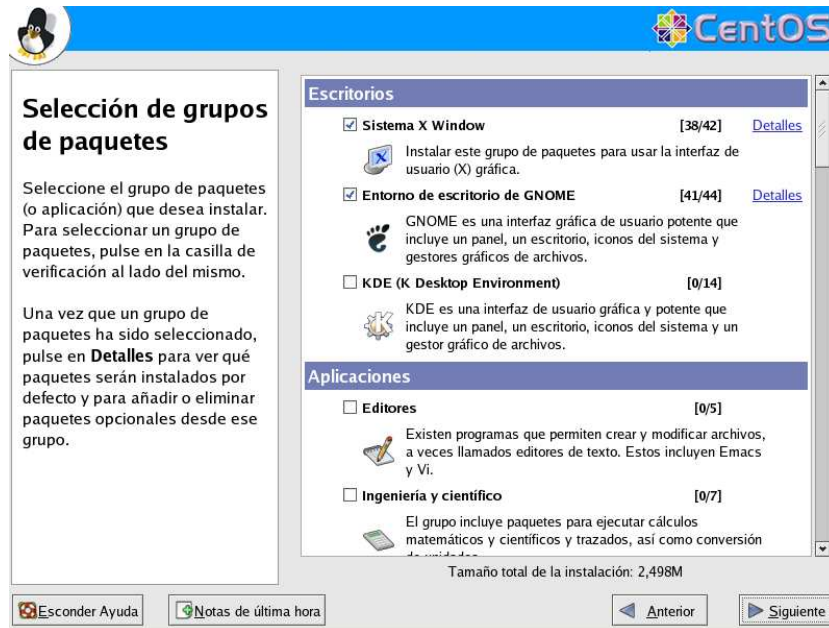


Figura 4.16: Selección de grupos de paquetes

Seleccionemos los componentes que deseamos instalar. Al seleccionar Todo (al final de la lista de componentes) se instalarán todos los paquetes incluidos con CentOS.

Una vez seleccionado un grupo de paquetes, podemos hacer click en Detalles para visualizar los paquetes que se instalarán por defecto y los paquetes opcionales que deseamos eliminar o añadir a ese grupo.

4.17. Listo para la instalación

En este paso (figura 4.17) podemos arrentirnos sin causar daños a nuestro equipo, una vez dando click en aceptar comenzara la copia de los archivos CentOS a nuestro disco.

El instalar nos informará de los discos que se necesitarán durante la copia, tenemos que verificar que los poseemos.

4.18. Instalación de paquetes

El proceso de instalación de paquetes puede durar de 15min a 1hr, dependiendo la cantidad de paquetes que hayamos elegido y la velocidad de nuestro equipo, durante este proceso podemos monitorear el avance(4.18). Aunque lo mejor es irnos a estirar las piernas y solo regresar a cambiar los discos.

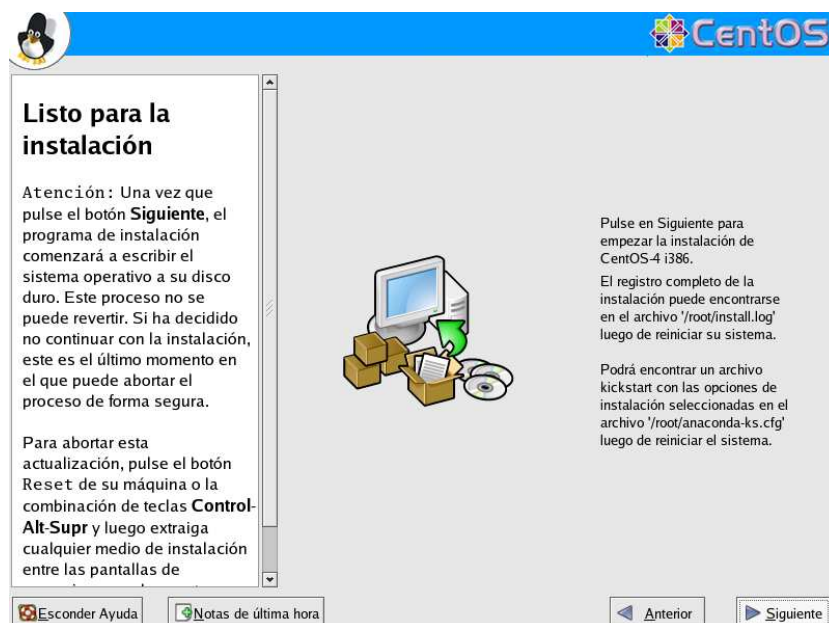


Figura 4.17: Listo para la instalación

Pero si estamos de necios y queremos quedarnos viendo todo el proceso de instalación podemos hacer uso de las consolas virtuales. (Para más información ver la sección 4.2)



Figura 4.18: Instalación de paquetes

4.19. Fin de la instalación

Una vez terminado el proceso de instalación el equipo se reiniciará, es necesario remover el cd de instalación de la unidad de CD-ROM.



Figura 4.19: Instalación de paquetes

Capítulo 5

Introducción al shell de GNU/Linux

5.1. Primeros pasos...

5.1.1. Entrar en sesión

Para poder utilizar nuestro sistema GNU/Linux, es necesario tener una *cuenta* de acceso. Esta cuenta estará dada por el administrador del sistema. Típicamente en un sistema con pocos usuarios, el nombre de la cuenta coincidirá con el nombre del usuario y en un sistema grande será el apellido o la composición de nombre y apellido, o simplemente un alias. Digamos que tenemos al usuario Francisco Medina y es el único que lleva el nombre de Francisco en nuestro trabajo, entonces lo más probable es que su cuenta sea simplemente `paco`. De otra manera, podría ser `fmedina` o `francisco.medina` o cualquier combinación que lo identifique en forma única a lo largo del sistema.

Después de iniciar el sistema operativo, que típicamente será tarea del administrador del sistema, en las terminales aparecerá un mensaje similar al siguiente:

```
CentOS release 4.2 (Final)
Kernel 2.6.9-22.0.1.106.unsupported SMP on an i686
```

```
titan login:
```

donde obviamente, el nombre del sistema operativo y la versión así como el nombre de la máquina, serán diferentes, lo único que podemos garantizar que será igual es la palabra `login:`. Esto nos indica que el sistema está listo para aceptar la entrada de un usuario. Para esto es necesario identificarnos dando el nombre de nuestra cuenta y a continuación aparecerá:

```
CentOS release 4.2 (Final)
Kernel 2.6.9-22.0.1.106.unsupported SMP on an i686
```

```
titan login: paco
```

En este punto paco deberá escribir su *password* o palabra secreta. Cuando la cuenta es creada, el password es fijado por el administrador del sistema. Es conveniente que la primera vez que se firma¹ se cambie el password. Esto se hace utilizando el comando `passwd` que primero nos pregunta el password anterior y después deberemos de dar el nuevo en dos ocasiones para garantizar que fue bien escrito:

```
[paco@titan diploDB]$ passwd
Changing password for user paco.
Changing password for paco
(current) UNIX password:
passwd: Authentication token manipulation error
[paco@titan diploDB]$ passwd
Changing password for user paco.
Changing password for paco
(current) UNIX password:
New UNIX password:
BAD PASSWORD: it is too simplistic/systematic
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Como podemos observar, la primera vez no se dió correctamente el password con lo cual el sistema no acepta la operación de cambiarlo. En la segunda ocasión, después de escribir el nuevo password dos veces, nos indica que es conveniente que usemos un password con caracteres en mayúsculas y minúsculas, así como dígitos y signos de puntuación.

No hay manera de garantizar un password completamente seguro, pero es buena costumbre no usar palabras que tengan sentido ni nombres propios. Si la esposa de paco nació en 1978 y se llama Reyna, `reyna1978` sería un pésimo password para paco, ya que varios de los que dicen ser sus amigos conocen el hecho además de que sería fácil de averiguar.

Sería igual de malo un password como `StarTrek` si sus compañeros y amigos saben que paco es un fan de la serie "Viaje a las Estrellas".

Un método tan bueno como cualquier otro para escoger un password, es tomar una frase no muy conocida y utilizar, por ejemplo, el segundo o tercer caracter de cada palabra que la forma, intercalarle dígitos y signos de puntuación y poner algunas de las letras en mayúsculas. Por ejemplo, con el verso de Oscar Wilde "A veces podemos pasarnos años sin vivir en absoluto, y de pronto toda nuestra vida se concentra en un solo instante." tomamos las letras del inicio de las primeras 8 palabras: `avppasv` y de las letras a las reemplazamos por 4: `4vpp4sv`. Luego de haber hecho esto, y además publicarlo, éste método queda totalmente invalidado.

Es mejor que cada persona idee su propio método para seleccionar el password —que además es conveniente cambiar con cierta frecuencia—, y que le sea fácil de recordar en todo momento. Nunca se debe de escribir un password en ningún sitio. Mucho menos tatuarselo en el pecho, que es de mal gusto y además como se habrá de cambiar con frecuencia, termina uno con la epidermis hecha un asco de tanto *password* tachado.

Habrán sistemas donde en lugar de con una aburrida pantalla de texto nos encontraremos con una excitante ventana en modo gráfico, pero el procedimiento será siempre el mismo.

¹Acción que comprende dar cuenta y password, y que en adelante llamaremos entrar en sesión.

Ahora intentemos con otros comandos, debemos escribir el comando tal cual se muestra, tecleando *Enter* al final de cada línea.

```
date
```

comando que muestra la fecha y hora.

```
who
```

muestra los nombres de usuarios conectados al sistema en este momento.

```
hostname
```

muestra el nombre de la máquina UNIX.

5.1.2. Cuentas, grupos y passwords

UNIX es un sistema operativo multitarea y multiusuario, por lo que se deben establecer ciertos mecanismos de para proteger los datos de cada usuario y que puedan ser compartidos en caso necesario. UNIX posee un mecanismo de permisos asociados a cada archivo. Este mecanismo permite que los archivos y directorios pertenezcan a un usuario en particular. UNIX también permite que los archivos sean compartidos entre usuarios y grupos de usuarios. El comportamiento por defecto en la mayoría de los sistemas es que todos los usuarios pueden leer los archivos de otro usuario, pero no pueden modificarlos o borrarlos.

Los grupos de usuarios se definen normalmente en función del tipo de usuario. Por ejemplo, en una Universidad, los usuarios pueden clasificarse como estudiantes, profesores, invitados, etc.

Cada usuario (perteneciente a un grupo de usuarios) tiene asociado un nombre, una palabra clave o password, un directorio propio (home) y un shell de inicio:

- **Nombre:** Identificación del usuario cuando entra en la máquina (login).
- **Clave:** Palabra oculta que sólo conoce el usuario.
- **UID, GID:** Números de identificación de usuario y grupo, respectivamente.
- **Directorio propio (home):** Directorio inicial donde se situará el usuario al entrar en el sistema.
- **Shell de inicio:** Primer proceso que se arranca una vez dentro del sistema.

Existen diferentes categorías de usuarios en función de sus privilegios:

- **Superusuario o root (UID=0):** Es el administrador del sistema. Tiene todos los privilegios.
- **Usuarios normales:** El resto de usuarios que pertenecen a distintos grupos, los cuales pueden tener una serie de propiedades comunes.
- **Usuarios especiales:** Asignados a tareas específicas por el sistema, generalmente de información o manejo de aplicaciones ya instaladas de uso común a usuarios externos o internos. Por ejemplo: mail (se encarga de recoger el correo y repartirlo a los diversos usuarios), lp (se encarga de aceptar trabajos de impresión y mandarlos a la impresora), bin, admin, ...

5.1.3. ¿Cómo ejecutar comandos?

Sintaxis de los comandos en Linux

```
% Comando opciones parámetros
```

Las opciones son modificadores para los comandos, que no siempre es necesario darlas, cada opción se representa, en la mayoría de los casos, con una sola letra precedida de un signo -, los parámetros son información que el comando necesita, lo que depende del comando en particular

5.1.4. Directorios

Cada usuario tiene un directorio propio, llamado a veces "directorio home". Cuando el usuario ingresa al sistema ya está ubicado en su directorio propio. El comando

```
pwd
```

muestra el directorio actual.

```
cd /home
```

cambia hacia el directorio /home, lo que puede verificarse con el comando *pwd*.

```
cd
```

sin parámetros devuelve al usuario a su directorio propio, desde cualquier lugar donde esté. Este comando es útil cuando se han hecho varios cambios de directorio y se quiere retornar a una situación conocida, ubicándose en el directorio propio.

5.1.5. Listado de archivos

Ensayemos el comando *ls*:

```
ls
```

lista archivos del directorio actual.

```
ls -l /bin
```

lista archivos en el directorio /bin; aquí se encuentran los archivos de comandos ejecutables del sistema. No cambia de directorio; el directorio actual sigue siendo el mismo.

```
ls -l
```

lista archivos en formato largo, dando detalles. El -l se llama opción o bandera; se lee "menos ele".

La salida obtenida consta de renglones parecidos a

```
-rw-rw-rw- 1 paco users 29656 abr 8 09:58 IntUnix.latex
```

y se interpretan así:

-

indica el tipo de archivo de que se trata, con esta convención:

- - archivo común,
- **d** directorio,
- **l** enlace o referencia a otro archivo.

```
rw-rw-rw
```

son los permisos del archivo;

- **r** (read) permiso para leer el archivo
- **w** (write) permiso para modificar o eliminar el archivo
- **x** (execute) si se trata de un archivo, permiso para ejecutarlo como programa; si se trata de un directorio, permiso para ingresar en él y recorrerlo.

Los tres grupos de 3 caracteres indican permisos para el dueño del archivo (paco), su grupo (users) y el resto del mundo.

```
1
```

cantidad de enlaces, referencias a este archivo desde otros archivos ubicados en diferentes lugares.

```
paco
```

nombre del usuario dueño del archivo. `users` nombre del grupo al que pertenece el archivo

`29656` tamaño en bytes del archivo. Si se trata de un directorio, este número es de control del sistema, sin interpretación inmediata para el usuario.

```
abr 8 09:58
```

fecha y hora de última modificación. Si no aparece el año, se asume el año corriente.

```
IntUnix.latex
```

 nombre del archivo. Notar que el nombre del archivo está siempre al final.

```
ls -a
```

muestra también archivos ocultos, normalmente no visibles en el listado. Los archivos cuyo nombre empieza con un punto son ocultos, en este sentido. Las entradas `.` y `..` representan el directorio actual y el directorio padre, respectivamente.

```
ls -la
```

formato largo y archivos ocultos.

```
ls -la /var
```

listado de archivos visibles y ocultos en formato largo del directorio `/var`.

5.1.6. Manual de UNIX

UNIX dispone de un manual en línea o “páginas man” con información sobre comandos, archivos y otros elementos del sistema operativo. Aunque muy técnicas y a veces difíciles de comprender,

son una referencia obligada para operar con solvencia.

```
man ls
```

muestra la página man del comando `ls`, paginada para poder leer una pantalla por vez. Para salir antes de terminar, teclear ‘q’.

```
man man
```

muestra la página man del propio comando `man`.

```
man man >man.txt
```

redirecciona la salida y graba el contenido de la página man en el archivo *man.txt*, lo que se puede verificar con `ls`.

5.1.7. Contenido de un archivo

```
cat man.txt
```

muestra el contenido del archivo. El archivo `man.txt`, por provenir de una página man, contiene muchos caracteres de control para regular su despliegue en pantalla. El comando `cat` no interpreta estos caracteres de control. El comando `cat` permite también concatenar archivos, según se verá. `head man.txt` muestra las primeras 10 líneas de un archivo. `tail man.txt`

muestra las 10 líneas finales de un archivo.

```
cat man.txt | more
```

lee una nota larga paginando; la salida de `cat` es tomada por `more`, que presenta la información página por página. Para interrumpir el paginado de `more`, teclear ‘q’. La tecla **espaciadora** avanza una pantalla, la combinación de teclas **Ctrl-B** retrocede una pantalla.

```
more man.txt
```

 presenta la información de `man.txt` ya paginada.

```
ls -l /etc | more
```

muestra el extenso contenido del directorio `/etc` paginando la salida.

5.1.8. Crear y borrar un archivo

```
touch nota.vacia
```

crea el archivo `nota.vacia`, sin contenido alguno.

```
ls -l nota.vacia
```

muestra datos del archivo creado.

```
cat nota.vacia
```

no muestra nada, el archivo está vacío.

```
touch .archivo_oculto
```

crea un archivo vacío y oculto.

```
ls
```

no muestra el archivo oculto creado, pero

```
ls -a
```

sí lo muestra.

```
ls -la
```

muestra datos en formato largo del archivo oculto.

```
rm nota.vacia
```

borra el archivo nota.vacia. Como muchos comandos de UNIX, su nombre deriva de palabras inglesas: rm proviene de "remove", eliminar o borrar.

```
rm -i man.txt
```

borrado interactivo, pide confirmación antes de actuar.

```
rm .archivo_oculto
```

```
ls -la
```

borra el archivo oculto y verifica listando los archivos del directorio.

```
echo Mensaje en pantalla
```

muestra en la pantalla la leyenda indicada.

```
echo Este es el archivo mensaje1 >mensaje1
```

direcciona la salida del comando echo y graba la leyenda en el archivo mensaje1.

```
ls -l mensaje1
```

```
cat mensaje1
```

muestra datos del archivo y verifica su contenido.

```
echo Esta linea es agregada >> mensaje1
```

```
echo Esta es otra linea agregada >> mensaje1
```

```
cat mensaje1
```

redirecciona la salida de echo para agregar dos líneas más al archivo ya existente mensaje1. Verifica el contenido con *cat*.

5.1.9. Usuarios en el sistema

```
who
```

muestra los usuarios que están actualmente en el sistema. Indica identificador de usuario, terminal en que está conectado, fecha y hora de ingreso al sistema.

```
who am i
```

da información sobre el usuario que está trabajando, indicando su máquina y nombre de usuario, terminal, fecha y hora.

```
whoami
```

presenta sólo el nombre del usuario que está operando.

`id` proporciona la identificación del usuario invocante, dando el nombre de usuario y su número (UID), nombre de grupo primario y su número (GID), nombres de otros grupos a los cuales pertenece (si los hay) y sus números.

```
id jperez
```

proporciona datos de identificación del usuario indicado (jperez).

```
finger
```

proporciona nombre del usuario en el sistema, nombre en la vida real y otros datos del usuario invocante, indicando si está en este momento en el sistema, y si tiene correo por leer.

```
finger paco
```

proporciona información sobre el usuario indicado.

5.1.10. Write

Con el comando *write* se envía algún mensaje a un usuario.

```
write username [tty]
```

el mensaje termina con la combinación de teclas *CTRL-D*.

5.1.11. Talk

Talk es un programa que permite a dos usuarios en el sistema comunicarse escribiendo en el teclado. Al invocar *talk* la pantalla se divide en dos partes, cada una correspondiente a uno de los usuarios. Ambos pueden escribir simultáneamente, y ambos ven la salida en su parte correspondiente de la pantalla.

```
talk usuario1
```

solicita apertura de una sesión de *talk* al usuario1, que debe responder con otro comando similar cuando recibe el pedido.

Para terminar la sesión de talk, cualquiera de los usuarios puede digitar *Ctrl-C*.

El comando *mesg* permite regular si se desea recibir mensajes o no. Para evitar recibir mensajes de talk es posible bloquear a otros usuarios el acceso a la terminal donde uno está trabajando;

quienes intenten iniciar una sesión recibirán un mensaje indicando que la terminal destino no está habilitada para recibir mensajes.

```
mesg n
```

deshabilita recepción de mensajes,

```
mesg y
```

habilita recepción de mensajes.

```
mesg
```

muestra el estado: si responde *z* está habilitada la recepción, si responde *n* se rechazan los pedidos de conexión.

5.1.12. Cambio de contraseña

La contraseña o password, debe ser una palabra difícil de adivinar por otras personas y fácil de recordar por el usuario. Se recomienda que esta palabra reúna las siguientes características: Que no sea una palabra del diccionario Que no sea una sucesión de letras adyacentes Evitar los diminutivos o alias Evitar nombres de familiares, registro federal y fechas de nacimiento. Una contraseña debe tener al menos seis caracteres. Utilizar iniciales de frases comunes, combinadas con números Puesto que la contraseña asegura la identificación ante el sistema, debe respetar los siguientes lineamientos: debe ser confidencial, de modo que no debe ser compartida con otra persona, no debe escribirse en ningún lugar. Tratarla como número de identificación personal (NIP). No reusar una contraseña ya utilizada, ya que incrementa la probabilidad de que alguien la obtenga. Cómo cambiar la contraseña?

passwd pide la vieja contraseña y luego la nueva; la nueva contraseña deberá ingresarse dos veces, para evitar posibles errores de digitación.

5.1.13. Fin de Sesión

```
exit
```

termina la sesión con UNIX.

Las teclas *Ctrl-D* también terminan la sesión.

5.2. El Sistema X Window

Mientras que el corazón de CentOS es el kernel, para muchos usuarios, la cara del sistema operativo es el entorno gráfico proporcionando por el Sistema X Window, también llamado simplemente X.

En el mundo UNIX, los entornos de ventanas han existido desde hace décadas, siendo éstos precursores de muchos de los utilizados en los sistemas operativos actuales. A través de los años X se ha convertido en el entorno gráfico (GUI) predominante para sistemas operativos del tipo UNIX.

El entorno gráfico para CentOS es suministrado por la Fundación X.Org, una implementación de código abierto creada para manejar el desarrollo y la estrategia para el sistema X y sus tecnologías asociadas. X.Org es un proyecto de gran escala que se apoya en un gran número de desarrolladores en todo el mundo. Presenta una amplia gama de soporte para diferentes dispositivos de hardware y arquitecturas, así como la posibilidad de ejecutarse en diferentes sistemas operativos y plataformas. Este lanzamiento de Red Hat Enterprise Linux incluye específicamente el lanzamiento X11R6.8 del sistema X Window.

El sistema X Window utiliza una arquitectura cliente-servidor. El servidor de X (el binario Xorg) escucha por conexiones desde las aplicaciones cliente X a través de la red o una interfaz local de loopback. El servidor gestiona la comunicación con el hardware, que puede ser una tarjeta gráfica, un monitor, un teclado o un ratón. Las aplicaciones cliente de X existen en el espacio del usuario, creando una interfaz gráfica del usuario (GUI) y pasando peticiones al servidor de X.

5.2.1. Entornos de escritorio y gestores de ventanas

Una vez que un servidor X se esté ejecutando, las aplicaciones cliente X pueden conectarlo y crear una GUI para el usuario. Un rango de GUIs están disponibles con Red Hat Enterprise Linux, desde el rudimentario Administrador de pestañas de ventanas hasta un entorno de escritorio altamente desarrollado, interactivo como GNOME, con el que la mayoría de los usuarios de Red Hat Enterprise Linux están familiarizados.

Para crear lo último, una GUI más avanzada, se deben conectar dos clases principales de aplicaciones clientes X al servidor X: un entorno de escritorio y un gestor de ventanas.

Entornos de escritorio

Un entorno de escritorio une diferentes clientes de X, los cuales cuando se usan juntos crean un ambiente de usuario gráfico común y una plataforma de desarrollo.

Los entornos de escritorio tienen características avanzadas las cuales permiten a los clientes X y a otros procesos en ejecución, comunicarse unos con otros a la vez que se permite a todas las aplicaciones escritas para funcionar en ese ambiente a que realicen tareas avanzadas, tales como operaciones de arrastrar y soltar.

CentOS proporciona dos entornos de escritorio:

GNOME Es el entorno de escritorio por defecto en Red Hat Enterprise Linux basado en el conjunto de herramientas gráficas GTK+ 2.

KDE Un entorno de escritorio alternativo basado en el conjunto de herramientas gráficas Qt 3.

Ambos entornos GNOME y KDE tienen aplicaciones de productividad avanzadas, tales como procesadores de palabras, hojas de cálculo y navegadores Web así como herramientas para personalizar la apariencia de la GUI. Adicionalmente, si ambas bibliotecas están presentes, la GTK+ 2 y la Qt, las aplicaciones KDE pueden ejecutarse en GNOME y viceversa.

Gestores de ventanas

Los gestores de ventanas son programas clientes de X que son o parte del entorno de escritorio o, en algunos casos, independientes. Su propósito principal es controlar la forma en que las ventanas gráficas son posicionadas, redimensionadas o movidas. Los manejadores de ventanas controlan las barras de títulos, el comportamiento del foco, los vínculos del botón del ratón y teclas especificadas por el usuario.

Se incluyen cuatro gestores de ventanas con CentOS:

kwin El gestor de ventanas KWin es el manejador por defecto para el entorno KDE. Es un manejador de ventanas eficiente que soporta temas personalizados.

metacity El gestor de ventanas Metacity es el manejador de ventanas por defecto del entorno GNOME. Es un manejador de ventanas simple y eficiente que también soporta temas personalizados.

mwm El gestor de ventanas Motif, es un gestor básico independiente. Puesto que está diseñado para ser un gestor que se ejecuta de forma independiente, no se debería utilizar en conjunto con los entornos de escritorios GNOME o KDE.

wm El minimalista Administrador de pestañas de ventanas, el cual proporciona el conjunto de herramientas más básicas de cualquier gestor de ventanas y puede ser usado bien sea de forma independiente o con un entorno de escritorio. Es instalado como parte de la versión X11R6.8.

5.2.2. GNOME

GNOME o Gnome es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window, se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

El proyecto **GNOME** (GNU Network Object Model Environment) surge en agosto de 1997 como proyecto liderado por Miguel de Icaza para crear un entorno de escritorio completamente libre para sistemas operativos libres, en especial para GNU/Linux. Desde el principio, el objetivo principal de GNOME ha sido proporcionar un conjunto de aplicaciones amigables y un escritorio fácil de utilizar.

En esos momentos existía otro proyecto anterior con los mismos objetivos, pero con diferentes medios: KDE. Los primeros desarrolladores de GNOME criticaban a este proyecto por basarse en la biblioteca de controles gráficos Qt por no ser compatible con los fundamentos del software libre. Años más tarde los problemas de licencia de Qt se han resuelto y estas críticas han cesado. Sin embargo, los dos proyectos siguen rumbos tecnológicos distintos y se hacen una competencia amigable.

Como con la mayoría de los programas GNU, GNOME ha sido diseñado para ejecutarse en toda la gama de sistemas operativos de tipo Unix con X Window, y especialmente pensado para GNU/Linux. Desde sus inicios se ha utilizado la biblioteca de controles gráficos GTK, originalmente desarrollada para el programa The GIMP.

A medida que el proyecto ha ido progresando en los últimos años, los objetivos del mismo se han extendido para tratar una serie de problemas en la infraestructura Unix existente.

Actualmente el proyecto evoluciona bajo amparo de la Fundación GNOME.

Cuando entramos a sesión en CentOS aparece un escritorio similar al de la figura 5.1.

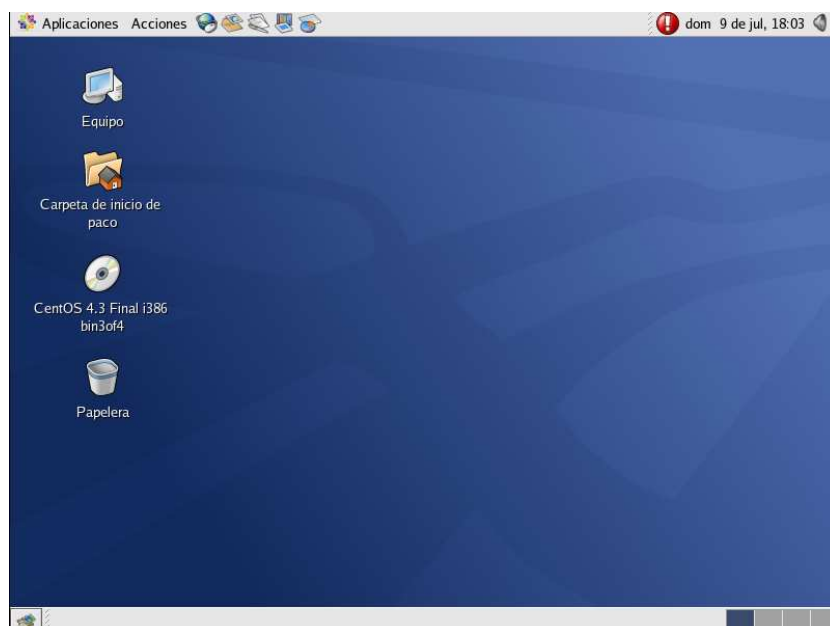


Figura 5.1: Gnome Desktop

Para salir del sistema basta con dar click sobre el icono de CentOS y seleccionar la opción *terminar sesión*. Una ventana de dialogo aparece para confirmar la que deseamos salir de sesión.

5.3. Interpretador de comandos: Shell

El intérprete de comandos o “shell” actúa entre el sistema operativo y el operador. Provee al usuario una interfaz hacia el sistema operativo. El usuario dialoga con el intérprete de comandos, y éste transfiere las órdenes al sistema operativo, que las ejecuta sobre la máquina.

El intérprete de comandos es el programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el sistema operativo.

En otras palabras el objetivo de cualquier intérprete de comandos es ejecutar los programas que el usuario teclea en el *prompt* del mismo. El *prompt* es una indicación que muestra el intérprete para anunciar que espera una orden del usuario. Cuando el usuario escribe una orden, el intérprete ejecuta dicha orden. En dicha orden, puede haber programas internos o externos: Los programas internos son aquellos que vienen incorporados en el propio intérprete, mientras que los externos son programas separados (ej: aplicaciones de */bin,/usr/bin,...*).

En el mundo Linux/Unix existen tres grandes familias de Shells como se muestra en la tabla 5.1. Estas se diferencian entre sí básicamente en la sintaxis de sus comandos y en la interacción con el usuario.

Tipo de Shell	Shell estándar	Clones libres
AT&T Bourne shell	sh	ash, bash, bash2
Berkeley "C" shell	csh	tcsh
AT&T Korn shell	ksh	pdksh, zsh
Otros interpretes	—	esh, gush, nwsh

Cuadro 5.1: Interpretes de comandos en Linux/Unix

5.4. Estructura de Comandos

La estructura general de un comando es la siguiente:

nombre *opciones argumentos*

nombre es el nombre del comando. Las opciones o banderas controlan la forma en que actúa el comando; van precedidas por el signo - (menos). Los argumentos son comúnmente nombres de archivos o nombres de login de usuarios.

Necesitamos crear dos archivos de prueba:

Con el comando *cat* puede crearse un archivo aceptando el teclado como entrada estándar y direccionando la salida estándar hacia un archivo:

```
cat >nota
```

teclar las siguientes líneas, tal cual están, terminando cada línea dando "Enter"; finalizar con "Ctrl-D". Digitar con cuidado: una vez pasados al renglón siguiente, no puede volverse atrás para corregir.

```
El shell es un intérprete de comandos.
Es también un lenguaje de programación.
Los programas escritos con el shell se llaman scripts.
```

Teclear "Ctrl-D" para finalizar el ingreso.

```
cat nota
```

muestra el contenido digitado.

Crear del mismo modo el archivo LEAME, haciendo

```
cat >LEAME
```

con este contenido:

```
Este es el archivo LEAME.
Recuerde que UNIX diferencia entre mayúsculas y minúsculas.
El archivo LEAME tendrá este contenido.
```

```
cat LEAME
```

para verificar el contenido.

```
pr -l23 nota pr -l23 LEAME
```

da formato y muestra el archivo indicado en páginas de 23 líneas de largo para que quepa en la pantalla.

```
pr -d -l23 LEAME
```

muestra el archivo LEAME a doble espacio, largo de página 23.

```
pr -d -o15 -l23 LEAME
```

doble espacio, margen izquierdo de 15 espacios, largo de página 23.

```
pr -l23 nota LEAME | more pr -l23 nota LEAME | more
```

muestra los dos archivos uno tras otro, pero los pagina separados.

```
cat nota LEAME
```

concatena los archivos; los presenta uno tras otro.

```
cat nota LEAME | pr -l23
```

concatena los archivos y luego les da formato a los dos juntos.

5.5. Expansiones de la línea de comandos

Los comodines son caracteres que sustituyen cadenas de caracteres.

Caracter	Descripción
*	secuencia de caracteres cualesquiera, 0 o más.
?	caracter cualquiera, uno y uno sólo; debe aparear un caracter.
[...]	sustituye los caracteres indicados individualmente.

5.6. Variables de ambiente

Las variables de ambiente son cadenas de caracteres de la forma

nombre=valor **nombre** es cualquier cadena de caracteres que no incluya \$ ni espacio (b); *valor* es cualquier cadena; puede incluir el espacio si el valor está entre comillas.

```
MUESTRA='hola chicos'
```

asigna "hola chicos" a la variable *MUESTRA*. `echo $MUESTRA`

muestra en pantalla el contenido de la variable *MUESTRA*. Para exhibir el contenido de una variable de ambiente se escribe `echo $nombre-variable`. En general, para hacer uso del contenido de una variable, se escribe *\$nombre-variable*.

Es una costumbre muy arraigada en UNIX usar mayúsculas para los nombres de variable, así como es una regla usar minúsculas para los comandos. Las opciones pueden ser mayúsculas y minúsculas; la opción `-a` no es lo mismo que `-A`.

Las variables de ambiente pueden ser usadas como nombres de comando o como argumentos de un comando.

```
XYZ='cat nota'  
$XYZ
```

la variable `XYZ` contiene un comando, y al ser invocado su contenido, el comando es ejecutado.

```
echo hola $MUESTRA chicos
```

la variable puede usarse en la formación de cadenas.

```
echo hola$MUESTRAchicos
```

la variable puede embeberse en una cadena; las llaves delimitan el nombre de la variable.

```
env
```

muestra las variables de ambiente definidas. Muchas son fijadas en el ingreso del usuario al sistema (variables de login); y otras son propias del shell (variables del shell).

```
echo $LOGNAME
```

muestra el nombre de login del usuario.

```
echo $HOSTNAME
```

muestra el nombre de la máquina.

```
echo $NOEXISTE
```

Las variables no definidas no muestran nada cuando son referenciadas

5.7. Entrecomillado de argumentos

Los espacios separan argumentos. El entrecomillado obliga a tratar una cadena con espacios como si fuera un solo argumento. Cuando se usan comillas dobles (`" "`) el shell interpreta las variables de ambiente incluídas, actuando según su contenido:

```
MUESTRA='Mi nombre de login es $LOGNAME'  
echo $MUESTRA
```

Cuando se usan comillas simples (`' '`) el shell no interpreta las variables de ambiente, tratando sus nombres como cadenas:

```
MUESTRA='La variable $LOGNAME contiene su nombre de login'  
echo $MUESTRA
```

Las comillas simples permiten usar comillas dobles:

```
echo '''hola chicos'''
```

5.8. PS1

Aunque generalmente los omitimos, en los ejemplos de esta sección mostraremos el indicador de comandos y la salida de ejecución.

La variable símbolo de indicador de comandos nivel 1 PS1 (Prompt Symbol level 1) es un valor de ambiente que puede no aparecer en la salida de env:

```
$ echo $PS1
```

```
$
```

responde con el indicador de comandos actual, \$.

```
$ PS1=hola:
```

```
hola:
```

carga un nuevo indicador de comandos; el sistema responde con el nuevo indicador de comandos, hola:.

```
hola: MUESTRA="ingrese comando:"
```

```
hola: PS1=$MUESTRA
```

```
ingrese comando: MUESTRA=hola
```

```
ingrese comando: echo $MUESTRA
```

```
hola
```

```
ingrese comando:
```

la variable *MUESTRA* cambió pero el indicador no cambió;

```
ingrese comando: PS1='$' $
```

repone el indicador de comandos.

5.9. Entrada estándar y salida estándar.

Los comandos leen como entrada una secuencia de caracteres (flujo de entrada o "input stream"), y escriben a la salida otra secuencia de caracteres (flujo de salida o "output stream"). Estas secuencias no tienen estructura interna alguna, y se les llama entrada estándar y salida estándar.

`cat nota >nota2` dirige la salida estándar a un archivo que crea o sobrescribe, en vez de a la pantalla. `mail juan <nota`

toma la entrada estándar de un archivo en vez del teclado.

```
pr <nota >nueva.nota
```

el archivo nota es la entrada, el archivo nueva.nota es la salida.

```
cat nueva.nota
```

muestra el archivo donde se grabó la salida con formato.

Los símbolos < y > redirigen entrada y salida estándar.

Ahora probemos los siguiente:

```
MUESTRA=/etc/passwd
```

```
pr <$MUESTRA >lista.usuarios
```

```
more lista.usuarios
```

Los comandos y los archivos tienen una estructura orientada a carácter. En muchos comandos se puede intercambiar teclado por archivos o dispositivos de hardware.

```
cat nota
```

```
cat <nota
```

son comandos equivalentes porque cat opera sobre un archivo y también sobre la entrada estándar.

```
cat nota LEAME nueva.nota
```

```
cat nota - nueva.nota <LEAME
```

son comandos equivalentes. El signo menos (-) aislado equivale a tomar en ese punto la entrada estándar, que usualmente está asignada al teclado.

5.10. Fin de flujo.

El ingreso de datos desde el teclado, así como el despliegue en pantalla, se manejan en UNIX como "flujos de caracteres", una serie de caracteres uno tras otro. Para indicar en el teclado el fin de un flujo de caracteres se usa Ctrl-D. Este símbolo no forma parte del ingreso; simplemente indica el final del ingreso, que ya no se escribirá más. En UNIX no hay un carácter especial para indicar fin de archivo; el sistema sabe cuando termina un archivo por medio de contadores.

```
cat nota - nueva.nota >arch.salida
```

procesa el archivo nota, lee lo que se digite en el teclado hasta recibir un Ctrl-D, procesa nueva.copia y escribe todo en el archivo arch.salida; se reúnen tres fuentes distintas en una única salida.

```
cat arch.salida
```

muestra todo lo reunido.

Agregar la salida estándar a un archivo.

```
cat <nota >nota.copia
cat nota nota.copia >doble.nota
rm nota.copia
```

doble.nota contiene nota dos veces. Se logra lo mismo con

```
cat nota >dup.nota
cat nota >> dup.nota
```

El símbolo >> redirige la salida estándar a un archivo, pero agregando al final del mismo en lugar de reemplazar su contenido.

5.11. Error estándar.

Además de los flujos de entrada y salida estándar, existe un tercer flujo de caracteres, el error estándar, hacia donde se dirigen los mensajes de error. El error estándar está dirigido habitualmente a la pantalla, pero mediante 2> o 2>> se redirige el error estándar hacia un archivo de errores. Los flujos estándar se reconocen por los siguientes números:

- 0 la entrada estándar, usualmente el teclado;
- 1 la salida estándar, usualmente la pantalla;
- 2 el error estándar, usualmente la pantalla.

```
echo Archivo de errores >salida.error
cat noexiste 2>>salida.error
cat salida.error
```

el archivo salida.error contiene el mensaje inicial y el de error.

```
ls noexiste 2>>salida.error
rm noexiste 2>>salida.error
cat salida.error
```

reúne los mensajes de error en el archivo salida.error.

5.12. Interconexión de comandos (entubamiento).

El operador | (“pipe”) hace que la salida del comando precedente sea la entrada del comando siguiente, creando un entubamiento o *interconexión* de comandos.

```
cat nota LEAME | pr >salida
```

hace que la concatenación de los archivos `nota` y `LEAME` sea servida al comando `pr`, cuya salida está redirigida a un archivo.

```
cat - LEAME <nota | pr >salida
```

tiene el mismo efecto. Prestar particular atención en la posición de la redirección `<`.

Los operadores `<` y `>` redirigen, `|` conecta comandos.

5.13. Filtros.

Muchos comandos están pensados para ser interconectados, pasando simplemente la entrada hacia la salida, por lo que se les llama habitualmente *filtros*. Los más comunes se encuentran descritos en la tabla 5.2.

Filtros	Función
sort	Ordena las líneas de un texto
cut	Corta secciones de una línea
od	Convierte archivos a forma octal u otras
paste	Une líneas de diferentes archivos
tac	Concatena e imprime archivos invertidos
tr	Traduce o borra caracteres
uniq	Remueve líneas repetidas
wc	Cuenta bytes, palabras y líneas

Cuadro 5.2: Algunos Filtros en línea de comandos Linux/Unix

Algunos filtros han llegado a ser tan complejos que son en sí, un lenguaje de procesamiento de texto, de búsqueda de patrones, de construcción de scripts, y muchas otras posibilidades. Estas herramientas pasan a ser parte de la siguiente sección. Entre ellos podemos mencionar herramientas tradicionales en Linux/Unix como `awk` y `sed` y otras más modernas como Perl.

```
echo 'hola chicos' | tr l p
```

`tr` reemplaza caracteres y produce 'hopa chicos'.

```
echo 'hola chicos' | tr lo pa
```

produce 'hapa chicas'.

5.14. Campos y delimitadores.

Un campo es una cadena de caracteres separada por un carácter delimitador. El archivo `/etc/passwd` tiene en cada línea una serie de campos separados por dos puntos (`:`).

```
more /etc/passwd
```

muestra el contenido de `/etc/passwd`.

```
cut -f1 -d: </etc/passwd
```

```
cut -f1,3,5 -d: </etc/passwd
```

muestra los campos pedidos usando el delimitador : (dos puntos).

```
cut -c1-8 </etc/passwd
```

muestra columnas 1 a 8.

```
ls -l | cut -c56-
```

corta el listado de archivos desde donde empieza el nombre al final.

```
sort </etc/passwd | cut -f1 -d: | more
```

ordena las líneas, corta el primer campo y presenta los nombres de usuarios habilitados en el sistema.

```
env | cut -f1 -d= | sort
```

muestra nombres de variables de ambiente ordenadas; el separador es =.

5.15. Valores de retorno de los comandos.

Los comandos devuelven un código de retorno 0 si el comando termina correctamente, o un número entre 1 y 255 según la razón de falla. El código de retorno del último comando queda en una variable llamada '?', que se interroga como \$?.

```
cat noexiste
```

```
ERROR=$?
```

```
echo $?
```

```
echo $ERROR
```

guarda el código de error; la asignación termina bien, por lo que en \$? queda 0, pero en ERROR quedó el número de error de la operación fallida. \$? es una 'variable de shell' mantenida internamente por el propio intérprete. Otras variables de shell son:

número de argumentos en el comando para la shell actual

\$ número de proceso para el shell actual

Estas variables se interrogan como \$# y \$\$.

5.16. El operador grave.

El acento grave (') asigna a una variable de ambiente la salida estándar de un comando. El largo de la variable es limitado, pero usualmente ¿5120 en SVR4.

```
MUESTRA='echo $LOGNAME'
```

```
echo $MUESTRA
```

escribe el nombre de login del usuario.

```
wc /etc/passwd
```

cuenta líneas, palabras y caracteres.

El comando wc (word count) acepta opciones -l para líneas, -w para palabras y -c para caracteres.

```
TOTALPALABRAS='cat * | wc -w'
```

```
echo $TOTALPALABRAS
```

cuenta las palabras en todos los archivos del directorio. También

```
echo 'cat * | wc -w'
```

```
echo Total de palabras en el directorio: 'cat * | wc -w'
```

El acento grave permite ejecutar un comando dentro de otro, encerrando el comando anidado entre acentos graves.

5.17. Secuencias de comandos.

El shell es también un lenguaje de programación. Pueden escribirse varios comandos en una misma línea separándolos con ; (punto y coma). `date ; echo Hola ; echo $LOGNAME MUESTRA='date ; echo Hola ; echo LOGNAME';echoMUESTRA`

5.18. Redirección del shell.

El shell que atiende los comandos del usuario es el login shell; arranca cuando el usuario ingresa al sistema y termina cuando sale. Escribir sh como comando invoca una segunda instancia del shell, que puede terminarse con el comando exit o con *Ctrl-D*.

Puede crearse un archivo de comandos haciendo `cat >datos.usuario`

y escribiendo las siguientes líneas. Pueden omitirse los comentarios (de # en adelante)

```
echo Salida del comando datos.usuario
```

```
echo Fecha: 'date ' # fecha y hora
```

```
echo Usuario: $LOGNAME # nombre de login del usuario
ps # procesos corriendo del usuario
echo Shell actual: $$ # número de proceso para el shell actual
```

finalizar con *Ctrl-D*. Los `textbg#` indican comentario.

Para convertir el archivo en ejecutable para el usuario, hacer

```
chmod u+x datos.usuario
```

Verificar con `ls -l`. Los comandos ingresados en el archivo pueden ejecutarse con `sh datos.usuario`

Este comando invoca una nueva instancia de shell que no es la del ingreso al sistema.

Agregar una línea más haciendo

```
cat datos.usuario - >> masdatos.usuario
```

y escribiendo

```
cat noexiste # intenta mostrar un archivo inexistente
```

Finalizar con *Ctrl-D*.

```
echo ''Archivo de errores del usuario'' >errores.usuario
```

coloca una línea descriptiva en el archivo `errores.usuario`.

```
cat errores.usuario
```

verifica su contenido.

```
sh <masdatos.usuario >salida.usuario 2>>errores.usuario
```

lee los comandos del archivo `masdatos.usuario` en la entrada estándar, redirige la salida estándar al archivo `salida.usuario` y redirige la salida del error estándar al archivo `errores.usuario`.

```
cat salida.usuario
```

```
cat errores.usuario
```

5.19. Comandos en *background*

Linux, como cualquier sistema Unix, puede ejecutar varias tareas al mismo tiempo. En sistemas mono-procesador, se asigna un determinado tiempo a cada tarea de manera que, al usuario, le parece que se ejecutan al mismo tiempo.

Para ejecutar un programa en *background*, basta con poner el signo ampersand (&) al término de la línea de comandos (ver sección ??). Por ejemplo, si se quisiera copiar el directorio `/usr/src/linux` al directorio `/tmp`:

```
#cp -r /usr/src/linux /tmp &
#
```

Cuando ha terminado la ejecución del programa, el sistema lo reporta mediante un mensaje:

```
#
[Done] cp -r /usr/src/linux /tmp
#
```

Si se hubiese ejecutado el programa y no se hubiese puesto el ampersand, se podría pasarlo a background de la siguiente manera:

1. Se suspende la ejecución del programa, pulsando *Ctrl+Z*.
2. Se ejecutamos la siguiente orden: `bg`

5.20. Variables de entorno

Una *variable de entorno* es un nombre asociado a una cadena de caracteres.

Dependiendo de la variable, su utilidad puede ser distinta. Algunas son útiles para no tener que escribir muchas opciones al ejecutar un programa, otras las utiliza el propio shell (*PATH*, *PS1*,...). La tabla 5.3 muestra la lista de variables más usuales.

Variable	Descripción
DISPLAY	Donde aparecen la salidas de X-Windows.
HOME	Directorio personal.
HOSTNAME	Nombre de la máquina.
MAIL	Archivo de correo.
PATH	Lista de directorios donde buscar los programas.
PS1	Prompt.
SHELL	Intérprete de comandos por defecto.
TERM	Tipo de terminal.
USER	Nombre del usuario.

Cuadro 5.3: Variables de entorno más usuales

La forma de definir una variable de entorno cambia con el interprete de comandos, se muestra `tsh` y `bash` siendo los dos mas populares en el ámbito Linux:

bash: `export VARIABLE=Valor`

tsh: `setenv VARIABLE Valor`

Por ejemplo, para definir el valor de la variable `DISPLAY`:

bash: `export DISPLAY=localhost:0.0`

tsh: `setenv DISPLAY localhost:0.0`

5.21. Alias

Un “alias” es un nombre alternativo para un comando. Así, en lugar de escribir el comando propiamente dicho, escribiríamos el *alias* de dicho comando.

Un *alias* se puede definir por varios motivos, por ejemplo:

- Dar nombres familiares a comandos comunes:
`alias md='mkdir'`
Crearía un alias para el comando *mkdir*, similar al de DOS.
- Dar nombres a comandos largos:
`alias tbz2='tar -cv --use-compress-program=bzip2 -f'`
Crearía un alias para el comando *tar* para que use el compresor *bzip2* en lugar de *gzip*.

Para no tener que escribir todos los alias siempre que entremos al sistema, escribiríamos dicho alias en el archivo `/.bash_profile` (ver sección 5.23).

5.22. Reutilización de comandos

El shell almacena una historia de los comandos que el usuario ha escrito. Por medio de esta historia es posible volver a ejecutar una orden que ya se ha escrito anteriormente sin tener que escribirla de nuevo.

El comando `history` muestra la secuencia de comandos, con un número a su izquierda. Con este número es posible llamar de nuevo el comando utilizando el carácter admiración “!”; Por ejemplo `history` retorna

```
1 history
2 ls
3 cd public_html
4 ls
5 rm *.bak
6 history
```

y para ejecutar nuevamente el comando `rm *.bak` solo es necesario escribir `!5`. También se puede pedir el último “`rm`” que se ha ejecutado escribiendo `!rm`.

El último comando se repite con doble admiración “!!”. Es posible también editar el último comando utilizando el carácter “^” pero este conocimiento se está volviendo poco útil, ya que los nuevos shells permiten viajar por la “historia” y editar los comandos usando únicamente las flechas del teclado.

5.23. Archivos de *bash*

Cada shell posee ciertos archivos donde mantiene su configuración. Estos tienen una jerarquía que va desde el archivo general de configuración del sistema para todos los shells, pasando por el archivo propio del shell, hasta los archivos personales del usuario.

Archivo	Descripción
/bin/bash	Ejecutable <i>bash</i> .
/etc/profile	Archivo de inicialización utilizado por los shells.
~.bash_profile	Archivo(s) de inicialización personal
~.profile	utilizado por los shells
~.bash_login	Ejecuta cuando entra al shell
~.bash_logout	Ejecuta cuando sale del shell
~.bashrc	Archivo personal de inicialización del shell.
~.inputrc	Archivo de inicialización individual.

Cuadro 5.4: Archivos de *bash*

5.24. El Sistema de Archivos

5.24.1. Definiciones

Entenderemos por *archivo* a un conjunto de caracteres o *bytes*. El sistema de archivos no impone ninguna estructura sobre el archivo y su contenido no tiene ningún significado para el; el significado depende exclusivamente de los programas que lo manipulan.

La estructura del sistema de archivos es jerárquica, es decir una gráfica dirigida o, vista de otro modo, una estructura arbórea.

El directorio principal, llamado raíz, tiene por nombre simplemente `‘/’` que a su vez es el caracter utilizado para separar los nombres de los subsiguientes directorios.

Por ejemplo, el directorio de trabajo de un usuario determinado puede ser: `/home/paco` y el archivo donde se encuentran los comandos de arranque está en: `/etc/init.d`.

Al ser GNU/Linux un sistema operativo multiusuario, tiene también un modo de protección al sistema de archivos, previsto para la privacidad entre usuarios. Este mecanismo esta formado por tres tipos de accesos: de usuario, de grupo y de “otros”. Esto quiere decir que el usuario fija los accesos para él, para las personas en su grupo y para el resto de los usuarios. En cada caso se tiene un permiso independiente para lectura, escritura y ejecución.

Varios usuarios pueden pertenecer al mismo grupo y un usuario puede, a su vez, pertenecer a varios grupos. Es decir, todos los usuarios del departamento de contabilidad pueden estar en el grupo `conta`², los de ingeniería en el grupo `inges` y a su vez, los jefes de ambos departamentos pueden estar en el grupo `jefes` además de en los antes mencionados.

El fin de esta caracterización de los usuarios es para permitir y restringir accesos a determinadas partes del sistema. En el ejemplo anterior, es obvio que ninguno de los usuarios, aparte de los del departamento de contabilidad, pueden tener acceso a los archivos donde se encuentra la nómina. A su vez, habrá directorios donde sólo los jefes de departamento podrán tener acceso y que nadie más (salvo el gerente y el administrador de la máquina) podrán leer y modificar los archivos contenidos.

²En realidad no hay ninguna razón para utilizar abreviaturas.

5.24.2. El sistema de archivos en GNU/Linux

Un sistema de archivos en GNU/Linux puede contener miles de archivos, cientos de directorios y cientos de enlaces simbólicos, dependiendo de la distribución y de lo que se haya instalado. Probemos ahora el siguiente comando:

```
$ ls -F /
```

es útil para recorrer el sistema de archivos, bajando luego a los subdirectorios.

En la tabla 5.5, se describen los directorios más comunes en un sistema GNU/Linux.³

5.24.3. Implantación del sistema de archivos

Esta sección pretende describir la forma de implantación del sistema tradicional de archivos en GNU/Linux.

Todos los discos que contienen sistemas de archivos en GNU/Linux tienen la organización que se muestra en la figura ???. El bloque 0 no es utilizado por GNU/Linux y contiene a menudo código para el arranque de la computadora. El bloque 1 que es el **superbloque**, contiene información crítica relativa a la organización del sistema de archivos:

- número de i-nodos
- número de bloques en disco
- inicio de la lista de bloques libres en disco.

La destrucción del superbloque provocará que el sistema de archivos quede inservible.

Después del superbloque están los **nodos-i** (abreviatura de nodos-índice, aunque nunca se les llama de esta forma). Se enumera de 1 hasta cierto máximo. Cada i-nodo tiene una longitud de 64 bytes y describe a un archivo. Un i-nodo contiene la información contable (propietario, bit's de protección, etc) así como la información suficiente para localizar todos los bloques del disco que contengan los datos del archivos.

Después de los nodos-i están los bloques de datos. Todos los archivos y directorios se almacenan aquí. Si un archivo o directorio consta de más de un bloque, los bloques no tienen por qué ser adyacentes en el disco. De hecho, es probable que los bloques de un archivo grande estén diseminados por todo el disco. Las mejoras de Berkeley se diseñaron para reducir esta dispersión.

Un directorio en el sistema tradicional de archivos consta de una colección no ordenada de entradas de 16 bytes. Cada entrada contiene un nombre de archivo (de hasta 14 caracteres arbitrarios) y el número del nodo-i del archivo. Para abrir un archivo en el directorio de trabajo, el sistema sólo lee el directorio, compara el nombre por buscar con cada entrada, hasta que encuentra el nombre o concluye que no está presente. Si el archivo está presente, el sistema extrae el número de nodo-i y lo utiliza

³El proyecto Filesystem Hierarchy Standard es el encargado de normar el nombre y contenido de los directorios, para mayor información consultar <http://www.pathname.com/fhs/>

Directorio	Descripción
/bin	binarios del sistema de uso frecuente
/boot	contiene archivos estáticos requeridos para arrancar el sistema, tales como el kernel de Linux. Estos archivos son esenciales para que el sistema arranque correctamente.
/dev	contiene entradas del sistema de archivos que representan dispositivos del sistema. Estos archivos son esenciales para el correcto funcionamiento del sistema. [subdirectorios propios de System V]
./dsk	dispositivos de disco
./fd	dispositivos descriptores de archivo
./kd	dispositivos de teclado y despliegue
./kmem	memoria
./null	dispositivo para descarte de salidas
./osm	mensajes de error del kernel
./pts	pseudo ttys; igual que /dev/pts*
./rdsk	dispositivos crudos de disco
./term	terminales; igual que /dev/tty*
./xt	pseudo ttys; para capas DMD
/etc	configuración de paquetes, configuración de sistema
./init.d	scripts de arranque y detención de programas
./rc?.d	enlaces a scripts, con K o S (Kill o Start), y número de secuencia para controlar el arranque
./skel	archivos de inicialización para nuevos usuarios
/export	directorios de usuarios en sistemas grandes
/home	objetos relacionados con los usuarios
/lib	bibliotecas de desarrollo y material de apoyo
/lost+found	archivos perdidos
/mnt	punto de montaje de dispositivos externos
/proc	contiene "archivos" especiales que o bien extraen información del kernel o bien la envían a éste.
/root	directorio propio para el superusuario (root)
/sbin	archivos ejecutables de administración
/tmp	archivos temporales
/usr	ejecutables, documentación, referencia
./X11R6	sistema X-Windows
./bin	más ejecutables
./doc	documentos de paquetes de software
./include	encabezados .h de bibliotecas en C
./info	archivos de info, información de Linux
./lib	más bibliotecas en C
./local	ejecutables instalados por el administrador
./man	subdirectorios de páginas del manual
./sbin	más archivos ejecutables de administración
./share	compartidos
./src	(source) código fuente del kernel
/var	archivos de log, auxiliares, archivos que crecen
./backup	respaldo de algunos archivos del sistema
./catman	páginas man ya formateadas
./lib	información propia de programas
./lock	control de bloqueos
./log	archivos de registro de mensajes (log) del sistema
./spool	colas de impresión, intermedios de correo y otros
./run	información de procesos (PIDs)

Cuadro 5.5: Estructura de directorios en GNU/Linux

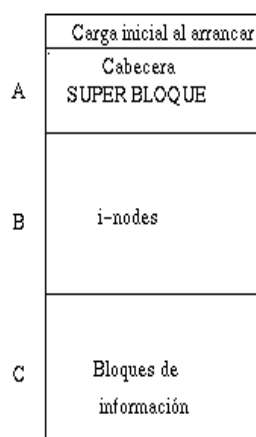


Figura 5.2: Organización del disco en los sistemas.

como un índice en la tabla nodos-i (en el disco) para localizar el nodo-i correspondiente y traerlo a la memoria. El nodo-i se coloca en la tabla de nodos-i, una estructura de datos en el kernel que conserva todos los nodos-i de los archivos y directorios abiertos en cada momento.

La búsqueda de un nombre absoluto de ruta de acceso como */home/paco/fundamentosSO.pdf* es un poco más compleja. En primer lugar, el sistema localiza el directorio */* (raíz), que siempre utiliza el nodo-i 2 (el nodo-i 1 se reserva para el manejo de los bloques defectuosos). Entonces busca la cadena "home" en el directorio */*, para obtener el número de nodo-i del directorio */home*. Se busca entonces este nodo-i y se extraen los bloques del disco de él, de forma que se pueda leer el directorio y buscar la cadena "paco". Al encontrar esta entrada se puede tomar el número de nodo-i para el directorio */home/paco*. Con el número de nodo-i del directorio */home/paco*, se puede leer este nodo-i y localizar los bloques del directorio. Por último se busca *fundamentosSO.pdf* y se encuentra su número de nodo-i. Así, el uso de un nombre relativo de una ruta de acceso no solo es más conveniente para el usuario, si no que también ahorra una cantidad sustancial de trabajo al propio sistema.

5.24.4. Manejo del sistema de archivos

El comando utilizado para visualizar el contenido de un directorio es `ls` con una serie de opciones. Las opciones a programas y comandos en GNU/Linux típicamente van precedidas por el carácter `-`, salvo cuando se indique lo contrario. Las opciones sirven generalmente para cambiar el comportamiento de un comando. Por ejemplo para ver el contenido de un directorio en forma somera, basta dar `ls`:

```
[paco@titan diploDB]$ ls
1erExamenSO.aux      fundamentosSO.pdf  practica2.pdf
1erExamenSO.dvi     fundamentosSO.tex  practica2.tex
```

```

1erExamenSO.log      fundamentosSO.tgz   practica3.aux
1erExamenSO.tex      fundamentosSO.toc   practica3.dvi
acct.eps             instalacion.tex     practica3.log
acct.png             intro.tex           practica3.pdf
acentos.sh           keyboard.eps        practica3.tex
admon.tex            lang.eps            practica6.aux

```

o en forma mas completa `ls -al`, donde la opción `a` indica que queremos todos los archivos (se puede “ocultar” un archivo a `ls`) y la opción `l` indica que queremos la versión completa o “larga” de la salida:

```

[paco@titan diploDB]$ ls -la
total 37836
drwxrwxr-x   4 paco paco   4096 dic  1 19:07 .
drwxr-xr-x  48 paco paco   4096 dic  1 19:01 ..
-rw-rw-r--   1 paco paco    265 dic  1 11:54 1erExamenSO.aux
-rw-rw-r--   1 paco paco  16004 dic  1 11:54 1erExamenSO.dvi
-rw-rw-r--   1 paco paco   5702 dic  1 11:54 1erExamenSO.log
-rw-r--r--   1 paco paco  13989 nov 29 20:27 1erExamenSO.tex
-rw-rw-r--   1 paco paco 2925744 nov 30 11:27 acct.eps
-rw-rw-r--   1 paco paco   49281 nov 30 11:27 acct.png
drwxrwxr-x   2 paco paco   4096 dic  1 15:03 uxo
-rw-rw-r--   1 paco paco   3670 dic  1 18:39 uxo.aux
-rw-rw-r--   1 paco paco  112080 dic  1 18:24 uxo.tex
-rw-rw-r--   1 paco paco   1946 dic  1 18:39 uxo.toc
-rw-rw-r--   1 paco paco 2925753 nov 29 20:28 x86bootloader.eps

```

Las primeras diez columnas indican el tipo de acceso que tiene cada archivo para los diferentes usuarios. Así, el archivo con nombre `1erExamenSO.tex` tiene los accesos que se indican en la tabla 5.6

```

          1  1 1    1    1    1 1    1
1234567890  1 2  3    4    5    6 7    8
-rw-r--r--   1 paco paco   2495 dic  1 19:01 fundamentosSO.tex

```

Cuadro 5.6: Accesos a un archivo. Cada columna está numerada, con los siguientes significados: 1) No es un archivo “especial” 2) El dueño (paco) puede leerlo 3) El dueño puede escribirlo 4) El dueño no puede ejecutarlo 5) Los del grupo (users) pueden leerlo 6) Los del grupo no pueden escribirlo 7) Los del grupo no pueden ejecutarlo 8) Los otros pueden leerlo 9) Los otros no pueden escribirlo 10) Los otros no pueden ejecutarlo 11) Número de ligas 12) Usuario 13) Grupo 14) Tamaño en bytes 15) y 16) Fecha de alteración 17) Hora de alteración 18) Nombre del archivo.

En el caso de los archivos ejecutables, aparece una `x` para denotarlos. Los caracteres para los archivos “especiales” son: `-` normales, `d` directorios, `l` liga simbólica, `b` dispositivo por bloques, `c` dispositivo por caracteres y `p` *pipe*⁴.

⁴Como veremos más adelante, la sintaxis del sistema de archivos de GNU/Linux permite crear canales de comunicación entre procesos, como si éstos se trataran de archivos.

Para cambiar los atributos de un archivo contamos con la instrucción `chmod`, que nos permite cambiar el modo de acceso del archivo. La sintaxis es:

```
chmod quienes operacion modo
```

donde *quienes* puede ser una combinación de `ugo` con `u` para fijar el modo para el usuario, `g` para el grupo y `o` para los otros, es decir, para los que no son el usuario y que tampoco pertenecen al grupo. La *operacion* es `+` para añadir y `-` para retirar el modo a los grupos seleccionados y *modo* puede ser cualquier combinación de `rxw` con el significado dado en la tabla 5.6. Por ejemplo: si el usuario `paco` quiere que un archivo en particular, digamos que se llama `diario`, no sea leído por nadie en el sistema salvo por él y por `root`⁵, tendría que usar: `chmod go-rw diario`. Así que si ahora examina los accesos del archivo con `ls`, estos serán:

```
-rw----- 1 paco usuarios 1789 Jun 13 01:54 diario
```

Lo mismo en caso de querer ocultar un directorio, por ejemplo el directorio `cursilerias`: `chmod go-rwx cursilerias` de tal manera que los demás usuarios del sistema ni siquiera podrán ver el contenido del directorio.

Ahora pensemos que `paco` es un gran programador⁶ y en sus ratos libres se dedica, además de escribir cartas, a escribir programas que desea compartir con los demás usuarios del sistema. Para esto, crea un subdirectorio donde los depositaría y además los dejará con los permisos adecuados para que todo mundo los pueda ejecutar y examinar. Digamos que decide que el directorio se llamará `bin`⁷, y ahí deposita todos los programas que hasta ahora ha hecho. La secuencia completa es la siguiente:

```
1:$ cd
2:$ mkdir bin
3:$ chmod u=rwx bin
4:$ chmod go=rx bin
5:$ cd bin
6:$ cp ../fuentes/* .
7:$ chmod u=rwx *
8:$ chmod go=rx *
```

El comando `ls` es de los que más opciones tiene. Conocerlas todas es demasiado pedir pese a que probablemente sea de los comandos más usados. ¿Qué hacer entonces? Bueno, el segundo comando más empleado en GNU/Linux es `man`, que es el comando que muestra la ayuda de otros comandos.

Las opciones más útiles son `-k` que lista las páginas de manual de los comandos referentes a la siguiente palabra que se dá, por ejemplo:

⁵`root` es el usuario con todos los privilegios en el sistema. También es llamado el *superusuario*, que es típicamente el administrador y además no se le puede esconder nada.

⁶En la mitología GNU/Linux se les conoce como *hackers*, pese a que el término últimamente a tendido a ser confundido con el de *crackers*, que se refiere a los *hackers* que se dedican a violar la integridad de los sistemas, pero cabe aclarar que no todos los *hackers* son *crackers* ni todos los *crackers* son verdaderos *hackers*.

⁷Este nombre de ninguna manera es fortuito, en GNU/Linux los programas que ejecutan los usuarios están en los directorios `/bin` y `/usr/bin`, donde *bin* es el acrónimo de binarios.

```
$ man -k change
chdir (2)          - Change working directory
chmod (1)         - Change the access permissions of files
chown (1)        - Change the user and group ownership of files
passwd (1)       - Change password
```

lista todos los comandos que contienen la palabra *change*. La otra opción más usada, sirve como una referencia rápida para saber que acción ejecuta un comando en particular. Es la opción `-f`. Por ejemplo:

```
$ man -f chdir
chdir (2)          - Change working directory
```

nos indica que `chdir` es el comando para movernos entre directorios.

Ahora ya sabemos como interpretar la salida de `ls`, como invocar la ayuda de un comando en particular, como buscar ayuda acerca de un tópico y como averiguar que hace un comando en particular. Estamos listos para comernos el mundo de GNU/Linux. O casi.

Para finalizar ésta sección hablaremos de algunos comandos para manipular archivos y directorios.

En caso de que quisiéramos ver el contenido de un archivo podríamos usar el comando `cat`.

Listemos el contenido de otro archivo. uno común a todos los GNU/Linux, el directorio `/etc` y el archivo `passwd`, que es donde el Sistema Operativo coteja a los usuarios válidos en el sistema:

```
[paco@titan ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpm:x:37:37:./var/lib/rpm:/sbin/nologin
haldaemon:x:68:68:HAL daemon:./sbin/nologin
netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
```

```
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
mailnull:x:47:47:/:/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:/:/var/spool/mqueue:/sbin/nologin
pcap:x:77:77:/:/var/arpwatch:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
ntp:x:38:38:/:/etc/ntp:/sbin/nologin
gdm:x:42:42:/:/var/gdm:/sbin/nologin
paco:x:500:500:Francisco Medina Lopez:/home/paco:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
snort:x:501:501:/:/home/snort:/bin/bash
ldap:x:55:55:LDAP User:/var/lib/ldap:/bin/false
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin
```

Pues resultó que este archivo tiene más líneas de las que caben en la pantalla. Para poder visualizarlo pantalla por pantalla podemos usar un paginador, `more`, con el comando `cat /etc/passwd | more`. Si ejecutamos este comando veremos que tenemos manera de detener la salida hasta que demos un espacio y cambia de pantalla, ¿pero qué fué eso de `| more`? Bueno, pues simplemente le estamos pidiendo al sistema operativo que la salida producida por el comando `cat` se la pase como entrada al comando `more`. Esta técnica se conoce como *piping* o “entubado”. En la introducción hablábamos de que el diseño de GNU/Linux tenía en mente el reunir una colección de pequeños programas muy generales que nos permitan unirlos o conectarlos para realizar tareas más complejas. Este es el primer ejemplo al respecto, más adelante veremos construcciones complejas e interesantes.

Claro que si el usuario avezado lee la página del manual de `more`, podrá ver que no hace falta pasar por `cat` y el `pipe`, sino que directamente podemos usar `more /etc/passwd`, pero el ejemplo está basado a propósito en la manera GNU/Linux de hacer las cosas.

Los usuarios con experiencia en MS-DOS dirían que esto no es exclusivo de GNU/Linux, pero cabe aclarar en este punto que mientras que MS-DOS simula esta comunicación entre procesos utilizando archivos —ejecuta el primer programa guardando en un archivo la salida producida por éste y luego ejecuta el programa `more` sobre el archivo producido— en GNU/Linux esto ocurre en memoria y en tiempo real. Esto es que mientras que en MS-DOS tiene que terminar la ejecución del primer programa (dado que no es multitareas) para poder ejecutar al siguiente programa con la salida del primero, en GNU/Linux, ambos están siendo ejecutados simultáneamente.

5.24.5. Redireccionamientos

Lo que acabamos de ver en la sección anterior se llama redireccionamiento de la salida y aunque este tema no es como tal parte del sistema de archivos, me permití ponerlo en esta sección porque lo necesito usar ya. Esta es una base muy fuerte de la construcción de herramientas en GNU/Linux, el poder tomar la salida de un programa y dársela como entrada a otro. Veremos más adelante en la sección ?? la manera de aprovechar este método, de momento nos basta con conocer los principios bajo los que operan.

En lo que no aprendemos a usar un editor, tenemos una manera rápida de escribir a un archivo utilizando la redirección. Para escribir directamente a un archivo podemos hacer:

```
$ cat > nombre_del_archivo
...
todo lo que queramos escribir sin posibilidad de regresarnos a la
línea anterior
...
^D
```

El carácter `^D`⁸ es el usado para significar fin de entrada.

La redirección `>` significa la salida mándala como entrada al archivo siguiente, no se utiliza cuando lo que le sigue es un comando o programa.

También existe la redirección `<` para tomar la entrada de un archivo. Por ejemplo, en lugar de `cat /etc/passwd` podríamos haber usado `cat < /etc/passwd`, pero esto es innecesario, ya que `cat` sabe como extraer el contenido de un archivo y no es necesario pedirle al sistema operativo que lo haga por él.

Resumiendo, sabemos que con `>` pasamos la salida de un comando a un archivo; con `<` tomamos el contenido de un archivo y se lo pasamos a un comando o programa y con `|` pasamos la salida de un programa a otro programa.

5.24.6. Navegando por el sistema de archivos

Hemos mencionado que el sistema de archivos es jerárquico y que podemos movernos dentro de él. Una de las herramientas indispensables para navegar en el sistema de archivos es `pwd` que nos dice en que lugar de la jerarquía nos encontramos. Por ejemplo, cuando el usuario `paco` entra en sesión, se encuentra inicialmente en su directorio "home", es decir en el directorio asignado a él dentro de la jerarquía de archivos. Si en ese momento ejecuta el comando `pwd` obtendrá:

```
[paco@titan ~]$ pwd
/home/paco
```

Digamos que ahora necesita escribir varias cartas y que éstas estén almacenadas de tal manera que pueda determinar directamente que son y a quién van dirigidas. De principio, bastaría con nombrarlas con el destinatario, pero pensemos que en un futuro la correspondencia aumentará y por cada persona tendrá varias cartas, y que además en su directorio almacenará otro tipo de información. Entonces, una buena solución es que tenga un directorio exclusivamente para almacenar las cartas, y que dentro de éste exista un directorio por cada destinatario. Así la estructura quedaría de ésta manera:

⁸En adelante para denotar los caracteres que se introducen a base de presionar al mismo tiempo la tecla control y otro carácter usaremos `^` y el carácter, así `^D` significa: presiónese simultáneamente control y D.

```
/home
  /paco
    /cartas
      /pedro
      /jefe
      /personal
      /proyectos
```

Para crear un directorio se utiliza la instrucción `mkdir` de la siguiente manera, `mkdir {nombre del directorio}`⁹. Una vez creado, para posicionarnos dentro de él usamos `cd`. Así, la secuencia de instrucciones que paco necesita ejecutar para crear la estructura descrita es:

```
[paco@titan ~]$ mkdir cartas
[paco@titan ~]$ cd cartas
[paco@titan cartas]$ mkdir pedro
[paco@titan cartas]$ mkdir jefe
etc.
```

Para recorrer los directorios o cambiar el directorio actual se utiliza el comando `cd`. Se puede usar en forma relativa o absoluta. En el primer caso basta con dar el nombre de un subdirectorio a partir de donde estamos. Por ejemplo, si estamos en `cartas`, para cambiarnos al directorio `proyectos`, usaremos `cd proyectos`. Una vez ahí, para irnos al directorio `personal` podemos usar `cd ..` y `cd personal`, con lo cual nos regresamos primero al directorio `cartas` y de ahí nos pasamos al directorio `personal`. También es posible usar `cd ../personal`. O incluso, para irnos de cobranzas a nuestro directorio de trabajo, podemos usar `cd ../../` ya que nos estamos moviendo al directorio padre del directorio padre del actual.

En el segundo caso, el de directorios absolutos, nos podemos mover dando la trayectoria completa. Por ejemplo, dentro del directorio `personal` que se encuentra dentro de `cartas` podemos saber nuestra posición en el árbol con `pwd: /home/paco/cartas/personal` y movernos a nuestro directorio de trabajo con `cd /home/paco`.

Existen tres maneras de regresarse al directorio original de trabajo. La primera es usando una variable de ambiente que tiene definido el valor de éste. Se usa de la siguiente manera: `cd $HOME`. La segunda es usando una pseudo variable que también tiene como definición a nuestro directorio de entrada: `cd ~`. La tercera es la más sencilla: simplemente usando `cd` sin ningún argumento, nos regresa a nuestro directorio “home” que es como se le conoce comunmente.

A la trayectoria que se ha de recorrer para llegar hasta un determinado directorio se le conoce como *path*. Así decimos que, en el caso del usuario paco, su *home* está en el *path* `/home/paco`.

Habrán ocasiones en que será necesario eliminar un directorio o un archivo. De hecho, para poder eliminar un directorio, debe de estar vacío o en otras palabras, no contener ningún archivo. Para borrar un archivo existe la instrucción `rm`, que entre sus opciones tiene dos que nos son de interés inmediato, `-i` y `-f`. Con `rm -i {nombre(s) de archivo(s)}` le indicamos que antes de borrar el archivo nos pida confirmación:

⁹En adelante usaremos `{}` para encerrar los argumentos a un programa o instrucción y `[]` cuando estos argumentos sean opcionales

```
[paco@titan ~]$ rm -i borrame
rm: remove 'borrame'? y
```

Con `-f` forzamos el borrado de un archivo aún en situaciones especiales.

Para eliminar un directorio —que debemos de recordar que tiene que estar vacío— existe el comando `rmdir`. Veamos lo que pasa si tratamos de borrar un directorio que no está vacío. Primero examinamos el contenido de un supuesto directorio `reyna` que digamos que existe en una máquina hipotética:

```
[paco@titan ~]$ ls -al reyna
total 29
-rw-rw-rw-  1  paco  users  2243   Mar   16   02:39  carta.txt
-rw-rw-rw-  1  paco  users 24987   Mar   16   02:39  intro.txt
```

Tiene dos archivos, tratemos de borrarlo:

```
[paco@titan ~]$ rmdir reyna/
rmdir: reyna: Directory not empty
```

Ahora lo haremos borrando primero los archivos contenidos en él:

```
[paco@titan ~]$ rm reyna/*
```

Los nombres de los archivos en el directorio `reyna` visto desde su padre, son: `reyna/carta.txt` y `reyna/intro.txt`. Al decirle que borre `reyna/*` le decimos que borre todos los archivos contenidos dentro de `reyna`. El carácter `*` es un comodín que significa cualquier carácter que se repite 0 o más veces. A diferencia de MS-DOS, el carácter `"."` es uno más sin ningún significado particular.

Si ahora examinamos el contenido del directorio `reyna`, vemos que sólo quedan dos directorios `.` y `..`, que siempre existen en todo directorio en GNU/Linux.

```
[paco@titan ~]$ ls -al reyna
total 2
drwxrwxrwx  2  paco  users  1024   Mar   16   02:43  ./
drwxrwxrwx  4  paco  users  1024   Mar   16   02:42  ../
```

Ahora podemos borrar este directorio sin ningún problema:

```
[paco@titan ~]$ rmdir reyna
```

De haber intentado borrar el directorio `reyna/..`

obtendríamos igualmente un error:

```
[paco@titan ~]$ rmdir reyna/..  
rmdir: reyna/..: Directory not empty
```

Le estamos diciendo al sistema operativo que borre el directorio padre de `reyna`, pero éste no está vacío, ya que al menos contiene a `reyna`, así que no lo puede borrar.

Habrán situaciones en que tenemos toda una parte de la estructura de archivos que necesitamos borrar y que sería toda una lata ir borrando directorio a directorio por lo compleja que pueda ser dicha estructura. Digamos que a `paco` se le asigna una nueva función dentro de su organización y que ya no se va a dedicar a escribir cartas. En ese caso decide borrar el contenido del directorio `cartas` y todos sus subdirectorios con todos los archivos que estos contengan. Por lo que hemos visto hasta ahora, `paco` tendría que recorrer los subdirectorios hasta llegar a aquellos que están en la parte más baja de la estructura —es decir, que ya no contienen subdirectorios— y comenzar a borrar desde allí hacia arriba. Afortunadamente existe una manera rápida de conseguir esto, con la opción `-r` de `rm`. ¿Pero no era este un comando para borrar archivos y no directorios? Bueno, a final de cuentas un directorio es un archivo más dentro de la estructura de archivos de GNU/Linux, y por características propias (que están fuera de la intención de éste manual) es más conveniente hacerlo usando `rm` en lugar de `rmdir`.

Así la manera más rápida, y peligrosa, de borrar el directorio `cartas` y su contenido es:

```
[paco@titan ~]$ rm -fr cartas
```

Nótese que no hubo necesidad de especificar las dos opciones por separado, es decir, `rm -f -r`; esto otra característica de los programas de GNU/Linux, cuando se utilizan juntas varias opciones, se pueden aglutinar todas con un sólo `-`.

Dijimos que era la manera más rápida y peligrosa. Lo primero es inmediato, `-f` asegura que en ningún momento `rm` se detendrá a preguntarnos si hace lo que le pedimos que hiciese. Lo segundo es un poco más difícil de explicar. Cuando se borra un archivo en GNU/Linux, se libera el espacio en disco que éste ocupaba y queda disponible para que el sistema operativo lo utilice en la creación de otro archivo. Como GNU/Linux es un sistema operativo multitareas y multiusuarios, mientras nosotros borramos un archivo, puede ocurrir que el sistema operativo u otro usuario estén creando uno nuevo, de tal manera que de inmediato se ocupe el espacio donde estaba nuestro archivo. Esto quiere decir que es muy probable que en cuanto nosotros borremos el archivo, el área de disco que ocupaba sea utilizada para escribir otro. O sea, que una vez borrado quizá nunca más lo volveremos a ver. En la mayoría de los casos, es prácticamente imposible recuperar un archivo una vez que ha sido borrado.

Renombrado de archivos

Ahora bien, si lo que necesitamos es cambiar el nombre de un archivo, existe la instrucción `mv` cuya función real es *mover* un archivo de un lugar a otro. En particular, podemos mover un archivo de un nombre a otro. Esto es, si tenemos un archivo que se llama `clientes.morosos` y lo queremos renombrar como `clientes.hospitalizados`, basta con dar:

```
[paco@titan ~]$ mv clientes.morosos clientes.hospitalizados
```

Si a `mv` se le da una lista de archivos y al final el nombre de un directorio, lo que hará es mover esos archivos al directorio especificado.

También se puede mover toda una estructura de archivos a otro directorio manteniendo dicha estructura.

Ligas a archivos

Ahora supongamos que dos usuarios, Jorge Torres y Rene Miranda, van a trabajar en el mismo proyecto y necesitan editar el mismo conjunto de archivos. Por conveniencia, ambos deberán pertenecer al mismo grupo. Supongamos que el directorio donde se encuentran los archivos con los que deben de trabajar, asumiendo que trabajan se encuentra en el directorio de trabajo del jefe del área, llamado Luis Perez. Sus cuentas se llaman `jorge`, `rene` y `luis` respectivamente y Jorge y Rene pertenecen a los grupos: `usuarios`, `proyectos` y `inges`. El jefe a su vez, pertenece a los grupos: `jefes`, `inges`, `usuarios`, `proyectos` y `confidencial`. Estas tres personas estarán trabajando en un nuevo proyecto que se refiere a un sistema de cómputo. El administrador del sistema, una vez enterado, decide crear el grupo `punte` e incluye a `jorge` y `rene` y `luis` en este grupo, de tal manera que ahora Luis Perez en su directorio de trabajo crea el directorio `proyecto.punte`, le cambia el grupo con `chgrp` a `punte`:

```
cd
cd trabajo
mkdir proyecto.punte
chgrp punte proyecto.punte
chmod u=rwx proyecto.punte
chmod g=rwx proyecto.punte
chmod o=rwx proyecto.punte
```

de tal manera que sólo él y los pertenecientes al grupo `punte` pueden ver y modificar el contenido de dicho directorio. Ahora, cada vez que alguno de ellos cree un nuevo archivo (o directorio) dentro del directorio, deberá cambiarle el grupo a `punte` y tener cuidado que los permisos sean los pertinentes.

Pero, ¿no será demasiada lata que Rene y Jorge tengan que recorrer toda la estructura cada vez que necesitan trabajar en éste directorio? La primera respuesta que nos viene a la mente es que no hay problema, sabemos perfectamente cómo hacerlo. Pero también hay una manera más fácil: las ligas de archivos.

Todo será más fácil si Rene y Jorge dentro de su directorio `trabajo` tienen un subdirectorio llamado `proyecto.punte` que es idéntico permanentemente al que está en el directorio de Luis Perez. Para esto existe la instrucción `ln` que construye ligas de un archivo o directorio a otro.

Como Rene y Jorge tienen acceso al directorio absoluto `/home/pedro/trabajo/proyecto.punte`, basta con que en su respectivo directorio `trabajo` cada uno haga lo siguiente:

```
cd
cd trabajo
ln /home/pedro/trabajo/proyecto.punte proyecto.punte
chgrp punte proyecto.punte
```

y como éste directorio —en el directorio original, `/home/luis/trabajo/proyecto.puente`— ya tiene los permisos adecuados, no necesitarán hacer nada más.

Cómo revisar archivos

Así como tenemos la instrucción `more` para revisar el contenido de un archivo en pantalla de una manera pausada, tenemos las instrucciones `head` y `tail` que permiten examinar nada más el principio y el final de un archivo respectivamente. Ambos pueden llevar como argumento el número de líneas que se desean examinar. Por ejemplo, para examinar las primeras cinco líneas y las últimas tres líneas de un archivo de datos, utilizamos:

```
$ head -5 datos
090145241|121 22:58401|105106122113109109093101001001001002072071
090145242|121 22:46400|109107107105130126124126000000001000050051
090145282|121 22:46480|142143162144168158127148002003003002039045
090145291|121 22:26388|094093098101125122108108002002001001065062
090145292|121 22:24387|099123114118117092086092001003004002070070
$ tail -3 datos
201022211|122 00:11785|055043045000104113120000004006004000095091
201022231|122 00:03368|003003002000071071071000001002002000029025
201022233|122 00:24 0|0050050050001451391470000000000000000028025
```

Ahora supongamos que queremos ver de la línea 496 a la línea 500 del mismo archivo:

```
$ head -500 datos | tail -5
091111024|121 20:40 0|097028000000120041000000001000000000051014
091111032|121 23:37429|115109115115175181173167000002002000049044
091111041|121 23:25680|202197192191236236228221003002006004088085
091111052|121 23:51446|116110102112174178183170001002001001037039
091111062|121 23:34504|128114119134178186178171001001002001064001
```

¿Y cómo sabemos cuántas líneas tiene el archivo? Con la instrucción `wc`, abreviatura de *word counter*. De hecho, `wc` nos da más información que el número de líneas del archivo:

```
$ wc datos
 27588   55941 5231568
```

donde cada columna nos dice el número de líneas, palabras¹⁰ y caracteres. En éste caso sólo nos interesa saber el número de líneas, así que repetimos el ejemplo con el modificador `-l`:

```
$ wc -l datos
 27588
```

De igual manera, si sólo queremos saber el número de palabras o de caracteres utilizaríamos los modificadores `-w` y `-c` respectivamente.

¹⁰En éste caso el número de palabras se refiere a cualquier caracteres alfanuméricos delimitados por signos de puntuación, incluyendo al espacio, tabulador y cambio de línea.

Impresión de archivos

Para imprimir un archivo existe la instrucción `lpr` que además de enviarlo a la impresora, lo *pagina* o separa por páginas y le añade a cada un *encabezado* con el nombre del archivo y número de página. Tiene varias opciones para indicarle a que impresora se envía, cuantas líneas por página permite la impresora o se desean, que debe de ir en la encabezado y otras opciones más que deberán ser examinadas en la página de manual, ya que de de una implementación a otra cambian algunos parámetros. En éste caso, lo más seguro es preguntar al administrador del sistema los nombres de las impresoras y las opciones que permite `lpr` de acuerdo a éstas.

5.24.7. Espacio en disco

En ocasiones es necesario saber cuanto espacio ocupa todo un directorio y cuanto espacio queda libre en un disco. Existen dos instrucciones para esto, `du` y `df`. `du` nos dice el espacio ocupado en kilobytes dentro de un directorio si se ejecuta sin ningún argumento:

```
[paco@titan ~]$ du
9821824 .
7051356 ./vmware
5252336 ./vmware/Windows XP Professional x64 Edition
2209312 ./Centos4.3
1799016 ./vmware/Red Hat Enterprise Linux 4 2
169820 ./evolution
```

En este caso, el directorio `./vmware` ocupa 7051356 kilobytes, `./Centos4.3` 2209312 Kb, `./vmware/Windows XP Professional x64 Edition` 5252336 Kb y todo el directorio actual junto con sus subdirectorios ocupa 9821824. La salida de `du` muestra los nombres de los subdirectorios con el prefijo `./` para indicar que es a partir del directorio actual.

Podemos pedir también el espacio ocupado por un directorio por su nombre absoluto:

```
[paco@titan ~]$ du /etc
41116 /etc/
```

La salida de `df` es un poco más entendible:

```
[paco@titan ~]$ df
S.ficheros Bloques de 1K Usado Dispon Uso% Montado en
/dev/hda3 15116868 14212968 135996 100% /
none 257496 0 257496 0% /dev/shm
/dev/hda2 10238812 7572620 2146088 78% /mnt/hda2
/dev/hda6 25466592 25197552 269040 99% /mnt/hda6
```

La primera columna nos indica el *filesystem* o sistema de archivos, o dispositivo físico, o en términos más comprensibles, el disco. La segunda nos dice el espacio total en el disco en unidades de 1024 bytes,

o 1 kilobyte, aunque las unidades varían de una implementación de GNU/Linux a otra. La tercera y cuarta columnas indican el espacio utilizado y el espacio disponible en las mismas unidades que la segunda. La quinta columna indica el porcentaje ocupado del disco. Por último, la sexta columna indica en que parte del sistema de archivos está *montado* el disco, así que si deseamos saber cuanto espacio queda disponible para el usuario *paco*, debemos considerar en que lugar del sistema de archivos está el directorio donde trabaja. Como está en `/home/paco` en éste caso, sabremos que el espacio que le queda disponible, son 3 839 Kb o 3.7490 Mb, dado que *paco* es un subdirectorio dentro de *home*.

Cabe hacer la aclaración que GNU/Linux provee de mecanismos para acotar el espacio en disco disponible para cada usuario, pero la mayoría de las veces esta restricción no se aplica, con lo cual si hay cinco usuarios en el sistema, estos cinco usuarios compiten por el espacio disponible en el disco donde estan montados sus directorios.

La elección de restringir el espacio en disco a cada usuario está determinada por las políticas de uso que emplee el administrador del sistema.

5.24.8. Montaje

Un sistema de archivos contiene un directorio principal, subdirectorios y archivos soportados en un área de disco. Pueden adjuntarse otros sistemas de archivos al árbol de directorios principal aplicando sobre un directorio del árbol principal el directorio superior del sistema de archivos que se adjunta. El punto del árbol principal donde se “cuelga” el sistema de archivos adjuntado se llama punto de montaje. Este artificio permite disponer de un árbol único de directorios formado por partes en distintos soportes locales o remotos. El comando `mount` habilita la operación de “colgar” un sistema de archivos a un directorio del árbol principal.

```
mount /dev/sda0 /usuarios
```

coloca el sistema de archivos almacenado en el disco `/dev/sda0` bajo el directorio `/usuarios`; para ver el contenido del sistema de archivos en disco, en lo sucesivo se hará `ls /usuarios`.

Un sistema de archivos se desvincula del árbol principal con el comando `umount`:

```
umount /usuarios
```

desmonta el sistema de archivos anterior del directorio `/usuarios`, impidiendo el acceso a los archivos contenidos en el disco. Para poder ejecutar `umount` no debe haber archivos abiertos, ni usuarios en directorios de la rama a desmontar, ni procesos corriendo cuyos ejecutables residan en él.

En la extensión semántica ya indicada, el término sistema de archivos se emplea así para designar la totalidad del árbol de directorios, formado en realidad por partes de diferente organización lógica y naturaleza física.

La lista de archivos montados en un sistema UNIX figura en el archivo `/etc/fstab`, `/etc/vfstab` o `/etc/checklist` según la variedad de UNIX. Esto permite realizar tareas sobre todos los sistemas de archivos, generalmente en el momento del arranque:

```
umount fsck -p
```

verifica la integridad de todos los sistemas de archivos según figuran en `/etc/fstab` (o su equivalente);

mount -a

monta todos los sistemas de archivos en `/etc/fstab` (o su equivalente).

Si el intento de desmontar un sistema de archivos fracasa por encontrarse éste ocupado, el comando `fuser` indicará los números de proceso y los nombres de usuario que lo ocupan. En FreeBSD, `fstat` hace algo similar.

Capítulo 6

Editores

6.1. Editor de pantalla completa `vi`

El editor más frecuente en Unix es `vi`, para el caso particular de GNU/Linux en `vim`, lo mismo pero más fácil. Es un editor que trabaja línea a línea y que muestra una pantalla de texto a la vez.

Para iniciar una sesión de edición, se ejecuta el programa `vi` seguido del nombre del archivo a editar, y dado el caso también la trayectoria. Por ejemplo, para editar el archivo `/tmp/borrarme`, basta con dar `vi /tmp/borrarme`, o para editar un archivo en el directorio actual: `vi borrarme`.

Si el archivo no existe `vi` lo crea. De igual manera, podemos simplemente invocar a `vi`, comenzar a escribir y después nombrar el archivo al momento de guardarlo.

Al ser ejecutado `vi`, presenta una pantalla con el texto del archivo, y las líneas después del final del archivo aparecen con el carácter `~` para indicar que a partir de ahí el archivo está vacío. Obviamente, si comenzamos a editar un archivo nuevo, todas las líneas aparecerán con éste carácter.

Tiene tres modos de trabajo. El modo de *inserción*, el de *edición* y el de *comandos*.

En el modo de inserción, toda la entrada que demos en el teclado se inserta en el archivo en el punto donde se encuentre el cursor. En el modo de edición daremos instrucciones que alteran el contenido, como por ejemplo para posicionarse en determinado punto, hacer reemplazos de texto, copiar o mover bloques de texto, etc. En el modo comandos se dan instrucciones para salvar el archivo, traer a edición otro, insertar otro archivo en el punto donde se está, terminar la edición, etc.

En el modo edición, las instrucciones para mover el cursor en del texto son:

comando	se desplaza:
l	un espacio a la derecha
h	un espacio a la izquierda
j	una línea hacia abajo
k	una línea hacia arriba
\$	al final de la línea
^	al principio de la línea
w	a la siguiente palabra
e	al final de la palabra
b	al principio de la palabra
)	al final de la frase
(al inicio de la frase
{	al inicio del párrafo
}	al final del párrafo
H	a la primera columna de la primera línea de la ventana
L	a la primera columna de la última línea de la ventana
nG	a la primera columna de la n-ésima línea del archivo

Para el control de la parte del texto que se despliega en la pantalla:

instrucción	acción:
^d	desliza el texto hacia arriba
^u	desliza el texto hacia abajo
^f	despliega la ventana de texto siguiente
^b	despliega la ventana de texto anterior
^l	redespliega el texto en la ventana actual

Las instrucciones para borrar texto:

dw	suprime la palabra donde está el cursor
dd	suprimir la línea donde está el cursor
D	borra el texto entre el cursor y el fin de la línea
x	borra el carácter sobre el que está el cursor

Para hacer reemplazos de texto:

cw	cambiar la palabra actual
cc	cambiar la línea actual
C	cambiar desde el cursor hasta el final de la línea
r	cambiar el carácter sobre el que está el cursor

Algunas instrucciones suplementarias:

u	anular la última instrucción dada
/	realiza una búsqueda hacia adelante
?	realiza una búsqueda hacia atrás
n	busca la siguiente ocurrencia de la última búsqueda
.	repite la última instrucción
Y	extrae la línea
p	se coloca en la línea de abajo
P	se coloca en la línea de arriba
ZZ	salva el archivo y termina la edición
ESC	cancela una orden
:	se cambia a modo comandos

Cuando vi esta en modo de edición, para cambiarse a modo inserción se hace con el caracter i y se posiciona antes del cursor y con a después de éste. Con ESC se cambia a modo edición de nuevo.

En modo de comandos, se tienen las siguientes funciones:

:w	salva el archivo en el disco
:q	abandona la edición sin guardar los cambios
:wq	escribe y termina
:q!	abandona sin escribir cuando se realizó algún cambio.
:r	carga otro archivo
:e	edita el archivo
:f	cambia o dá nombre al archivo actual
:n	se posiciona en la n-ésima línea
:Esc	se pasa a modo de edición

vi tiene muchas más instrucciones y es capaz de realizar tareas muy complejas. Es importante conocer la mayoría de ellas para hacer más eficiente nuestro trabajo, pero son demasiado extensas para incluirlas todas en este curso. Existen libros especializados donde se describe con toda sobriedad esta poderosa herramienta.

Existen además de vi otros editores para Unix, incluso más poderosos que él, pero esto depende de cada implementación de Unix. vi y ex (que por ser algo más limitado, no trataremos aquí) son los únicos que se garantiza que se pueden encontrar en cualquier instalación de Unix. Entre los editores más populares se encuentra Emacs, pero cuenta con un conjunto de instrucciones realmente complejo, lo cual forma parte de su tradición de ser para usuarios avanzados.

Haremos ahora un ejemplo para familiarizarnos con vi. Comenzaremos editando un archivo nuevo que se llamará borrego.txt. Invocamos a vi con este argumento,

```
$vi borrego.txt
```

ahora la pantalla se limpia y aparecen varios renglones con tildes en el margen izquierdo y en la parte inferior izquierda aparece el mensaje "borrego" [New file], que indica que se esta creando un archivo nuevo.


```
a
Artifex vitae, artifex sui.
```

```
Muy cerca de mi ocaso, yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
```

```
Porque veo al final de mi rudo camino
que yo fui el arquitecto de mi propio destino;
que si extraje las mieles o la hiel de las cosas,
fue porque en ellas puse hiel o mieles sabrosas;
cuando planté rosales coseché siempre rosas.
```

```
.
w en_paz
389
q
$
```

Después de invocar a `ed` inmediatamente le pedimos que añada el texto con el comando `a` y comenzamos a guardar un fragmento del texto de un pequeño poema. Terminamos poniendo un `.` como único carácter en la línea y después le pedimos que salve el archivo con el nombre de `en_paz`. `ed` nos responde con el número de caracteres que contiene, y terminamos la sesión de edición con una `q`.

Si quisieramos añadir unas cuantas líneas más, podríamos hacerlo con:

```
$ ed en_paz
389
a
. . . Cierto, a mis lozanías va a seguir el invierno;
mas tú no me dijiste que mayo fuese eterno!
.
q
?
w
489
q
$
```

En esta ocasión, invocamos a `ed` directamente con el nombre de un archivo que ya existe, así que nos responde con el número de caracteres que contiene el archivo. Comenzamos a añadir unas líneas, nos detenemos y tratamos de salirnos. Esta vez `ed` nos avisa con `?` que el archivo no ha sido salvado. No debemos esperar una comunicación más comprensible de `ed`. Su manera de avisarnos que algo no le gusta o que las cosas no van bien es con un representativo `?`. En este caso una segunda `q` le indicaría que realmente nos queremos salir sin salvar los cambios que hicimos. En todo momento una `Q` le indica a `ed` que deseamos terminar la sesión sin guardar los cambios.

Cuando queremos revisar el texto ya escrito, le podemos pedir que imprima las líneas en un cierto rango con la instrucción `a, bp` donde `a` y `b` son números de línea. Por ejemplo, para ver el segundo párrafo:

```

$ ed angel
489
3,6p
Muy cerca de mi ocaso, yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
.,$p
Porque veo al final de mi rudo camino
que yo fui el arquitecto de mi propio destino;
que si extraje las mieles o la hiel de las cosas,
fue porque en ellas puse hiel o mieles sabrosas;
cuando planté rosales coseché siempre rosas.
. . . Cierto, a mis lozanías va a seguir el invierno;
mas tú no me dijiste que mayo fuese eterno!
q
$

```

Los caracteres . y \$ tienen el significado especial de ser la línea actual y la última línea respectivamente como podemos ver en la segunda parte del ejemplo.

Si damos un enter por sí sólo o un - seguido de un enter, nos lista la siguiente línea y la anterior respectivamente, moviéndose hacia adelante y hacia atrás en el archivo. No permite moverse más allá de la primera y de la última línea, ni imprimirlas en orden inverso.

Podemos hacer búsquedas utilizando los operadores /patron/ y ?patron? hacia adelante y hacia atrás respectivamente. Una vez que encontramos el primero, podemos traer el siguiente con las formas // y ??.

```

$ ed angel
489
/fue/
fue porque en ellas puse hiel o mieles sabrosas;
//
mas tú no me dijiste que mayo fuese eterno!
??
mas tú no me dijiste que mayo fuese eterno!
q
$

```

Y podemos usar una búsqueda como parte de un rango:

```

1,/inmerecida/p
Artifex vitae, artifex sui.

```

```

Muy cerca de mi ocaso, yo te bendigo, Vida,
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
.-1,.+3p

```

```
porque nunca me diste ni esperanza fallida
ni trabajos injustos ni pena inmerecida;
```

```
Porque veo al final de mi rudo camino
que yo fui el arquitecto de mi propio destino;
```

Y como vemos, también podemos usar el `.` como ancla e imprimir de la línea anterior a tres más adelante de la actual.

La forma genérica de los comandos es número de línea o rango seguido del comando. Para añadir a partir de una línea a partir de línea se usa `#a`, para insertar antes de la línea `#i`. Para borrar el rango de líneas de la *i*-ésima a la *j*-ésima `i, jd` y `i, jc` para reemplazar el rango de líneas por las que se dan a continuación.

Para hacer reemplazos en base a patrones se utiliza `s/viejo/nuevo/`:

```
3p
Muy cerca de mi ocaso, yo te bendigo, Vida,
3s/ocaso/fin/
3p
Muy cerca de mi fin, yo te bendigo, Vida,
```

Después del cambio se puede usar el modificador `g` para que haga el reemplazo en todas las ocurrencias de la línea o en todas las ocurrencias en el rango de las líneas: `s/viejo/nuevo/g`. Por supuesto que la sintaxis es extensible a rangos delimitados por patrones de búsqueda: `4,9s/viejo/nuevo/`, `1,$s/viejo/nuevo/`, `1,/viejo/s/viejo/nuevo/`, etc.

Cuando necesitamos incluir el patrón viejo dentro del nuevo, no necesitamos retectarlo, el carácter `&` toma el valor:

```
3p
Muy cerca de mi ocaso, yo te bendigo, Vida,
3s/Vida/& mia/p
Muy cerca de mi ocaso, yo te bendigo, Vida mia,
```

Y como vemos, la `p` después del comando de reemplazo imprime la línea.

Para copiar y mover bloques, tenemos las instrucciones `i, jtk i, jmk` donde el primero copia las líneas en el rango de la *i*-ésima a la *j*-ésima después de la línea *k*-ésima. Igual en el segundo caso, pero borrándolas de su posición original.

También podemos incluir un archivo en el actual con la instrucción `r` en la forma `nr archivo` con la cual lo insertamos en el actual a partir de la línea *n*-ésima.

Con `i, jw archivo` copiamos de la línea *i*-ésima a la *j*-ésima en el archivo denominado. Con `i, jW archivo` copiamos de la línea *i*-ésima a la *j*-ésima *al final* del archivo denominado.

Capítulo 7

Introducción a la Administración de GNU/Linux

7.1. Objetivo del administrador de sistemas

El objetivo principal del administrador de sistemas consiste en “proporcionar y mantener acceso a los recursos del sistema.”

Independientemente de la plataforma informática de que se trate, todos los sistemas operativos proporcionan mecanismos para manipular recursos.

Entre estos recursos se encuentran los archivos, las aplicaciones, los periféricos, el ancho de banda, los ciclos de procesador, la memoria y el espacio de almacenamiento.

7.2. ¿Porqué necesitan los sistemas ser administrados?

Cualquiera persona puede, si quiere, ejecutar aplicaciones en un sistema UNIX, pero no cualquiera lo instala y lo mantiene funcionando.

Para este ultimo punto es necesario de una persona capacitada que normalmente se le conoce como administrador de sistemas.

Dado que las computadoras son dispositivos de propósito general, y casi todos los sistemas operativos y el software está escrito para su uso general, es labor del administrador adaptar estas herramientas a las necesidades específicas de los usuarios. Estas necesidades cambian con el tiempo y son distintas para los diferentes usuarios.

7.3. Tareas del Administrador del Sistema

- *Mantenimiento de los usuarios del sistema.* Los procesos de agregado y eliminación de cuentas de usuario pueden ser en parte automatizados mediante scripts (programas en UNIX u otro lenguaje a nivel de sistema operativo), ya sea en modo gráfico o de caracteres, pero siempre debe tomarse algunas decisiones. Al crear una cuenta de usuario es preciso determinar la incorporación a grupos de trabajo, lugar del directorio propio, máquinas habilitadas, alias de correo electrónico de interés para el usuario. Al eliminar una cuenta de usuario los archivos creados por ese usuario deben ser eliminados, respaldados en cinta o transferidos a otro usuario, conservando la información de interés y la responsabilidad individual sobre la misma, así como los recursos del sistema.
- *Agregar y quitar hardware.* Al adquirir o transferir de un equipo a otro una parte de hardware puede ser preciso actuar sobre el sistema para asegurar el reconocimiento y uso del equipo. La tarea puede consistir en enchufar una impresora, abrir la máquina para instalar un disco, una tarjeta controladora u otro dispositivo, correr un script para reconocer o inicializar el nuevo hardware, o editar complejos archivos de configuración.
- *Realizar respaldos de información.* Una tarea esencial frecuentemente olvidada o realizada con descuido. Es responsabilidad del administrador verificar la realización regular de respaldos, la identificación clara de los medios utilizados, la eficacia de la restauración.
- *Instalación de nuevo software.* El software nuevo debe ser instalado, configurado y comprobado su funcionamiento. Luego, los usuarios deben ser informados de su existencia, ubicación y particularidades de uso. El software no perteneciente al sistema operativo debe instalarse en un lugar aparte, para asegurar que las actualizaciones del sistema operativo no lo sobrescriban.
- *Monitoreo del sistema.* Incluye verificar funcionamiento de correo, servidor web y de noticias, mirar los archivos de registro de eventos y errores (log) para detectar o anticipar fallas, asegurar acceso a todas las subredes, controlar recursos del sistema tales como espacio en disco.
- *Detección y reparación de fallas.* Ante fallas de hardware o software, el administrador del sistema es responsable del diagnóstico, así como de contactar el servicio técnico, realizar la reparación por sí mismo o delegar en su personal, supervisando en todos los casos la tarea.
- *Mantenimiento de documentación local.* El uso del sistema lo va cambiando respecto a la instalación primaria. Todos los aspectos en los que el sistema vaya siendo alterado o ampliado deben quedar documentados. La documentación abarca particularidades de configuración del sistema operativo, paquetes de software incorporados, tendido de cables, registros de mantenimiento del hardware, estado de respaldos, procedimientos y políticas de administración.
- *Seguridad.* El administrador del sistema debe implementar una política de seguridad y verificar periódicamente que la seguridad de sus sistemas no ha sido violada. Según el tamaño de los sistemas y su conectividad, las precauciones de seguridad puede incluir simples limitaciones de acceso hasta elaborados procedimientos de captura y auditoría. La seguridad es cada vez más crucial y demandante de recursos.
- *Ayuda a los usuarios.* Aunque pocas veces figura entre las tareas del administrador de sistema, el apoyo a usuarios es muy frecuente e insume mucho tiempo. Es conveniente disponer de procedimientos y ayudas escritos para poner límite al tiempo destinado a estas tareas.

7.4. Responsabilidad del administrador del sistema.

Una o más personas deben ser responsables de cada máquina conectada a la Internet. Esta persona DEBE tener la autoridad, acceso y herramientas necesarias para configurar, operar y controlar el acceso al sistema. Para máquinas esenciales en tiempo compartido, servidores primarios de dominio y relays de correo o gateways, la o las personas responsables DEBEN encontrarse accesibles vía telefónica 24 horas al día, 7 días a la semana¹.

Para máquinas en tiempo compartido de menos importancia o computadores personales o estaciones de trabajo de un solo usuario, el responsable individual DEBE estar preparado para una posible intervención en su máquina por parte del administrador de red, si la solución de un problema en la Internet lo requiriera².

¹RFC1173, "Responsibilities of Host and Network Managers", A Summary of the "Oral Tradition of the Internet", 3. Responsibilities of Host System Managers.

²idem

Capítulo 8

Administración de Usuarios

La creación y eliminación de usuarios es una tarea de rutina en la administración de sistemas. Suele ser realizada a través de programas asistentes de administración, usualmente de interface gráfica, o a través de scripts. Aquí se describen las tareas paso a paso en su realización manual. Finalmente se hace referencia a comandos existentes en algunos sistemas para facilitar y asegurar la ejecución de estos procesos. El mantenimiento de usuarios es un tema de importancia para la seguridad del sistema; las cuentas poco usadas, o de contraseña trivial, son el blanco preferido de los hackers¹.

Existen servicios especiales para el manejo de cuentas a nivel de red, simplificando la administración y permitiendo el ingreso del usuario en cualquier máquina de la red. Estas facilidades tienen sus propios problemas de seguridad, son un tema aparte, no se tratan aquí.

8.1. El archivo `/etc/passwd`.

El archivo `/etc/passwd` contiene la lista de usuarios en el sistema, una línea por usuario. Es consultado cuando el usuario ingresa al sistema para determinar su UID; en muchos casos, contiene también una cadena que representa a la contraseña del usuario, aunque la tendencia actual es retener esta cadena en otro archivo, `/etc/shadow`, con permisos de acceso más restringidos. En este último caso, el proceso de login consulta ambos archivos, `/etc/passwd` y `/etc/shadow`.

Los campos de `/etc/passwd` y `/etc/shadow` están separados por ":", al igual que la mayoría de los archivos de configuración en UNIX.

Campos del archivo `/etc/passwd`:

```
username:password:uid:gid:gecos:home-directory:login-shell
```

Ejemplo de entradas en `/etc/passwd`:

```
root:x:0:0::/root:/bin/bash
```

¹Entendemos como hacker al pillito informático, sin embargo recordemos que antaño se llamaba hacker a los masters en cómputo.

```
bin:x:1:1:bin:/bin:
ftp:x:404:1::/home/ftp:/bin/bash
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
postmaster:x:14:12:postmaster:/var/spool/mail:/bin/bash
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
man:x:13:15:man:/usr/man:
guest:x:405:100:guest:/dev/null:/dev/null
nobody:x:65534:100:nobody:/dev/null:
paco:x:501:501:Francisco Medina:/home/paco:/bin/bash
```

En su versión clásica, el archivo */etc/passwd* contiene los siguientes campos:

login: nombre único, no más de 8 caracteres, sin signos de puntuación; generalmente en minúscula para compatibilidad con todos los agentes transporte de correo. Evitar nombres totalmente en mayúsculas, porque el sistema asumirá que el terminal no soporta minúsculas y abrirá una sesión totalmente en mayúsculas. El nombre de login es un dato público; deben elegirse nombres que favorezcan el reconocimiento de los usuarios en la vida real. En el acceso a diferentes máquinas, es conveniente asignar a un mismo usuario el mismo nombre de login. Conviene asimismo evitar el uso del mismo nombre para usuarios diferentes en distintas máquinas; los agentes transporte de correo y los usuarios pueden caer en confusión. También debe asegurarse no usar un nombre de login que sea a la vez un alias de correo para otro usuario.

contraseña cifrada: la contraseña se cifra² con un algoritmo llamado DES; debe ser fijada por el comando `passwd` (o `yppasswd` si se usa NIS), o copiar una contraseña ya cifrada de otra cuenta. Si este campo está en blanco, el acceso es sin contraseña; esto es una gran hoyo de seguridad. Si se coloca un asterisco en este campo, la cuenta no puede accederse hasta que el superusuario asigne una contraseña (comando `passwd`, o `yppasswd` si se usa NIS). En sistemas que usan */etc/shadow*, la contraseña se encuentra en este archivo y no en */etc/passwd*. Los archivos */etc/passwd* y */etc/shadow* deben mantenerse consistentes; existen comandos que atienden esta situación específicamente.

número UID: número identificatorio del usuario, generalmente entre 0 y 32767 o 65535, según se usen enteros con o sin signo. Muchos sistemas han pasado ya a números de 32 bits. Debe ser único en toda la red local. Los números de 0 a 99 se reservan para usuarios no humanos; 0 para root, 1 para bin, 2 para daemon, etc. Debe evitarse la reutilización de números de usuarios que han dejado la institución, para evitar confusiones si han quedado archivos del usuario anterior en el sistema o al restaurar desde respaldo; recordar que los usuarios en el sistema son reconocidos por número y no por nombre³.

A continuación se presenta los UID y su uso en Solaris:

²en realidad no es la contraseña cifrada, es el resultado de cifrar una cadena de 8 ceros usando como llave la contraseña

³Para UNIX no somos mas que un simple número

UID	Nombre de cuentas	Reservada para:
0 - 99	root, daemon, bin, sys, etc.	Cuentas del sistema
100 - 2147483647	Usuarios mortales	Cuentas de uso general
60001	nobody	Usuarios sin autenticarse
60002	noaccess	Compatibilidad con Solaris 2.0 y compatibilidad con otras versiones de SVR4

En el caso de Linux, las cuentas de usuario comienzan en el 500.

número GID por defecto: número del grupo, también entre 0 y 32767 o 65535. Los números bajos se reservan para grupos del sistema: 0 para el grupo root o wheel, 1 para el grupo daemon. Los grupos se definen en el archivo `/etc/group`. El número GID define el grupo de pertenencia de los archivos creados por el usuario, o de procesos iniciados por él.

información de usuario (campo "GECOS"): históricamente usado para transferir trabajos en batch desde una máquina UNIX hacia un mainframe corriendo GECOS (General Electric Computer Operating System). Se usa ahora, sin una sintaxis fija, para contener información del usuario. El comando `finger` interpreta este campo como una lista separada por comas conteniendo:

1. nombre en la vida real,
2. edificio y número de oficina,
3. teléfono interno,
4. teléfono de casa.

El comando `chfn` permite al usuario cambiar su información propia.

directorio propio (home): cuando el usuario ingresa al sistema, es colocado en su directorio propio; si éste no existe, algunos sistemas impiden el ingreso; otros emiten un mensaje de error, aceptan el ingreso y colocan al usuario en el directorio raíz. El directorio propio del usuario suele tener su mismo nombre de login; se ubica (o se monta) habitualmente bajo `/home`.

shell de login: al ingresar al sistema, el usuario dispone de un intérprete de comandos por defecto, generalmente

- `/bin/sh`,
- `/bin/csh`,
- `/bin/bash`,
- `/bin/ksh`,
- `/bin/tcsh`, u otros.

El usuario puede cambiar su intérprete con el comando `chsh`, o a veces simplemente invocando el nuevo shell; el archivo `/etc/shells` contiene los intérpretes habilitados para elección de los usuarios con `chsh`. Al editar `/etc/shells`, en los nombres de invocación de los shell debe figurar la vía completa.

fecha de expiración de la contraseña: día contado a partir del 1-1-1970, en que la cuenta expirará. Si el campo está en blanco, la cuenta no expira nunca. Si ha transcurrido esta fecha, el campo debe ser reemplazado por el administrador para rehabilitar la cuenta.

banderas: reservado para usos futuros.

8.3. El archivo `/etc/group`.

El archivo `/etc/group` contiene los nombres de los grupos UNIX definidos en el sistema y una lista de sus miembros. Hay una línea por cada grupo. Los campos están separados por ":".

Ejemplo de entradas en `/etc/group`:

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:man
usuarios::100:paco
nogroup::-2:
```

Cada línea contiene:

nombre del grupo: algunos sistemas piden 8 caracteres o menos.

contraseña cifrada : histórico, no se usa. El comando `newgrp` no cambia el grupo por defecto de un usuario si su nombre no está listado en el grupo al que quiere cambiar, aún cuando este campo esté en blanco (lo usual).

número GID: número único identificador del grupo. Para mantener coherencia en un sistema heterogéneo es recomendable no usar como grupo por defecto de los usuarios un grupo del sistema o del proveedor (por ejemplo, `users`, o `staff`), sino un grupo creado por el administrador (por ejemplo, `usuarios`), ya que diferentes sistemas pueden asignar diferente GID a los mismos grupos del sistema.

número GID: número único identificador del grupo. Para mantener coherencia en un sistema heterogéneo es recomendable no usar como grupo por defecto de los usuarios un grupo del sistema o

del proveedor (por ejemplo, `users`, o `staff`), sino un grupo creado por el administrador (por ejemplo, `usuarios`), ya que diferentes sistemas pueden asignar diferente GID a los mismos grupos del sistema.

lista de integrantes: nombres de login de los integrantes del grupo separados por comas, sin blancos.

8.4. Creación de cuentas de usuarios

Antes de abrir una cuenta a un nuevo usuario, éste debe firmar su conformidad con el documento descriptivo de política de uso del sistema local⁵. La experiencia demuestra que es mucho más difícil de conseguir una firma de conformidad después de haber sido abierta y concedida la cuenta.

La creación de usuarios se realiza en tres fases: dos para establecer el ambiente del usuario y una tercera para objetivos administrativos.

Etapas del proceso de creación de usuarios:

1. Requeridos:
 - definir la cuenta del usuario, ingresándola en los archivos `/etc/passwd` y `/etc/shadow`;
 - fijar una contraseña inicial;
 - crear el directorio propio (home) del usuario.
2. En beneficio del usuario:
 - copiar hacia el directorio personal del usuario archivos de inicialización;
 - fijar el imbox de correos del usuario y establecer alias de correo.
3. Para administración:
 - agregar el usuario al archivo `/etc/group`;
 - registrar información contable, si es necesario;
 - ingresar información de contacto con el usuario (dirección, teléfono);
 - configurar cuotas de disco;
 - verificar el establecimiento correcto de la cuenta.

8.4.1. Edición de archivos.

Si estas herramientas existen en el sistema, la edición de `/etc/passwd` debe hacerse con `vipw`, que bloquea el archivo para evitar interferencias si un usuario está cambiando su contraseña simultáneamente. Se insiste en la necesidad absoluta de mantener coordinados los archivos `/etc/shadow` y `/etc/passwd`. La edición de `/etc/group` se hace con `vigr`. Si el sistema provee scripts para crear, modificar o borrar usuarios, éstos se ocupan de mantener la coherencia y bloquear los archivos antes de modificarlos. El siguiente análisis de creación manual de un usuario muestra el procedimiento subyacente en las herramientas habituales de administración.

⁵comunmente no lo hacemos pero deberíamos

8.4.2. Fijar contraseña inicial.

El superusuario puede cambiar en cualquier momento la contraseña de cualquier usuario mediante el comando `passwd nombre_usuario`.

Un usuario común puede cambiar su propia contraseña digitando solamente `passwd`. El comando `passwd` suele exigir un mínimo de largo o uso de caracteres de puntuación, números, mezcla de mayúsculas y minúsculas, para alcanzar un cierto nivel de inviolabilidad. No debe dejarse nunca una cuenta sin contraseña, especialmente en ambientes de red o con acceso a Internet.

8.4.3. Crear directorio propio del usuario.

La siguiente secuencia de comandos crea un directorio para el usuario `jperez`:

```
mkdir /export/home/jperez
```

```
chown jperez /export/home/jperez
```

```
chgrp usuarios /export/home/jperez
```

```
chmod 700 /export/home/jperez
```

En ambientes menos restrictivos, los permisos de los directorios propios pueden ser `755`, lo que permite a todos los usuarios del sistema ver los directorios personales. Los usuarios siempre pueden limitar el acceso sobre los subdirectorios contenidos en su directorio personal.

8.4.4. Copiar archivos de inicialización.

Algunos comandos admiten personalización colocando archivos en el directorio propio del usuario. Estos archivos generalmente empiezan con punto, lo que los hace normalmente ocultos, y terminan con `rc`, por "run command". Algunos ejemplos comunes:

Comando	Archivo	Usos
csh	.login	fija tipo de terminal, variables de ambiente, opciones para biff y mesg.
	.cshrc	fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo; fija variables promtp, history, savehist.
	.logout	imprime recordatorios, borra la pantalla.
sh	.profile	fija tipo de terminal, variables de ambiente, opciones para biff y mesg; fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo, etc.
bash	.bash_profile	fija tipo de terminal, variables de ambiente, opciones para biff y mesg; fija alias de comandos, rutas de búsqueda, valor de umask, cdpath para búsqueda de nombres de archivo, etc.
	.basrh	fija alias de comandos, rutas de búsqueda, valor de umask, prompt
vi	.exrc	opciones para el editor vi.
emacs	.emacs.pro	Opciones y asignación de teclas para el editor emacs.
mailx	.mailrc	alias personales para correo, opciones para el lector de correo.
tin	.newsrsc	especifica grupos de interés para noticias.
xrdb	.Xdefaults	fija configuración de X11 (sistema de ventanas): tipos de letra, colores, etc.
startx	.xinitrc	fija ambiente inicial para X11.
xdm	.xsession	fija ambiente inicial para X11.

Suele haber un conjunto ya preparado de archivos de inicialización, usualmente en */etc/skel*; también pueden crearse en */usr/local/lib/skel*, editándolos para reflejar configuraciones útiles al usuario corriente.

Ejemlos de archivos de inicialización:

Ejemplo 1 - Archivo *.profile*

[Define la variables de busqueda path.] `PATH=$PATH:$HOME/bin:/usr/local/bin:/usr/ccs/bin:.`

[Define la ruta del inbox del usuario.] `MAIL=/var/mail/$LOGNAME`

[Define el nombre del servidor para el canal de noticias de Usenet.] `NNTPSERVER=server1`

[Define la variable de busqueda de páginas de manual.] `MANPATH=/usr/share/man:/usr/local/man`

[Define la impresora por default.] `PRINTER=printer1`

[Define los permisos de archivos nuevos.] `umask 022`

[Da amplitud global a las variables.] `export PATH MAIL NNTPSERVER MANPATH PRINTER`

Ejemplo 2 - Archivo *.cshrc*

[Define la variables de busqueda path.] `set path=($PATH $HOME/bin /usr/local/bin /usr/ccs/bin)`

[Define la ruta del inbox del usuario.] `setenv MAIL /var/mail/$LOGNAME`

[Define el nombre del servidor para el canal de noticias de Usenet.] `setenv NNTPSERVER server1`

[Define la impresora por default.] `setenv PRINTER printer1`

[Crea un alias para el comando `history`] `alias h history`

[Define los permisos de archivos nuevos.] `umask 022`

[Corre algun script] `source /net/server2/site-init-files/site.login`

8.4.5. Editar `/etc/group`.

Si bien no es necesario asignar el usuario a su grupo por defecto en `/etc/group`, conviene igualmente hacerlo para tener actualizada la lista de integrantes del grupo. Sí debe editarse `/etc/group` para incluir al usuario en grupos secundarios. Algunos sistemas requieren que el usuario figure en el grupo `wheel` para poder hacer `su`.

8.4.6. Tabla de usuarios y guía telefónica.

Es fácil crear una tabla de usuarios y números telefónicos extrayendo la información del campo GECOS del archivo `/etc/passwd`, sobre todo si se tomaron en cuenta las recomendaciones para uso de comas en este campo. Esta tabla puede luego interrogarse con un script simple tal como:

```
#!/bin/sh
# telefono: devuelve linea con datos del usuario
grep $1 /usr/local/pub/telefonos
```

que se invoca simplemente como

```
telefono jperez y devuelve una línea con los datos del usuario indicado en el parámetro.
```

8.4.7. Fijar cuotas.

Si en el sistema se utilizan cuotas para controlar el uso de disco, es preciso fijar la cuota para el nuevo usuario con el comando `edquota`. Este comando admite diversos parámetros para fijar límites en el uso de disco, pero en la creación de usuarios es más frecuente usarlo en modo prototipo, copiando el perfil de cuota de algún usuario tomado como modelo: `edquota -p usuario-modelonuevo-usuario`

8.4.8. Verificar la nueva cuenta.

Para verificar la validez de la nueva cuenta, ingresar al sistema como el nuevo usuario y ejecutar los comandos

```
pwd
```

```
ls -la
```

para verificar la ubicación inicial en el directorio propio y la existencia de los archivos de inicialización.

El usuario debe ser notificado por algún medio **seguro** de su nombre de login y contraseña. Este momento es oportuno para entregarle documentación descriptiva del sistema. Ya debe haber firmado su conformidad con el documento de recomendaciones de uso y comportamiento esperado; solo resta recordarle la necesidad de cambiar su contraseña en cuanto ingrese al sistema.

8.5. Baja de usuarios

Cuando un usuario abandona la institución, su cuenta debe ser inmediatamente bloqueada. Se procederá luego a eliminar la cuenta de los archivos de contraseñas y grupos, el inbox de correo, las referencias en la tabla de usuarios, listados telefónicos y otros, conservando los datos históricos que la institución especifique.

Los archivos podrán transferirse en propiedad a otro usuario o ser eliminados. En definitiva, todo rastro del nombre de login debe ser eliminado, pero debe recordarse la recomendación de no reutilizar los números UID.

Una lista de control incluiría:

- Fijar cuotas del usuario en 0, si se está usando cuotas;
- Eliminar al usuario de guías telefónicas, bases de datos o listas;
- Eliminar al usuario del archivo de alias, o redirigir su correo a otra dirección;
- Eliminar el archivo de procesos diferidos (crontab) y trabajos pendientes con at.
- Eliminar procesos del usuario que se hallen aún corriendo.
- Eliminar archivos temporales del usuario en /tmp y /var/tmp.
- Eliminar al usuario de los archivos passwd, shadow, group.
- Eliminar el directorio propio del usuario.
- Eliminar el archivo de distribución de correo del usuario en /var/spool/mail.

8.6. Inhabilitar cuentas

Cuando una cuenta debe ser temporalmente inhabilitada, por ejemplo por ausencia del titular, se solía colocar simplemente un asterisco en el lugar de la contraseña en el */etc/shadow*.

8.7. Caducidad de contraseñas.

Los UNIX modernos disponen de una facilidad para obligar al usuario a cambiar su contraseña periódicamente. Esta práctica está un tanto cuestionada, ya que el usuario obligado a disponer de varias contraseñas se ve fácilmente tentado hacia elecciones menos seguras. Sí se insiste o aún obliga a los usuarios a cambiar su contraseña cuando han habido violaciones de seguridad o sospechas fundadas.

8.8. Pseudo logins.

Son cuentas de usuario que no corresponden a personas. Es el caso de bin y daemon. Es posible crear otras cuentas de pseudo login, como shutdown, halt, who; estas cuentas tienen como shell un programa que sólo ejecuta el comando indicado.

8.9. Herramientas para la administración de usuarios.

En el caso particular de CentOS contamos con las siguientes herramientas para la administración de usuarios:

/usr/sbin/useradd Añade cuentas de usuarios. Esta herramienta también es utilizada para especificar memberships primaria y secundarias a grupos.

/usr/sbin/userdel Elimina cuentas de usuarios.

/usr/sbin/usermod Modifica los atributos de las cuentas incluyendo algunas funciones relacionadas a la expiración de contraseñas. Para un control más refinado, utilice el comando passwd. También se utiliza usermod para especificar memberships de grupos primaria y secundaria.

passwd Configura una contraseña. Aunque se utiliza principalmente para modificar la contraseña de un usuario, esta también puede controlar todos los aspectos de la expiración de contraseñas.

/usr/sbin/chpasswd Lee un archivo que consiste de pares de nombres de usuarios y contraseñas, y actualiza cada contraseña de usuario de acuerdo a este.

chage Cambia las políticas de envejecimiento de contraseñas. También se puede utilizar el comando passwd para este propósito.

chfn Cambia la información GECOS de un usuario.

chsh Cambia el intérprete de comandos por defecto de un usuario.

Aunque útiles, estos comandos y scripts rara vez implementan todas las políticas de uso locales. Es aconsejable reescribir o adaptar los scripts adduser y rmuser para realizar vía scripts todo el proceso de creación y remoción de usuarios.

8.10. Herramientas para la administración de grupos

Para la administración de grupos contamos con los siguientes comandos:

/usr/sbin/groupadd Añade grupos, pero no asigna usuarios a estos grupos. Se deben utilizar los programas useradd y usermod para asignar usuarios a un grupo dado.

/usr/sbin/groupdel Elimina grupos.

/usr/sbin/groupmod Modifica nombres de grupos o GIDs, pero no cambia la membresía del grupo. Se tiene que utilizar `useradd` y `usermod` para asignar usuarios a un grupo determinado.

gpasswd Cambia la membresía de un grupo y configura contraseñas para permitir a los usuarios que no sean miembros que conocen la contraseña, unirse al grupo. También se utiliza para especificar administradores de grupos.

/usr/sbin/grpck Verifica la integridad de los archivos `/etc/group` y `/etc/gshadow`.

Capítulo 9

Inicio y Baja del sistema

El sistema operativo Linux es un sistema complejo; los procesos de arranque y detención de una máquina Linux implican muchas tareas; deben realizarse correctamente si se desea mantener la salud del sistema. No basta con prender o apagar un interruptor.

Aunque lo dicho aquí es aplicable en general, los procesos de arranque y detención son dependientes de hardware: puede haber diferencias para un equipo en particular.

9.1. Bootstrapping

El arranque del sistema suele llamarse “booting” o “booteo” en la jerga informática.

Durante el arranque no están disponibles los servicios del sistema; éste debe levantarse a sí mismo iniciando todos sus servicios (bootstrapping). Cuando una máquina se enciende, ejecuta un programa de carga cuyas instrucciones se encuentran almacenadas en ROM. Este programita determina como cargar en memoria el núcleo del sistema operativo (kernel) y comenzar a ejecutarlo. El kernel examina el hardware probando todos los dispositivos conectados, e inicia un proceso llamado `init`, siempre con identificador de proceso PID 1. Se verifican los sistemas de archivos, se montan, se arrancan los demonios del sistema, siguiendo los dictados de una serie de scripts en lenguaje de *shell* llamados *scripts rc* (`rc` = run command). El contenido y estructura de los scripts `rc` determinan la situación final del sistema.

9.2. Arranque automático y arranque manual.

La mayoría de los sistemas operativos tienen un modo de arranque manual y otro automático.

En *modo automático*, el sistema operativo realiza las tareas correspondientes al proceso de arranque en forma autónoma, sin necesidad de intervención del administrador, ejecutando todos los scripts de arranque e iniciando todos los procesos necesarios para brindar los servicios habituales a los usuarios.

En *modo manual* el sistema operativo ejecuta una primera parte del proceso de arranque pero casi enseguida transfiere el control al administrador. Se ejecutaron sólo unos pocos scripts, hay pocos procesos corriendo, sólo el superusuario puede acceder al sistema: se está ejecutando en modo *monousuario* (single-user mode).

En la operativa diaria el sistema arranca en modo automático con sólo encender el equipo. Algunas fallas pueden obligar al arranque en monousuario: una falla en una tarjeta de red o un sistema de archivos corrupto. Es preciso conocer bien el proceso de arranque para configurar el arranque automático de los servicios requeridos o intervenir en caso de falla.

9.2.1. Pasos del proceso de arranque.

El proceso de arranque tiene varios pasos que en general son los siguientes:

1. Carga e inicialización del núcleo (kernel).
2. Detección y configuración de dispositivos.
3. Creación automática de procesos base.
4. (Intervención del administrador - solo en modo monousuario).
5. Ejecución de scripts de inicialización.
6. Operación en multi-usuario.

El administrador tiene poco control de esta secuencia, pero sí puede alterar los scripts de inicialización, donde se arrancan los procesos capaces de brindar servicios a los usuarios.

9.2.2. Inicialización del núcleo (kernel).

El núcleo del Linux es un programa, y como todo programa debe cargarse previamente en memoria para poder ejecutarse. El núcleo reside en un archivo llamado *unix*, *vmunix*, *vmlinuz* o similar.

Al encender el equipo comienzan a ejecutarse instrucciones en ROM cuyo objeto es transferir a memoria un pequeño programa de arranque (boot program) encargado de cargar el kernel en memoria y comenzar a ejecutarlo. Esta primera parte del proceso, hasta alcanzar la ejecución del kernel, la realiza el hardware de la máquina. Una vez iniciado, el kernel verifica la cantidad de memoria, separa una parte para sí mismo e informa la cantidad de memoria total, lo reservado para sí y lo disponible para procesos de usuario.

9.2.3. Configuración del hardware.

Al comenzar su ejecución el núcleo intenta localizar e inicializar los dispositivos que le hayan sido asignados en su construcción. Prueba estos dispositivos uno por uno, intentando determinar parámetros de funcionamiento no especificados interrogando al propio dispositivo. Los dispositivos no hallados o que no responden son inhabilitados.

Una vez completado este proceso, si se agrega un nuevo dispositivo deberá esperar el próximo arranque del sistema para ser reconocido. Los sistemas Linux suelen venir con uno o más núcleos genéricos donde están preconfigurados los dispositivos más comunes, pero es posible reconstruir el núcleo optimizándolo estrictamente al hardware disponible. También es posible prever el agregado dinámico de módulos complementarios del kernel, cargándolos en memoria solo al detectarse la presencia del dispositivo físico o solicitarse su acceso.

9.2.4. Procesos del sistema.

Una vez completada la inicialización básica el núcleo crea algunos procesos espontáneos, llamados así por no haber sido creados por fork, el mecanismo habitual en Linux. Los procesos espontáneos difieren entre sistemas en la tabla 9.1 se mencionan los creados en sistemas BSD, mientras que en la tabla 9.2 los creados en Unix System V.

swapper	Proceso 0
init	Proceso 1
pagedaemon	Proceso 2

Cuadro 9.1: Procesos espontáneos en BSD

sched	Proceso 0
init	Proceso 1
varios manejadores de memoria y procesos del núcleo	

Cuadro 9.2: Procesos espontáneas en System V

En Linux no hay un proceso visible con PID 0, sino varios procesos de manejo además de init, diferentes según la versión del kernel como se observa en la tabla 9.3

init	Proroceso 1 varios manejadores de memoria y procesos del núcleo (kflushd, kupdate, kpiod, kswapd).
------	---

Cuadro 9.3: Procesos espontaneos en Linux

9.2.5. Intervención del operador (solo en arranque manual).

Si el sistema fue arrancado en modo monousuario el proceso init es invocado por el núcleo con un parámetro indicativo, pidiendo la contraseña de superusuario y arrancando un intérprete de comandos (shell); al finalizar la ejecución del intérprete init continúa con el proceso normal de arranque. Digitando Ctrl-D en lugar de la contraseña del superusuario continúa el arranque sin invocar el shell.

En el shell monousuario el superusuario puede trabajar como en cualquier sesión, pero solo dispondrá de comandos existentes en la partición raíz, la única montada. Esta partición puede, además, haber sido montada en solo lectura para ser verificada; si /tmp está en la partición raíz, programas necesitados de archivos temporales como vi no funcionarán. Volver a montar la partición raíz en modo lectura escritura puede diferir entre sistemas, pero en general mount / leerá de nuevo /etc/fstab para

montar / en el modo habitual. Otros sistemas de archivos, como /usr, pueden ser montados a mano si es necesario. Una falla habitual de arranque son sistemas de archivos con inconsistencias; en este caso, deberá correrse fsck en forma manual antes de montar el recurso. En el modo automático los sistemas de archivos son verificados como parte del proceso de arranque; no así en monousuario.

9.2.6. Scripts de arranque (scripts rc).

Al momento de correr los scripts rc el sistema ya es un Linux típico. La ubicación exacta y la organización de esos archivos depende del sistema. En la sección 9.3 detallaremos estas rutas y scripts para Linux.

9.2.7. Operación en multiusuario.

Una vez ejecutados los scripts de arranque el sistema se encuentra operativo, pero para aceptar el ingreso de usuarios (login) init debe arrancar un procesos getty de escucha en cada una de las líneas de conexión de terminales. Esto completa el proceso de arranque. En los sistemas configurados para usar login en ambiente gráfico init arranca estos servicios (xdm, gdm, dtlogin u otro).

El proceso init sigue desempeñando un rol importante durante todo el período de funcionamiento del sistema, arrancando procesos y recibiendo en herencia procesos sin padre. En BSD init tiene solo dos estados, monousuario y multiusuario; en System V existen diferentes estados mono y multiusuario (run levels), cada uno con diferente espectro de recursos y servicios.

9.3. Scripts de inicialización de Linux

En BSD, los scripts rc se ubican bajo */etc*, con nombres comenzados por "rc"; pueden invocarse unos a otros. En System V, como es el caso de Linux, los scripts se guardan bajo */etc/init.d*, con enlaces hacia ellos en directorios */etc/rc0.d*, */etc/rc1.d*, ..., correspondientes cada uno a un nivel de arranque.

Usualmente los scripts rc realizan, entre otras, las siguientes tareas:

- Fijar el nombre de la máquina.
- Fijar la zona horaria.
- Verificar los discos con fsck (en arranque automático).
- Montar los discos del sistema.
- Eliminar archivos del directorio /tmp.
- Configurar las interfaces de red.
- Arrancar los demonios del sistema y los servicios de red.

9.4. Scripts rc estilo System V.

El estilo System V es el más usado actualmente. Se definen en él 7 niveles de arranque (run levels), cada uno con un conjunto de servicios particular:

Nivel 0:	sistema bajo, no hay nada corriendo.
Nivel 1 o S:	monousuario.
Niveles 2 a 5:	diversos (o idénticos) niveles multiusuario.
Nivel 6:	rearranque (reboot), el sistema se detiene y vuelve a arrancar.

Run Levels en Solaris:

Run Level	Init State	Type	Usar este nivel para:
0	Power-down state	Power-down	Dar de baja el sistema operativo, para agarrar el equipo sin ningun problema.
S or S	Single-user state	Single-user	para ir a single user con todos los sistemas de archivos montados y accesibles.
1	Estado administrativo	Single-user	Acceder a todos los sistemas de archivos sin que los usuarios puedan entrar al sistema.
2	Estado Multiusuario	Multiuser	Para operaciones normales. Los usuarios pueden acceder al sistema y a todos los sistemas de archivos. Todos los daemons esta esperando con excepción de los servicios de NFS.
3	Multiusuario con servicio NFS	Multiuser	Para operaciones normales con NFS.
4	Estado Multiusuario especial		Este nivel no esta disponible.
5	Estado de apagado	Power-down	Para apagar el equipo. Si es posible apagar el equipo de manera automática.
6	Estado Reboot	Reboot	para llevar el equipo al nivel 0, y despues rebootear a modo multiusuario. (o cualquier otro nivel definido en el archivo initab).

Run Levels en CentOS:

Run Level	Descripción:
0	halt
1	Modo single user
2	Multusuario, sin NFS (Igual que el nivel 3 pero sin red)
3	Modo Multiusuario con red
4	No definido
5	X11
6	reboot

Como nivel multiusuario se usa únicamente el nivel 2, o el 2 y el 3 para configuración sin y con red, por ejemplo; los niveles 4 y 5 no suelen ser usados. Los niveles 1 y S difieren según los sistemas. Los niveles 0 y 6 no son en realidad estados de funcionamiento, sino transitorios.

El nivel 1 es el monousuario por excelencia; el S fue creado para pedir la contraseña del superusuario. En Solaris S es el monousuario, pero en Linux se usa 1 como monousuario.

Los niveles se definen en el archivo `/etc/inittab`. Aunque el formato varía, el propósito de este archivo es definir los comandos a correr en cada nivel; uno de ellos define el nivel por defecto que ha de alcanzar el sistema. Los scripts se ejecutan pasando ordenadamente por cada nivel, creciendo en el arranque y decreciendo en la detención. Los scripts de cada nivel detienen los servicios correspondientes a niveles superiores y arrancan los servicios correspondientes al nivel propio. No suele ser necesario tocar el archivo `/etc/inittab`; el control puede realizarse totalmente a través de los scripts `rc` correspondientes a cada nivel, confiando en el sistema para la invocación ordenada de los scripts `rc` de cada nivel según el directorio en que se encuentran.

Los scripts colocados en `/etc/init.d` manejan cada uno un demonio, servicio o aspecto particular del sistema. Todos los scripts deben entender al menos los parámetros `start` (arrancar), `stop` (detener); muchos entienden también `restart`, un `stop` seguido de un `start`. Esto permite al administrador gobernar un servicio manualmente con comandos tales como:

```
/etc/init.d/sshd start
/etc/init.d/sshd stop
```

El conjunto de servicios a detener y arrancar en cada nivel se controla disponiendo de un directorio para cada nivel y estableciendo en él enlaces hacia los scripts en `init.d`, según la siguiente convención de nombres:

```
[K|S][0-9][0-9]nombre_script
```

nombre de enlace iniciado en **K** invoca el script en `init.d` con parámetro `"stop"`;

nombre de enlace iniciado en **S** invoca el script en `init.d` con parámetro `"start"`;

el número de 2 cifras que sigue a **K** o **S** indica el orden de ejecución;

sigue el nombre del script en `init.d`.

Ejemplos: `S34named`, `K29bind`.

Los directorios para cada nivel siguen la convención de nombres `rcN.d`, donde **N** es el número de nivel. Ejemplos: `rc0.d`, `rc2.d`, `rcS.d`. Cuando el sistema está subiendo, se ejecutan los scripts **S**; cuando está bajando se ejecutan los scripts **K**, siempre en el orden definido por los números en los nombres de sus enlaces simbólicos. Los comandos

```
ln -s /etc/init.d/sshd /etc/rc2.d/S99sshd
ln -s /etc/init.d/sshd /etc/rc2.d/K25sshd
ln -s /etc/init.d/sshd /etc/rc6.d/K13sshd
```

crean los enlaces simbólicos necesarios para arrancar y detener el servicio `sshd` en el nivel multiusuario 2. La colocación del enlace en el nivel 6, `reboot`, es para asegurarse que la detención del servicio se haga en el reinicio, si el sistema no recorre por sí en forma ordenada los niveles de detención.

Las distribuciones de Linux usan distintos esquemas de scripts rc: Debian sigue el esquema System V descrito, en forma similar a Solaris y HP-UX; Slackware usa el esquema BSD, como FreeBSD; Redhat usa un híbrido System V y BSD con algunas complicaciones de propia cosecha.

Para determinar el estado en el que se encuentra Linux: `who -r`

9.5. Problemas de arranque.

Las dificultades de arranque pueden deberse a diferentes causas:

- Problemas de hardware.
- Problemas con el gestor de arranque del sistema operativo.
- Problemas en el sistema de archivos.
- Problemas de configuración del núcleo (kernel).
- Errores en los scripts de arranque (scripts rc).

9.5.1. Problemas de hardware.

Antes de diagnosticar un problema de hardware es preciso descartar fehacientemente errores en la configuración del software; muchos de esos errores sugieren fallas de hardware a los ojos inexpertos. Un mensaje de error específico proveniente directamente de un dispositivo es un buen indicador de falla de hardware (error de memoria, error de controlador de disco). Siempre es conveniente verificar aspectos de instalación tales como:

- Alimentación de corriente para todas las partes del equipo: gabinete principal, gabinetes externos con discos, cinta u otros periféricos.
- Alimentación de dispositivos internos (discos, disqueteras, unidades de cinta, CDROM).
- Conexiones entre diferentes dispositivos (cables de señal de discos, disqueteras, puertos).
- Conexiones de red, sobre todo si la máquina depende de recursos externos (estación sin disco, sistemas de archivos montados de otro servidor de red).
- Si existen, revisar e interpretar las luces indicadoras de estado de los dispositivos.
- Si la máquina comienza el arranque, verificar la aparición de todos los dispositivos conectados; cada uno emite un mensaje de una línea al ser reconocido. Un mensaje de error o la falta de detección de un dispositivo conectado correctamente puede indicar una falla de hardware en ese dispositivo, en su plaqueta controladora o en la plaqueta principal.
- Cuando sea posible, usar programas de diagnóstico; los hay para dispositivos independientes del sistema operativo. Las máquinas de hardware propietario disponen de rutinas de diagnóstico de hardware en ROM. En las computadoras personales el BIOS ejecuta en el arranque una prueba de autoverificación llamada POST (Power On Self Test) que prueba la memoria, presencia de discos y otros elementos básicos; los periféricos suelen traer programas para diagnóstico (tarjetas de red, de video, impresoras).

- Al detectarse una falla conviene dejar el equipo apagado unos 30 segundos y luego arrancarlo, para asegurarse de llevar el hardware a un estado conocido.

9.5.2. Problemas con el gestor de arranque del sistema operativo.

En las máquinas de arquitectura propietaria existe un programa de carga del sistema operativo almacenado en ROM. Una falla común es la alteración o borrado de una variable de ambiente almacenada en memoria no volátil. En este y otros casos es preciso seguir las indicaciones del fabricante para reponer el arranque.

Los sistemas Linux para computadores personales vienen programas de carga tales como Grub para Linux o BootEasy para FreeBSD. Estos gestores permiten arrancar electivamente Linux u otros sistemas operativos instalados en diferentes particiones de un mismo disco. Una falla, alteración o borrado del programa gestor obliga a arrancar el sistema con un medio alternativo (disquete, CD), y reinstalar el programa de carga o corregir y reponer su configuración correcta.

Las fallas de carga pueden darse en un sistema operativo instalado correctamente; la reposición del esquema de arranque recupera el sistema completamente.

9.5.3. Problemas en el sistema de archivos.

Un sistema de archivos puede fallar por razones físicas (superficie dañada, disco roto), o lógicas (inconsistencias en las estructuras de almacenamiento). En caso de falla física habrá que cambiar el disco o al menos realizar una detección de bloques defectuosos para evitar su uso, crear nuevamente el sistema de archivos y reponer desde respaldos. Una falla lógica puede ser reparable con fsck, eventualmente con alguna pérdida; de no ser así, será preciso recrear el sistema de archivos y reponer desde respaldo. La verificación deberá realizarse arrancando en modo monousuario, con el sistema de archivos en falta desmontado y sobre el dispositivo, ejecutando fsck reiteradamente en modo forzoso hasta no obtener errores.

Si la falla es en el sistema de archivos raíz el sistema no podrá arrancar. Si se dispone de una partición de arranque alternativa, puede levantarse el sistema desde ella y reponer el servicio rápidamente. Según los sistemas, esto puede hacerse con comandos de ROM o del programa de carga; es preciso tener muy clara esta secuencia, y haber probado el arranque al crear la partición alternativa, así como mantener permanentemente sincronizadas (idénticas) ambas particiones de arranque, mediante copia cruda con dd. Estas particiones deben estar en discos distintos para obtener una confiabilidad razonable. En ausencia de partición de arranque alternativa será preciso reinstalar el sistema operativo, eventualmente reponiendo desde respaldo los archivos de datos.

Una falla lógica en el sistema de archivos raíz puede intentar repararse arrancando el sistema desde un medio alternativo (CD, disquete de rescate o disquete de instalación) y correr fsck sobre el sistema de archivos raíz, no montado por el arranque desde medio alternativo.

9.5.4. Problemas de configuración del núcleo (kernel).

Cuando se genera un nuevo núcleo debe mantenerse siempre un respaldo del núcleo anterior, y saber como levantar el sistema con él; un núcleo recientemente generado puede muy bien no funcionar. En

Linux es posible generar el nuevo núcleo en disquete, para fines de prueba, y recién después copiar la nueva versión sobre la anterior en disco.

9.5.5. Errores en los scripts de arranque.

Los errores en los scripts de arranque son la causa más frecuente de falla en el arranque; son también las de más fácil solución: basta arrancar el sistema en monousuario, editar los scripts rc, corregir los errores y continuar con el arranque normal. Es preciso verificar qué editor hay disponible en modalidad monousuario y saber manejarlo; algunos sistemas requieren montar /usr para disponer de vi. El editor ed, poco amigable, suele estar disponible en todos los Linux. En Linux, el disquete de rescate ofrecido por las distintas distribuciones suele tener un conjunto de herramientas razonables para lidiar con estas dificultades; puede dar mejores resultados recurrir al disquete de rescate que intentar el arranque monousuario.

9.6. Baja del sistema

En otros sistemas operativos es práctica común arrancar de nuevo el sistema operativo como primer intento para resolver cualquier problema¹. Los sistemas Linux suelen realizar tareas críticas, atender múltiples usuarios o soportar redes de datos; no puede pensarse en *reiniciar* de nuevo el sistema así como así; es imprescindible pensar primero e intentar solucionar los problemas sin reiniciar la máquina. Esto no es una aspiración inalcanzable, es posible en muchos casos. Es normal para una máquina Linux pasar meses sin detenerse. Solo en casos de agregar un dispositivo, sufrir una falla de hardware, un estado de confusión del sistema o la imposibilidad de acceder, puede pensarse en rearrancar. En suma, cuando realmente no hay otra solución.

Si se han hecho cambios en los scripts de arranque conviene reiniciar el sistema para verificar el funcionamiento correcto de los cambios efectuados sin esperar el próximo inicio, que puede darse en un momento donde el autor de los cambios no esté presente o ya no recuerde lo que ha hecho².

Hay diferentes formas de bajar un sistema o arrancarlo de nuevo:

- Botonazo.
- Usar el comando shutdown.
- Usar el comando halt y reboot (BSD, Linux, Solaris).
- Usar init para cambiar el nivel de ejecución del proceso init³ (System V).

Ni siquiera en sistemas Linux personales es aceptable bajar el equipo presionando el botón de poder: pueden perderse datos o corromperse lógicamente los sistemas de archivos. Apagar con el botón de poder puede ser inevitable en caso de catástrofe natural o de bloqueo total del sistema.

¹Y que conste que no dije Windows

²Pasa muy amenudo

³El proceso *init* no es lo mismo que el comando *init*

9.6.1. ¿Cuándo apagar un equipo?

Un equipo Linux tiene que apagarse cuando se ejecuten una de las siguientes tareas:

- Añadir o quitar algún dispositivo.
- Prepararse para una interrupción del suministro eléctrico.
- Hacer actividades de mantenimiento, como respaldos o instalar o remover software.

9.6.2. shutdown

El comando shutdown es el método seguro y completo de apagar un sistema Linux. Dispone de una variedad de opciones, variables según los Linux, pero en general permite fijar anticipadamente el momento del apagado, enviar avisos a los usuarios, decidir si será un apagado, un reinicio o un cambio de runlevel a monousuario. Procede enviando señales de terminación a todos los procesos en forma ordenada, sincroniza y desmonta los sistemas de archivos, avisa cuando terminó ("System halted", "Power down", o similar).

Sintaxis:

```
# shutdown -iinit-state -ggrace-period -y
```

donde:

-iinit-state Lleva el sistema al nivel diferente al de default. Las opciones son 0, 1, 2, 5, y 6.

-ggrace-period Indica el tiempo (en minutos) antes de que el sistema se apage. El tiempo por default son 60 segundos.

-y Continúa el proceso de apagado sin intervención; de otra manera es necesaria la intervención del administrador para continuar con el proceso de apagado.

Ejemplos:

```
shutdown -r now
```

en Linux rearranca (-r) la máquina en forma inmediata (now). La opción -h detiene; la opción -f no realiza verificación de discos en el siguiente inicio.

9.6.3. halt

El comando halt realiza las tareas esenciales mínimas para bajar el sistema ordenadamente: registra la bajada, termina los procesos no esenciales, sincroniza los discos y termina la ejecución.

La opción -n no sincroniza discos; se usa cuando se ha hecho un fsck sobre la partición raíz para impedir al kernel sobrecribir el superbloque reparado con el defectuoso que retiene en memoria.

9.6.4. reboot

El comando reboot procede como al halt pero arranca de nuevo el sistema después de bajarlo. Tiene también opción -n, como halt.

9.6.5. Cambiar nivel de ejecución (System V).

El comando init permite avisarle al procesoinit que cambie a otro nivel de ejecución.

```
init S
```

pasa el sistema a modo monousuario en Solaris y HP-UX; en Linux

```
init 1
```

realiza la misma acción (S solo abriría un nuevo shell). No hay período de gracia ni advertencias.

```
shutdown -il
```

es una forma más elegante de pasar a monousuario. init es útil para probar cambios al archivo /etc/inittab:

```
init -q
```

obliga a init a releer el archivo inittab.

Capítulo 10

Administración de Software

10.1. Administración de paquetes con RPM

En este tema se introducen los conceptos básicos de la administración de paquetes, que son la forma principal de instalar y administrar el software de su sistema Linux.

Package management system (sistema de administración de paquetes) es la parte del sistema Linux que permite instalar, desinstalar y gestionar el software de una máquina. Haciendo una burda comparación con sistemas operativos Microsoft Windows, se trata de algo similar a la opción Añadir y Quitar programas en el panel de control de Windows, siendo en Linux normalmente mucho más flexible y fácil de consultar.

Casi todas las distribuciones Linux incluyen algún tipo de sistema de administración de paquetes. El más difundido es el **Red Hat package manager (RPM)**. Distribuciones basadas en RedHat, como Suse o Mandriva, utilizan este administrador de paquetes.

Debian utiliza otro administrador de paquetes llamado `dpkg`. Este es, en muchos aspectos, superior a RPM, especialmente en lo que se refiere al manejo de las dependencias del paquete. Con su interfaz de acceso `dselect` y `apt`, `dpkg` es bastante fácil de usar (suponiendo que sepa contestar las numerosas preguntas de configuración que los paquetes Debian suelen preguntar).

Aunque RPM es el programa principal de instalación para un sistema Linux, en realidad, es mucho más que eso. La mayor parte del software que se instala está disponible en formato de paquete, lo que quiere decir que virtualmente todo el software que use se puede instalar con el uso de RPM. Las excepciones a esto se harán compilando el código fuente de la aplicación.

RPM tiene cinco modos de operación: instalación, desinstalación, actualización, búsquedas y verificación. Este sección describe la forma de instalación de paquetes usando la herramienta `rpm`.

10.1.1. Instalación de Paquetes

Los paquetes RPM tienen típicamente nombres de archivo como `foo-1.0-1.i386.rpm`, que incluye el nombre del paquete (`foo`), versión (`1.0`), desarrollo (`1`), y arquitectura (`i386`). Instalar un paquete es tan sencillo

como:

```
# rpm -Uvh foo-1.0-1.368.rpm}
foo                               #####
```

Como podemos ver, RPM imprime en pantalla el nombre del paquete (el cual no es necesariamente el mismo que el archivo, el cual puede ser 1.rpm), e imprime una sucesión de gatos a medida que el paquete es instalado, como un medidor del progreso de la instalación.

La instalación de paquetes está diseñada para ser sencilla, pero no exenta de errores. Los errores más comunes a los que nos podemos enfrentar son:

- El paquete que queremos instalar ya este instalado.
- Existe un conflicto de archivos.
- Existen dependencias no resueltas.

Paquete ya instalado

Si el paquete está ya instalado, veremos algo similar a:

```
# rpm -ivh foo-1.0-1.i386.rpm
foo                package foo-1.0-1 is already installed
error: foo-1.0-1.i386.rpm cannot be installed
```

Esta advertencia protege al sistema, pero si estamos seguros de querer forzar la instalación podemos usar la opción `--replacepks`, lo que le dice a RPM que ignore el error.

Conflicto de archivos

Si intentamos instalar un paquete que contiene un archivo que ha sido ya instalado por algún otro paquete, veremos algo como esto:

```
# rpm -ivh foo-1.0-1.i386.rpm
foo                /usr/bin/foo conflicts with file from bar-1.0-1
error: foo-1.0-1.i386.rpm cannot be installed
```

Para hacer que RPM ignore el error, tenemos que usar la opción `--replacefiles`.

Dependencias no resueltas

Los paquetes RPM generalmente “dependen” de otros paquetes, lo cual significa que requieren que otros paquetes sean instalados para funcionar correctamente. Si intentamos instalar un paquete para el cual existe una dependencia no satisfecha, veremos lo siguiente:

```
# rpm -ivh bar-1.0-1.i386.rpm
failed dependencies:
    foo is needed by bar-1.0-1
```

Este es el lado obscuro del manejo de paquetes de RedHat, para poder instalar este paquete tendríamos que buscar cual es el paquete que cumple con la dependencia. Si aún queremos instalar el paquete, con la salvedad que no funcione, podemos usar la opción `--nodeps`.

Para resolver este problema usamos el comando `yum`, el cual es un manejador de paquetes por encima de `rpm` que busca en internet los paquetes que sean necesario para cumplir con las dependencias.

10.1.2. Desinstalación de paquetes

Para desinstalar un paquete solo tenemos que usar la opción `-e`, como se muestra a continuación:

```
# rpm -e foo
```

Pongamos especial atención en que usé el nombre “foo” para el paquete, no el nombre del paquete original “foo-1.0-1.i386.rpm”.

En el proceso de desinstalación nos podemos topar de nuevo con el problema de las dependencias:

```
# rpm -e foo
removing these packages would break dependencies:
    foo is needed by bar-1.0-1
```

Para hacer que RPM ignore el error y desinstale el paquete de todas maneras (lo cual es una mala idea porque el paquete que depende de éste probablemente falle y no funcione correctamente), utilizamos la opción `--nodeps`.

10.1.3. Actualización de paquetes

Actualizar un paquete es casi como instalar un paquete.

```
# rpm -Uvh foo-2.0-1.i386.rpm
foo #####
```

La única diferencia radica en el hecho de que RPM desinstala automáticamente cualquier versión antigua del paquete foo. De hecho lo mejor es usar siempre la opción `-U` para instalar paquetes, ya que funciona bien incluso cuando no hay ninguna versión anterior del paquete instalada.

Dado que actualizar es realmente una combinación de desinstalación e instalación, nos podemos enfrentar al problema de las dependencias, como lo muestra el siguiente ejemplo:

```
# rpm -Uvh foo-1.0-1.i386.rpm
foo package foo-2.0-1 (which is newer) is already installed
error: foo-1.0-1.i386.rpm cannot be installed
```

Para obligar a RPM a “actualizar” de todas maneras, podemos usar la opción `--oldpackage`.

10.1.4. Consulta de paquetes

Como todo en la vida, hay cosas buenas y cosas malas, el manejo de las dependencias es lo más criticable de RedHat, no así las consultar que se pueden hacer a la base de datos de paquetes.

Consultar la base de datos de paquetes instalados se realiza mediante la opción `rpm -q`. Un uso simple es `rpm -q foo` lo que imprimirá el nombre, versión y número de desarrollo del paquete instalado foo:

```
$ rpm -q foo
rpm-2.0-1
```

En lugar de especificar el nombre del paquete, podemos usar las siguientes opciones con `-q` para especificar de qué paquete(s) queremos hacer una consulta. Éstas son llamadas *Opciones de Especificación de Paquetes* (Package Specification Options).

- `-a` consulta todos los paquetes instalados.
- `-f ¡file!` consultará el paquete al que pertenece ¡file!.
- `-p ¡packagefile!` consulta el paquete ¡packagefile!.

Hay varias formas de especificar qué información mostrar sobre los paquetes consultados. Las siguientes opciones son usadas para seleccionar la información que sea de nuestro particular interés. Son las llamadas *Opciones de Selección de Información*.

- `-i` presenta información del paquete como nombre, descripción, desarrollo, tamaño, fecha de construcción, fecha de instalación, vendedor, y otra información miscelánea.
- `-l` presenta la lista de archivos que el paquete “posee”.
- `-s` presenta el estado de todos los archivos del paquete. Hay sólo dos posibles estados: normal y perdido.
- `-d` presenta una lista de archivos marcados como documentación (páginas “man”, páginas “info”, README’s, etcétera).

- `-c` presenta una lista de archivos marcados como archivos de configuración. Estos son los archivos que personalizamos después de la instalación para adaptar el paquete a nuestras necesidades (sendmail.cf, passwd, inittab, etcétera).

Para aquellas opciones que presenten listas de archivos, usted puede añadir la opción `-v` para obtener la lista en el conocido formato `ls -l`.

10.1.5. Verificación de paquetes

Verificar un paquete es comparar la información sobre los archivos instalados desde un paquete con la misma información del paquete original. Entre otras cosas, verificar compara el tamaño, chequeo MD5, permisos, tipo, usuario y grupo de cada archivo.

`rpm -V` verifica un paquete. Podemos usar cualquiera de las *Opciones de Selección de Paquetes* (Package Selection Options) listadas para consultar, para especificar los paquetes que deseamos verificar. Un uso simple es `rpm -V foo` lo que verifica que todos los archivos del paquete `foo` estén como cuando fueron originalmente instalados. Por ejemplo:

Para verificar que un paquete contiene un archivo en particular:

```
rpm -Vf /bin/vi
```

Para verificar TODOS los paquetes instalados:

```
rpm -Va
```

Para verificar un paquete instalado con su correspondiente paquete RPM:

```
rpm -Vp foo-1.0-1.i386.rpm
```

Esto puede ser útil si sospechamos que nuestra base de datos de paquetes RPM está corrupta.

Si todo es verificado adecuadamente no habrá ninguna salida en pantalla. Si hay alguna discrepancia sí habrá información presentada. El formato de la salida es una cadena de 8 caracteres, un posible "c" denotando un archivo de configuración, y después el nombre del archivo. Cada uno de los 8 caracteres denota el resultado de la comparación de un atributo del archivo con el valor de ese atributo en la base de datos RPM. Un solo "." (punto) significa que el test ha sido pasado. Los siguientes caracteres denotan fallo de ciertas pruebas:

5 Chequeo MD5

S Tamaño del archivo

L Enlace simbólico

T Modificación de la fecha del archivo

D Dispositivo

U Usuario

G Grupo

M Modo (incluye permisos y tipos de archivo)

Si vemos alguna salida en pantalla, tendremos que decidir si borramos o reinstalamos el paquete o idear alguna manera de resolver el problema.

10.2. Administración de paquetes con yum

Yum (Yellow dog Updater) es una herramienta de administración de paquetes que resuelve de forma automática los problemas de dependencias que nos encontramos al instalar paquetes de forma tradicional con *rpm*.

El único inconveniente es que nuestro equipo tiene que contar con conexión a internet.

10.2.1. Actualización del sistema con yum

Para actualizar nuestro sistema contamos con la opción `update`:

```
yum update
```

10.2.2. Búsquedas de paquetes

Yum nos permita buscar paquetes en los repositorios¹ que tengamos configurados. Para realizar una búsqueda usamos la opción `search`:

```
yum search cualquier-paquete
```

Ejemplo:

```
yum search httpd
```

10.2.3. Consulta de información

Para obtener información contenida en un paquete en particular:

```
yum info cualquier-paquete
```

Ejemplo:

```
yum info httpd
```

¹Entenderemos como repositorio un servidor que contiene paquetes para CentOS

10.2.4. Instalación de paquetes

Para instalar un paquete, usamos la opción `install` y el nombre de paquete, si no conocemos cual es el nombre del paquete usamos la opción `search` descrita en la sección 10.2.2.

```
yum install cualquier-paquete
```

Ejemplo:

```
yum install httpd
```

10.2.5. Desinstalación de paquetes

Para desinstalar un paquete y todo lo que depende de él, usamos la opción `remove`.

```
yum remove cualquier-paquete
```

Ejemplo:

```
yum remove httpd
```

10.2.6. Listado de paquetes

Si deseamos ver todos los paquetes disponibles para instalación más los que tenemos instalados usamos:

```
yum list available | less
```

Si sólo queremos un listado de todos los paquetes instalados en el sistema usamos:

```
yum list installed | less
```

Con la opción `list updates` se listarán todos los paquetes instalados en el sistema y que pueden (deben) actualizarse:

```
yum list updates | less
```

10.2.7. Limpieza del sistema

Yum, como resultado de su uso, deja cabeceras y paquetes RPM almacenados en el del directorio `/var/cache/yum/`. Esto ocupa espacio considerable en nuestro preciado disco, por lo que si queremos depurar usamos:

```
yum clean all
```


Capítulo 11

Administración básica de los servicios de red

11.1. xinetd

El demonio *xinetd* es un superdaemon que controla el acceso a un subconjunto de servicios de red tales como FTP, IMAP y Telnet. También proporciona opciones de configuración específicas al servicio para el control de acceso, registro mejorado, redireccionamiento y control de utilización de recursos.

Cuando un host cliente intenta conectarse a un servicio de red controlado por *xinetd*, el superdaemon recibe la petición y verifica por cualquier regla de control de acceso wrappers TCP. Si se permite el acceso, *xinetd* verifica que la conexión sea permitida bajo sus propias reglas para ese servicio y que el servicio no esté consumiendo más de la cantidad de recursos o si está rompiendo alguna regla. Luego comienza una instancia del servicio solicitado y pasa el control de la conexión al mismo. Una vez establecida la conexión, *xinetd* no interfiere más con la comunicación entre el host cliente y el servidor.

11.1.1. Archivos de configuración xinetd

Los archivos de configuración para *xinetd* son los siguientes:

/etc/xinetd.conf El archivo de configuración global de *xinetd*.

/etc/xinetd.d/ El directorio que contiene todos los archivos específicos al daemon.

/etc/xinetd.conf

El archivo */etc/xinetd.conf* contiene parámetros de configuración generales los cuales afectan cada servicio bajo el control de *xinetd*. Se lee una vez cuando el servicio *xinetd* es iniciado, por esto para que los cambios de la configuración tomen efecto, el administrador debe reiniciar el servicio *xinetd*. Ejemplo del archivo */etc/xinetd.conf*:

```
defaults
{
    instances            = 60
    log_type             = SYSLOG authpriv
    log_on_success       = HOST PID
    log_on_failure       = HOST
    cps                  = 25 30
}

includedir /etc/xinetd.d
```

Donde:

instances Configura el máximo número de peticiones que xinetd puede manejar simultáneamente.

log_type Configura xinetd para usar la facilidad de registro authpriv, el cual escribe las entradas de registro al archivo `/var/log/secure`.

log_on_success Configura xinetd a registrar si la conexión es exitosa. Por defecto, la dirección IP del host remoto y el ID del proceso del servidor procesando la petición son grabados.

log_on_failure Configura xinetd para registrar si hay una falla de conexión o si la conexión no es permitida.

cps Configura xinetd para no permitir más de 25 conexiones por segundo a cualquier servicio dado. Si se alcanza este límite, el servicio es retirado por 30 segundos.

includedir /etc/xinetd.d/ Incluye las opciones declaradas en los archivos de configuración específicos del servicio localizados en el directorio `/etc/xinetd.d/`. Consulte a Sección 15.4.2 para más información sobre este directorio.

`/etc/xinetd.d/`

Los archivos en el directorio `/etc/xinetd.d/` contienen los archivos de configuración para cada servicio manejado por `xinetd` y los nombre de los archivos que se correlacionan con el servicio. Como sucede con `xinetd.conf`, este archivo es de sólo lectura cuando el servicio `xinetd` es arrancado. Para que los cambios tengan efecto, el administrador debe reiniciar el servicio `xinetd`.

El formato de los archivos en el directorio `/etc/xinetd.d/` usan las mismas convenciones que `/etc/xinetd.conf`. La razón principal por la que la configuración para cada servicio es almacenada en archivo separados es hacer más fácil la personalización y que sea menos probable afectar otros servicios.

Capítulo 12

Respaldos

12.1. Introducción

En la inmensa mayoría de los sistemas informáticos, los datos almacenados en el sistema tienen mucho mayor costo y son mucho más difíciles de recuperar que el sistema en sí. Entre los riesgos de pérdida de datos se cuentan *los errores en el software, la falla de equipos, el error humano, el daño intencional, las catástrofes naturales*. El respaldo de datos es la generación de una copia, en un momento determinado, de los datos del sistema con vistas a su eventual reposición en caso de pérdida. Todos los sistemas informáticos deben respaldarse cuidadosamente, en momentos predeterminados, siguiendo un cronograma preestablecido.

12.2. Dispositivos y Medios

Las causas de pérdida de datos generan los siguientes requisitos de respaldo:

- uso de un medio removible, utilizable en otra máquina; los daños o fallas pueden afectar varias partes de hardware al mismo tiempo, impidiendo la utilización de la máquina original.
- conservación de los respaldos en un lugar físico suficientemente apartado. Existen actualmente empresas para realizar respaldos vía Internet, pero la mayoría de las instituciones conservan sus respaldos localmente.

La inmensa mayoría de los dispositivos de respaldo son de tipo magnético. Esto los hace vulnerables a la proximidad de elementos generadores de campos magnéticos. Los respaldos deben mantenerse apartados de dispositivos tales como bocinas, transformadores, unidades de potencia ininterrumpida (UPSs), unidades de disco o disquetera no confinados en gabinetes metálicos, monitores aún apagados, detectores de metales como los usados en los aeropuertos. El campo magnético terrestre termina, con el tiempo, afectando los medios magnéticos de grabación; esto limita la duración efectiva de los respaldos; para períodos largos, se aconseja usar medios ópticos o regrabar periódicamente. Puede asumirse una duración de 3 años para los medios magnéticos.

Muchos fabricantes de unidades de respaldo proveen compresión incorporada, citando el almacenamiento y la velocidad de transferencia en la presunción optimista de un 2 a 1. En términos reales sólo puede asumirse la capacidad real en bytes de la cinta, y la velocidad de transferencia de esa misma cantidad de bytes. En principio no puede asegurarse compresión alguna sin conocimiento previo del tipo de datos.

Características de los principales medios de respaldo, en orden de capacidades crecientes:

Medio	Capacidad	Reuso	Acceso aleatorio	Comentarios
Disquete	1.44/2.8 MB	Sí	Sí	muy común; poca capacidad, incómodo; poca duración (2 años); útil para respaldo de archivos de configuración o transferencia de archivos chicos; alto costo por MB
Zip	100/250 MB	Sí	Sí	bastante común; varias interfaces de conexión; alto costo por MB
CD-R	650 MB	No	Sí	muy común; varias interfaces de conexión; menor duración que CDs pregrabados, mucho mayor que los medios magnéticos; buenos para datos permanentes, incómodo para respaldos regulares
CD-RW	650 MB	Sí	Sí	ventajas del CD-R; limitado en capacidad
DVD-R	4.7 a 17 GB	No	Sí	Aún poco común; U\$S 5 el de 4.7GB.
Jaz, Orb	2 GB	Sí	Sí	discos removibles; buena velocidad de transferencia
Cinta 8 mm	7 GB	Sí	No	cinta video formato chico; las unidades suelen ser llamadas Exabyte
Cinta 4 mm DAT/DDS	20 GB	Sí	No	DDS (Digital Data Storage) es similar al DAT (Digital Audio Tape) para audio; original 2 GB, DDS-4 en 20 GB; buena velocidad de transferencia; tamaño reducido
Disco fijo	40 GB	Sí	Sí	muy común; excelente transferencia, bajo costo; apto para crear espejos de discos; menor transportabilidad

Existen cintas de nueva tecnología, con mejoras en capacidad, precio, transferencia o duración: Travan (varios fabricantes), ADR (OnStream), DLT (Quantum), AIT (Sony), Mammoth (Exabyte); verificar soporte para el hardware en la versión de UNIX a utilizar. DAT y Exabyte son soluciones baratas para la mayoría de las empresas chicas y medianas; DLT, AIT y Mammoth están orientadas a grandes corporaciones o universidades.

Existen diversos tipos de equipo para cambio automático de volúmenes, de alto costo y con software propio, en capacidades de terabytes. Una adecuada partición en sistemas de archivo, un calendario adecuado y un poco de paciencia permiten respaldar un sistema con razonable esfuerzo.

12.3. Régimen de respaldo incremental

Los comandos tradicionales de respaldo y recuperación son *dump* y *restore*. Según los sistemas, pueden tener nombres similares (*ufsdump*, *ufsrestore* en Solaris). Otros comandos también tradicionales son *tar* y *cpio*, con múltiples opciones de control adaptables a variadas necesidades.

12.3.1. Respaldo de un sistema de archivos

El comando *dump* recorre el sistema de archivos haciendo una lista de los archivos modificados o nuevos desde una corrida anterior de *dump*; luego empaqueta todos esos archivos en uno solo y lo vuelca en un dispositivo externo tal como una cinta. Exhibe estas características:

- soporta multivolumen;
- soporta todo tipo de archivo, incluso de dispositivos;
- conserva permisos, dueños y fecha de modificación;
- maneja nombres largos y rutas de anidamiento profundo;
- maneja bien los archivos con huecos (bloques sin datos) producidos por algunos programas; en una copia común estos archivos se agrandan desmesuradamente;
- admite respaldo incremental.

dump maneja el sistema de archivos en crudo, leyendo la tabla de inodos para decidir qué archivos deben respaldarse. Esto aumenta su eficiencia, pero obliga a manejar cada sistema de archivos en forma independiente, e impide el respaldo de sistemas de archivos remotos tipo NFS. No obstante, es posible respaldar un sistema de archivos sobre una unidad de respaldo remota usando *rdump*.

12.3.2. Niveles de un respaldo incremental

El respaldo incremental se implementa asignando un nivel a cada respaldo, de 0 a 9. Un respaldo nivel 0 copia **todos** los archivos; un respaldo nivel 1 copia sólo los archivos modificados luego de la fecha del último respaldo nivel 0; un respaldo nivel 7 copia sólo los archivos modificados luego del último respaldo nivel 6. La recuperación de un sistema de archivos requiere reponer primero el respaldo nivel 0 y luego sucesivamente el último nivel 1, el último de nivel 2, y siguientes. Los números de niveles pueden no ser contiguos: se toma el nivel anterior más próximo existente como referencia. La información de *dump* se guarda en el archivo */etc/dumpdates*; en caso necesario, este archivo puede ser editado manualmente.

```
dump 0uf /dev/st0 /usr
```

crea un respaldo nivel 0 del sistema de archivos */usr* usando el dispositivo de cinta con rebobinado */dev/st0* (opción *f*) actualizando */etc/dumpdates* (opción *u*).

```
dump 3uf /dev/st0 /usr
```

análogo para un nivel 3.

```
dump 0uf /dev/nst0 /usr
```

crea un respaldo nivel 0 del sistema de archivos */usr* usando el dispositivo de cinta no rebobinado */dev/nrst0*, para reiterar el comando y colocar varios respaldos en una misma cinta. Los dispositivos de cinta suelen tener dos archivos de dispositivo, uno con rebobinado automático (*/dev/st0*) y otro sin rebobinado (*/dev/nst0*); ambos se refieren al mismo dispositivo físico. El manejo de la unidad de cinta se hace con el comando *mt*.

```
rdump 0uf pino:/dev/rtape /usr
```

crea un respaldo nivel 0 del sistema de archivos */usr* usando el dispositivo remoto */dev/rtape* en la máquina pino. El acceso a la cinta remota es controlado por el archivo *.rhosts* de la máquina remota. Si no se confía en la privacidad de la red puede convenir más implementar un túnel seguro con *ssh*.

El respaldo en cinta requiere conocer su tamaño y características. El fin de cinta (EOT, End Of Tape) es generalmente detectado, para habilitar los respaldos multivolumen. Un error en el largo de cinta o en la elección de dispositivo rebobinado puede arruinar el respaldo.

```
dump 5usdf 60000 6250 /dev/st0 /trabajos
```

indica un largo de cinta ficticio de 60000 pies (opción *s*, size), densidad de grabación 6250 dpi (opción *d*, densidad), para engañar una versión *dump* incapaz de reconocer cintas de más de 1.5 GB (DAT DDS-1); como además se comprime, se confía en la capacidad de *dump* para recibir la señal EOT en una cinta con capacidad en exceso.

```
ufsdump 0uf /dev/rmt2 /dev/rdisk/c0t3d0s5
```

crea un respaldo nivel 0 con el comando *ufsdump* (Solaris), del sistema de archivos sobre el dispositivo crudo */dev/rdisk/c0t3d0s5*; algunas versiones no aceptan el punto de montaje. Desgraciadamente, el comando *dump* en Solaris existe, pero su propósito es examinar archivos objeto, lo cual induce a error a muchos administradores honestos.

12.3.3. Respaldo en Solaris 8

Ejemplos del uso del comando *ufsdump*

Bytes necesarios para hacer el respaldo:

```
ufsdump S /dev/dsk/c0t3d0s0
```

Intruccion para respaldar un sistema de archivos

```
ufsdump 0ucf /dev/rmt/0n /dev/dsk/c0t1d0s0
```

donde:

```
0 -> nivel de respaldo (completo)
u -> actualiza el registro de respaldot (/etc/dumpdates)
c -> Cartucho
f -> arhichivo de respaldo
```

/dev/rmt/0n -> unidad de cinta
/dev/dsk/c0t1d0s0 -> sistema de archivos a respaldar

Comando para el manejo de cinta: *mt* (magnetic tape control)

```
mt [ -f tapename ] command ... [ count ]
```

```
mt -f /dev/rmt/0n eof
```

Pone una marca de fin de archivo

```
mt -f /dev/rmt/0n status
```

Da el informacion acerca de la cinta

```
mt -f /dev/rmt/0 fsf 2
```

Avanza dos bloques

```
mt -f /dev/rmt/0n bsf 2
```

Retrocede 2 bloques

```
mt -f /dev/rmt/0n rewind
```

Rebobina la cinta

```
mt -f /dev/rmt/0n offline
```

Rebobina la cinta y la expulsa de la unidad

12.3.4. Esquemas de respaldo

Los niveles de respaldo pueden elegirse arbitrariamente; sólo tienen sentido respecto de un respaldo de nivel menor. Un esquema de respaldo se define en función de

- actividad de cada sistema de archivos;
- capacidad del dispositivo de respaldo;
- redundancia deseada;
- cantidad de volúmenes;
- complejidad operativa.

Si el sistema de archivos cabe en un volumen, puede hacerse un nivel 0 diario (o semanal), con un grupo de cintas que se va reutilizando; cada N días, la cinta se conserva. Este esquema presenta redundancia masiva y es fácil para restaurar

Un esquema más optimizado consiste en realizar un nivel 9 diario (7 cintas), un nivel 5 semanal (5 cintas), un nivel 3 mensual (12 cintas), un nivel 0 cuando ya no alcanza un volumen o al menos una vez al año.

El esquema clásico más completo de respaldo tiene conexión con el algoritmo matemático de las Torres de Hanoi. Equilibra la aspiración de retener la mayor información posible por el mayor tiempo posible con limitaciones prácticas como el número de cintas y el tiempo disponible. Emplea los 9 niveles, el 0 se conserva, y reitera el ciclo cada 45 días. Este esquema suele ser demasiado complicado aún para sistemas grandes, aunque provee excelente redundancia; también es complicado restaurar.

```
>> 0 > 3 > 2 > 5 > 4 > 7 > 6 > 9 > 8
> 1A > 3 > 2 > 5 > 4 > 7 > 6 > 9 > 8
> 1B > 3 > 2 > 5 > 4 > 7 > 6 > 9 > 8
> 1C > 3 > 2 > 5 > 4 > 7 > 6 > 9 > 8
> 1D > 3 > 2 > 5 > 4 > 7 > 6 > 9 > 8
```

12.3.5. Elección de un esquema de respaldo

Dado que una gran parte de los archivos no cambian, el esquema incremental más simple ya elimina un gran volumen del respaldo diario. La inserción de niveles divide aún más finamente en grupos los archivos activos. El respaldo incremental permite respaldos más frecuentes con menos cintas, más alguna redundancia por repetición de archivos. El esquema de respaldo elegido surge de evaluar estas condiciones.

12.4. Recomendaciones generales

Las siguientes recomendaciones no son universales ni infalibles, pero surgen de la experiencia.

Respaldar todo desde la misma máquina: aunque es más lento, la facilidad de administración y la posibilidad de verificar la corrección del respaldo en todas las máquinas justifica su realización a través de la red. Se corre *rdump* en la máquina remota a respaldar, vía *rsh*, dirigiendo la salida hacia la unidad de respaldo en la máquina local. Al restaurar, deben tomarse en cuenta eventuales diferencias de sistema operativo; en algunos casos hay inversión de bytes, que pueden arreglarse con *dd*; esto no resuelve diferencias entre versiones de *rdump*.

Etiquetar las cintas: fecha, hora, máquina, sistema de archivos, número serial constituyen el mínimo absoluto. Una cinta no identificada sólo sirve para ser regrabada. Los sistemas de archivos */y* */usr* deben poder restaurarse sin referencia alguna a scripts o información en línea; la sintaxis exacta de los comandos, densidades, opciones y otros valores deben figurar en la documentación de la cinta. Un registro más completo se hace imprescindible cuando se respaldan muchos sistemas de archivos en un mismo volumen.

Intervalo de respaldos razonable: el sistema de archivos de usuarios puede respaldarse a diario en sistemas grandes, o semanalmente en sistemas chicos; la frecuencia de respaldo para otros sistemas de archivos dependerán del uso y la criticidad. El respaldo consume recursos y tiempo de operador; es responsabilidad del administrador del sistema balancear costos y riesgos al definir frecuencias de respaldo sobre cada sistema de archivos.

Elegir sistemas de archivo: según el movimiento, cada sistema de archivos puede tener un esquema diferente. Un grupo de archivos muy activo en un sistema de archivos inactivo puede copiarse

diariamente hacia otro sistema de archivos con respaldo diario. Sistemas de archivos de noticias, o el sistema de archivos */tmp*, no deben respaldarse; las noticias son efímeras, y */tmp* no debe contener nada importante.

Respaldo diario en un solo volumen: buscar un esquema o un medio para que el respaldo diario quepa en un solo volumen; es la única forma simple de realizar un respaldo diario, cargando la cinta a última hora y ejecutando el respaldo en cron. Recordar: el respaldo de varios sistemas de archivos en una cinta única requiere usar el dispositivo no rebobinado y documentar bien las cintas.

Crear sistemas de archivo acordes con el tamaño del medio de respaldo: con las capacidades actuales, es posible crear sistemas de archivo de tamaño razonable que quepan en una cinta. Debe haber una buena razón para crear sistemas de archivo muy grandes.

Mantener las cintas fuera del lugar: pero realmente en otro lugar, apartado del lugar de la instalación; hay un sin número de historias sobre pérdida de los respaldos conjuntamente con el sistema. El volumen actual de medios es suficientemente reducido como para trasladar un montón de información en un portafolios. Puede recurrirse a una institución especializada en conservación de datos o llevar las cintas a casa del gerente.

Seguridad del respaldo: el robo de un respaldo equivale al robo de toda la información vital de una organización. Las precauciones de seguridad con los respaldos debe ser tanta como la dispensada al propio sistema, con el agravante de que es más fácil llevarse unas cintas que la información en disco.

Limitar la actividad durante dump: la actividad en los archivos mientras se ejecuta *dump* puede ocasionar confusión en el momento de restaurar. Esto no es tan crítico en niveles superiores, pero sí en el nivel 0. Puede hacerse el respaldo en horas de escasa actividad, nocturnas o en fin de semana. Es preciso cuidar de no coincidir con los programas del sistema que modifican el sistema de archivos; éste debe permanecer estacionario mientras se realiza el respaldo. La solución más segura es hacer el respaldo en monousuario. En ocasiones especiales, como al encarar una actualización del sistema operativo, deberá uno armarse de paciencia, bajar la máquina a monousuario y respaldar el sistema en nivel 0. Existen programas especiales (archivadores o "filers") capaces de tomar un registro periódico del estado del sistema y resincronizar el respaldo. Debe considerarse esta posibilidad cuando no sea posible reducir la actividad del sistema en ningún momento.

Verificar el respaldo: hay también historias de respaldos tomados cuidadosamente que nunca pudieron restaurarse. Releer la cinta con *restore* para generar la tabla de contenido es una prueba razonable; probar recuperar un archivo en particular obliga a recorrer partes más alejadas de la cinta, ofreciendo una comprobación más sólida. Debe verificarse también que es posible restaurar desde varias cintas, que se pueden leer cintas grabadas tiempo atrás, y que es posible leer en otras unidades además de la habitual.

Vida útil de las cintas: como todo en el mundo, las cintas tienen una vida limitada, indicada por los fabricantes en cantidad de pasadas. Un respaldo, una restauración o un salto de archivos representan, cada uno, una pasada.

Crecimiento del sistema: el bajo precio de discos tiende a generar un aumento desordenado de la capacidad de almacenamiento. Imponer un criterio ordenado de crecimiento, con calificación de datos por criticidad y actividad, su separación racional en distintos sistemas de archivos, la concentración de información vital en un servidor confiable, son algunas medidas coactivas para alcanzar un respaldo confiable y humanamente posible.

12.5. Recuperación

12.5.1. Restaurar archivos

En este apartado se trata la restauración de un archivo o un grupo de archivos, en contraposición a la restauración de un sistema de archivos completo.

1. Determinar en qué cinta se encuentran los archivos a restaurar. Los usuarios generalmente buscan la última versión del archivo, pero no siempre es así. La existencia y ubicación del archivo dependen del esquema de respaldo empleado. Si se conservan catálogos (listas con todos los archivos respaldados en una fecha), la búsqueda se simplifica: basta con verificar si el archivo buscado se encuentra en el catálogo. Si no es así, se deberán revisar las cintas más probables según la fecha indicada por el usuario, o recorrer todo el conjunto desde el respaldo nivel 0 inmediato anterior.
2. Crear un directorio donde recuperar los archivos. Muchas versiones de restore requieren reponer la ruta entera de directorios para recuperar el archivo. No usar /tmp; su contenido será borrado en un rearranque imprevisto.
3. Si se han colocado varios archivos de respaldo en una misma cinta, se deberá consultar la documentación de ubicación de cada uno, determinar el lugar en que se encuentra el de interés, y usar el comando mt para ubicar el comienzo del archivo de respaldo.
4. Restaurar el archivo. Usar el comando complementario del respaldo: si se usó dump para respaldar, usar restore; si se usó rdump, usar rrestore.
5. Entregar el archivo al usuario. Se puede copiar el archivo hacia el directorio del usuario, verificando que no exista ya un archivo con ese nombre. En ningún caso debe sobrescribirse un archivo de otro usuario. Otra alternativa es dejarlo en el lugar de recuperación para que el usuario lo copie. En este caso, será preciso limpiar regularmente el directorio de recuperación.
6. Notificar al usuario.

Ejemplo completo de recuperación del archivo perdido.arch del usuario juanpe, con la cinta en la máquina roble:

Montar la cinta en la unidad de la máquina roble.

```
$ su -
```

ingresa como supervisor, pide contraseña.

```
# cd /var/tmp
```

la recuperación se hará en el directorio /var/tmp.

```
# rsh roble mt -f /dev/nrst1 fsf 3
```

salta hasta el 4o. archivo de respaldo en la cinta.

```
# rrestore xf roble:/dev/nrst1 >/usr/users/juanpe/docs/perdido.arch
```

ejecuta el comando e imprime mensajes; observar continuación del comando en la segunda línea.

```
# ls /var/tmp/usr/users/juanpe/docs perdido.arch
```

muestra la presencia del archivo recuperado.

```
# ls /usr/users/juanpe/docs otro1.arch otro2.arch
```

verifica que el archivo recuperado no existe en el directorio propio del usuario, para no reescribirlo.

```
# cp -p /var/tmp/usr/users/juanpe/docs/perdido.arch /usr/users/juanpe/docs
```

copia el archivo recuperado hacia el directorio propio del usuario.

```
# mail -s "Archivo recuperado" juanpe
Su archivo perdido.arch fue recuperado.
Se encuentra en su directorio, bajo docs.
Saludos,
  El Administrador.
.
```

Envía correo al usuario avisando la recuperación.

```
# exit $
```

Fin de la tarea.

restore admite la opción *i*, para uso interactivo: el comando lee el catálogo de la cinta; se recorren los archivos como si se tratara de un árbol de directorios común, usando *ls*, *cd* y *pwd*; se van seleccionando los archivos a restaurar con *add*; cuando se han seleccionado todos, indicando *extract* se los recupera de la cinta.

Ejemplo de recuperación interactiva. El indicador de supervisor # cambia a *restore*; al operar dentro del comando.

```
# rrestore if roble:/dev/nrst1
restore> ls
.:
arnoldo/ beiro/ juanpe/ lost+found/ vega/
restore> cd juanpe
restore> ls
carta01.txt core docs/ mbox perdido.arch varios/
restore> add perdido.arch
```

El archivo se agrega a la lista de archivos a recuperar. Agregar un directorio agrega todo su contenido.

```
restore> ls
carta01.txt core docs/ mbox perdido.arch* varios
```

El asterisco indica que está marcado para recuperar.

```
restore >extract
```

Muestra mensajes; si no se sabe en qué volumen está el archivo, debe comenzarse por el último y proceder hacia el principio; aquí asumimos saber que está en el primer volumen.

Specify next volume #: 1

Se realiza la extracción; pregunta si el directorio raíz de la cinta debe interpretarse como directorio corriente; se usa sólo al restaurar sistemas de archivo completos.

```
set owner mode for '.'? [yn] n
#
```

Sale de restore, fin de la tarea.

12.6. Restaurar sistemas de archivos

Antes de restaurar un sistema de archivos completo, se debe estar seguro de haber eliminado las causas que provocaron su destrucción.

1. Crear un sistema de archivos en la partición donde se va a restaurar; montarlo.
2. Cambiar al directorio raíz (punto de montaje) del nuevo sistema de archivos. Montar la primera cinta del último respaldo nivel 0. Arrancar la restauración con *restore r*. El comando pedirá las cintas sucesivas.
3. Al terminar de restaurar el nivel 0, continuar con los diferentes niveles en el mismo orden del esquema de respaldos empleado.

Ejemplo de restauración de un sistema de archivos a partir de un respaldo de 3 niveles.

```
# newfs /dev/dsk/c201d6s0 QUANTUM_PD1050S
# mount /dev/dsk/c201d6s0 /home
# cd /home
```

Crea el sistema de archivos, lo monta y se posiciona en él. Montar ahora la cinta 1 del último respaldo nivel 0 de /home.

```
# restore r
```

Montar las restantes cintas del respaldo nivel 0. Montar luego la primer cinta del respaldo nivel 1 siguiente.

```
# restore r
```

Montar las siguientes cintas del respaldo nivel 1. Montar luego la primer cinta del respaldo nivel 2 siguiente.

```
# restore r
```

Montar las siguientes cintas del respaldo nivel 2. Montar luego la primer cinta del respaldo nivel 3 siguiente.

```
# restore r
```

Esta secuencia repone el sistema de archivos a su estado original más cercano al momento de pérdida. En el esquema de respaldos empleado, la única diferencia es la mágica aparición de los archivos que fueron borrados. Hay versiones de restore que llevan registro de los archivos borrados.

En una actualización del sistema operativo, debe hacerse un respaldo nivel 0 antes de la actualización, efectuar luego la actualización cuidando de reponer los archivos de configuración necesarios en los sistemas de archivos afectados por la actualización. Una vez que todo esté funcionando, realizar inmediatamente un nuevo respaldo nivel 0. Esto es imprescindible para asegurar la coherencia de los siguientes niveles, ya que la actualización puede haber modificado fechas de archivos preexistentes.

12.6.1. Recuperación de datos en Solaris 8

El comando que se usa para recuperar archivos en solaris se llama *ufsrestore*

Para imprimir la tabla de contenido.

```
# ufsrestore tf /dev/rmt/0n
```

Para entrar en modo interactivo

```
# ufsrestore if /dev/rmt/0n
ufsrestore > help
ufsrestore > ls
ufsrestore > add archivo
ufsrestore > delete archivo
ufsrestore > extract
ufsrestore > marked //lista los archivos marcados
ufsrestore > quit
```

12.7. Otros comandos de respaldo

Los clásicos comandos *tar* (BSD) y *cpio* (System V) sirven para respaldar archivos, directorios y grupos diversos de archivos y directorios. Se usan frecuentemente para trasladar información, distribuir software o aún copiar árboles de directorios dentro de un mismo sistema, sobre todo por su capacidad de conservar dueños, permisos y fechas, no siempre posible con *cp*.

Copia de un árbol de directorios usando tar:

```
tar cf dir_origen | ( cd dir_destino ; tar xfp - )
```

Copia de un árbol de directorios usando cpio:

```
find dir_origen -depth -print | cpio -pdm dir_destino
```

Este comando es poco usado, habiendo cedido lugar a *tar*.

Al usar estos comandos verificar estas posibles limitaciones: soporte multivolumen, nombres de ruta largos, recuperación de errores, fijación de factor de bloqueo (20 por defecto). Muchas han sido levantadas, en particular en las versiones de GNU, pero conviene verificar en la documentación y comprobar en la práctica.

Cuando es preciso realizar transformaciones de datos es útil el comando *dd* (data duplicator). Sin parámetros, este comando sólo copia entrada en salida. Admite un cierto número de opciones que permiten cambiar formato de los datos o transferir entre distintos medios.

Copia de una cinta entre dos unidades:

```
dd if=/dev/rmt8 of =/dev/rmt9 cbs=16b
```

Copia de una cinta en una sola unidad:

```
dd if=/dev/rmt8 of =/tmp/cinta.archdev cbs=16b
```

cambiar la cinta;

```
dd if=/tmp/cinta.arch of =/dev/rmt8 cbs=16b
```

Conversión desde cinta con diferente orden de byte:

```
dd if=/dev/rst8 conv=swab | tar xf -
```

volcopy permite realizar una copia exacta de un sistema de archivos completo hacia otro dispositivo, eventualmente con cambio en el bloqueo. Está disponible en Solaris, HP-UX y Linux.

Para el control de la unidad de cinta, se usa el comando *mt*, especialmente útil cuando se graba más de un archivo en la misma cinta. Su sintaxis básica es

```
mt [-f dispositivo_cinta] comando [cantidad]
```

Los comandos de respaldo en cinta graban una señal de fin de archivo (EOF, End Of File) al terminar de ejecutar; esto permite ubicar un respaldo en particular desplazándose en la cinta. Entre los comandos admitidos por *mt* se destacan:

rew rebobinado de la cinta

offline saca de línea la unidad de cinta; en los modelos que lo permiten, eyecta el casete.

status muestra información de la cinta.

fsf [**cantidad**] avanza la cinta la cantidad de archivos indicada, 1 por defecto.

bsf [**cantidad**] retrocede la cinta la cantidad de archivos indicada, 1 por defecto.

12.8. Amanda

Amanda (Advanced Maryland Automatic Network Disk Archive) es un software para respaldo de red capaz de actuar en una LAN hacia una unidad de cinta ubicada en un servidor, cumpliendo todas las

exigencias deseables para un administrador. Usa *dump* y *restore* como comandos de base, pero también puede manejar *tar* de GNU y *smbtar* de Samba para respaldar datos de máquinas Windows. Corre en muchas versiones de UNIX, soporta extensa variedad de hardware, permite compresión sobre el cliente antes de la transferencia, graba registros completos del respaldo en la cinta. Amanda es software libre, escala bien, es muy configurable, evoluciona rápido; está siendo usado extensamente en todo el mundo.

Capítulo 13

Monitoreo del Sistema

13.1. Monitoreo de recursos

Como ya hemos mencionado una buena administración del sistema gira en torno al uso eficiente de los recursos, pero ¿cuales recursos?.

Antes de poder monitorear recursos, primero tenemos que conocer cuales recursos hay que supervisar. Todos los sistemas tienen disponibles los siguientes recursos:

- CPU
- Memoria
- Almacenamiento
- Ancho de banda

Las herramientas disponibles en CentOS para monitorear estos recursos son: *free*, *top*, *vmstat*, *Sysstat*.

13.1.1. free

El comando *free* muestra la utilización de la **memoria del sistema**.

```
[paco@centos ~]$ free
              total        used        free      shared    buffers     cached
Mem:          333176      210096      123080          0       18292     146648
-/+ buffers/cache:    45156      288020
Swap:          688120           0       688120
```

La columna *Mem*: muestra la utilización de la memoria física, mientras que la columna *Swap*: muestra la utilización del espacio de intercambio (swap) del sistema. La columna *-/+ buffers/cache*: muestra la cantidad de memoria actualmente dedicada a las memorias intermedias del sistema (buffers).

Esta herramienta nos es útil para determinar rápidamente si un problema relacionado con la memoria está en progreso actualmente. Aunque *free* tiene la opción de mostrar repetidamente los números de utilización de memoria a través de su opción *-s*, la salida se desplaza, haciendo difícil detectar cambios en la utilización de memoria.

13.1.2. top

top es una herramienta completa que nos monitorea: utilización del CPU, estadísticas de procesos, utilización de memoria es decir todo. Además, a diferencia de *free* command, el comportamiento predeterminado de *top* es el de ejecutarse de forma continua.

```
Tasks:  57 total,   1 running,  56 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.4% us,  1.5% sy,   0.1% ni,  97.7% id,   0.4% wa,   0.0% hi,   0.0% si
Mem:    333176k total,  210416k used,  122760k free,   18460k buffers
Swap:   688120k total,    0k used,   688120k free,  146696k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	16	0	2580	560	480	S	0.0	0.2	0:01.41	init
2	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
3	root	5	-10	0	0	0	S	0.0	0.0	0:01.53	events/0
4	root	5	-10	0	0	0	S	0.0	0.0	0:00.02	khelper
5	root	15	-10	0	0	0	S	0.0	0.0	0:00.00	kacpid
18	root	5	-10	0	0	0	S	0.0	0.0	0:00.51	kblockd/0
28	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pdflush
29	root	15	0	0	0	0	S	0.0	0.0	0:01.16	pdflush
31	root	7	-10	0	0	0	S	0.0	0.0	0:00.00	aio/0
19	root	15	0	0	0	0	S	0.0	0.0	0:00.00	khubd
30	root	25	0	0	0	0	S	0.0	0.0	0:00.00	kswapd0
105	root	25	0	0	0	0	S	0.0	0.0	0:00.00	kseriod
178	root	25	0	0	0	0	S	0.0	0.0	0:00.00	scsi_eh_0
192	root	6	-10	0	0	0	S	0.0	0.0	0:00.00	kmirrord
193	root	6	-10	0	0	0	S	0.0	0.0	0:00.00	kmir_mon
201	root	15	0	0	0	0	S	0.0	0.0	0:03.26	kjournald
1082	root	6	-10	2100	440	368	S	0.0	0.1	0:00.22	udevd
1411	root	5	-10	2368	1076	784	S	0.0	0.3	0:00.18	dhclient
1686	root	7	-10	0	0	0	S	0.0	0.0	0:00.00	kauditd
1738	root	25	0	0	0	0	S	0.0	0.0	0:00.00	kjournald
2235	root	15	0	3588	1052	760	S	0.0	0.3	0:00.18	dhclient
2272	root	16	0	2648	628	536	S	0.0	0.2	0:00.67	syslogd

La pantalla se divide en dos secciones. La parte superior contiene información relacionada con el estatus general del sistema ¿ tiempo ejecutándose, carga promedio, cuentas de procesos, estado del CPU y estadísticas de utilización para la memoria y el espacio de intercambio. La sección de abajo muestra estadísticas a nivel de procesos. Es posible cambiar lo que se muestra mientras *top* se ejecuta. Por ejemplo,

por defecto top muestra procesos activos y ociosos. Para mostrar solamente procesos activos o que no estén ociosos, presione [i]; otro toque lo retorna al modo de visualización predeterminado.

13.1.3. vmstat

Con *vmstat*, es posible obtener una vista general de los procesos, memoria, swap, E/S, sistema y actividad de CPU:

```
[paco@centos ~]$ vmstat
procs -----memory----- ---swap-- -----io----- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us  sy id wa
  0  0       0 122440 18832 146720    0    0     9     4 1003   42  0  1 98  0
```

La primera línea divide los campos en seis categorías, incluyendo procesos, memoria, swap, E/S, sistema y estadísticas relacionadas al CPU. La segunda línea identifica aún más los contenidos de cada campo.

Los campos relacionados a procesos son:

- **r** El número de procesos ejecutables esperando para acceder al CPU
- **b** El número de procesos en un estado dormido continuo

Los campos relacionados a la memoria son:

- **swpd** La cantidad de memoria utilizada
- **free** La cantidad de memoria libre
- **buff** La cantidad de memoria utilizada por las memorias intermedias
- **cache** La cantidad de memoria utilizada como caché de páginas

Los campos relacionados a swap son:

- **si** La cantidad de memoria intercambiada desde el disco
- **so** La cantidad de memoria intercambiada hacia el disco

Los campos relacionados con E/S son:

- **bi** Los bloques enviados a un dispositivo de bloques
- **bo** Los bloques recibidos desde un dispositivo de bloques

Los campos relacionados al sistema son:

- **in** El número de interrupciones por segundo
- **cs** El número de cambios de contexto por segundo

Los campos relacionados al CPU son:

- **us** El porcentaje de tiempo que el CPU ejecutó código de nivel del usuario
- **sy** El porcentaje de tiempo que el CPU ejecutó código de nivel del sistema
- **id** El porcentaje de tiempo que el CPU estaba desocupado
- **wa** Esperas de E/S

13.1.4. Sysstat

Las herramientas vistas hasta el momento, solo nos ayudan para conocer el estado actual del sistema. Si lo que necesitamos es llevar un historico del desempeño del equipo el conjunto de herramientas *sysstat* nos resuelve el problema.

Sysstat contiene las siguientes herramientas relacionadas con reunir estadísticas de E/S y CPU:

iostat Muestra una descripción general de la utilización del CPU, junto con las estadísticas de E/S para uno o más unidades de disco.

mpstat Muestra estadísticas de CPU más complejas.

Sysstat también contiene herramientas para reunir datos de utilización de los recursos y crear informes diarios basados en esos datos. Estas herramientas son:

sadc Conocida como el coleccionador de datos de actividad del sistema, *sadc* reúne información sobre la utilización de recursos y la escribe a un archivo.

sar Los informes *sar*, producidos a partir de los archivos creados por *sadc*, se pueden generar interactivamente o se pueden escribir a un archivo para un análisis más intensivo.

13.1.5. iostat

El comando *iostat* en su forma más básica proporciona una descripción general de las estadísticas del CPU y E/S de disco:

```
[root@centos ~]# iostat
Linux 2.6.9-34.EL (centos)      13/07/06

cpu-med:  %user   %nice   %sys  %iowait   %idle
           0.44    0.05    1.43    0.40    97.68
```

Device:	tps	Blq_leid/s	Blq_escr/s	Blq_leid	Blq_escr
hdc	0.00	0.05	0.00	848	0
sda	0.73	17.91	8.72	315063	153356
sda1	0.03	0.05	0.00	944	4
sda2	1.45	17.80	8.72	313151	153352
dm-0	1.43	17.76	8.72	312402	153352
dm-1	0.00	0.02	0.00	360	0

Debajo de la primera línea (la cual contiene la versión del kernel del sistema y el nombre del host, junto con la fecha actual), *iostat* muestra una vista general de la utilización promedio del CPU desde el último arranque. El informe de utilización del CPU incluye los porcentajes siguientes:

- Porcentaje de tiempo en modo usuario (ejecutando aplicaciones, etc.)
- Porcentaje de tiempo en modo usuario (para procesos que han alterado su prioridad de planificación usando nice(2))
- Porcentaje de tiempo en modo kernel
- Porcentaje de tiempo ocioso

Debajo del informe de utilización del CPU está el informe de utilización de dispositivos. Este informe contiene una línea para cada dispositivo en el sistema e incluye la información siguiente:

- La especificación de dispositivos, mostrada como dev;major-number¿-sequence-number, donde ;major-number¿ es el número principal ("major") del dispositivo[1] y ;sequence-number¿ es un número secuencial comenzando desde cero.
- El número de transferencias (u operaciones de E/S) por segundo.
- El número de bloques de 512 bytes leídos por segundo.
- El número de bloques de 512 bytes escritos por segundo.
- El número total de bloques de 512 bytes leídos.
- El número total de bloques de 512 bytes escritos.

Esto es solamente un muestra de la información que se puede obtener usando *iostat*. Para más información, consulte la página man de *iostat*(1).

13.1.6. mpstat

El comando *mpstat* aparece primero sin diferencias con el informe de utilización de CPU producido por *iostat*:

```
[root@centos ~]# mpstat
Linux 2.6.9-34.EL (centos)      14/07/06
00:06:14    CPU   %user   %nice %system %iowait   %irq   %soft   %idle   intr/s
00:06:14    all    0.38    0.04   1.28    0.35    0.01    0.00   97.94   1002.83
```

Con la excepción de una columna adicional mostrando las interrupciones por segundo manejadas por el CPU, no hay diferencia real. Sin embargo, la situación cambia si se utiliza la opción de `mpstat, -P ALL`.

```
[root@centos ~]# mpstat -P ALL
Linux 2.6.9-34.EL (centos) 14/07/06

00:06:08 CPU %user %nice %system %iowait %irq %soft %idle intr/s
00:06:08 all 0.38 0.04 1.28 0.35 0.01 0.00 97.94 1002.83
00:06:08 0 0.38 0.04 1.28 0.35 0.01 0.00 97.94 1002.83
```

En sistemas multiproceso, `mpstat` permite desplegar de forma individual la utilización de cada CPU, haciendo posible determinar que tan efectivamente se utiliza cada CPU.

Apéndice A

Resumen de comandos básicos en Unix

A.1. Comandos de Linux

Comando/Sintaxis	Descripción	Ejemplos
<code>cat fich1 [...fichN]</code>	Concatena y muestra un archivos archivos	<code>cat /etc/passwd</code> <code>cat dict1 dict2 > dict</code>
<code>cd [dir]</code>	Cambia de directorio	<code>cd /tmp</code>
<code>chmod permisos fich</code>	Cambia los permisos de un archivo	<code>chmod +x miscript</code>
<code>chown usuario:grupo fich</code>	Cambia el dueño un archivo	<code>chown nobody miscript</code>
<code>cp fich1...fichN dir</code>	Copia archivos	<code>cp foo foo.backup</code>
<code>diff [-e]arch1 arch2</code>	Encuentra diferencia entre archivos	<code>diff foo.c newfoo.c</code>
<code>du [-sabr] fich</code>	Reporta el tamaño del directorio	<code>du -s /home/*</code>
<code>file arch</code>	Muestra el tipo de un archivo	<code>file arc_desconocido</code>
<code>find dir test acción</code>	Encuentra archivos.	<code>find . -name "*.bak" -print</code>
<code>grep [-cino] expr archivos</code>	Busca patrones en archivos	<code>grep mike /etc/passwd</code>
<code>head -count fich</code>	Muestra el inicio de un archivo	<code>head prog1.c</code>
<code>mkdir dir</code>	Crea un directorio.	<code>mkdir temp</code>
<code>mv fich1...fichN dir</code> <code>mv fich1 fich2</code>	Mueve un archivo(s) a un directorio Renombra un archivo.	<code>mv a.out prog1</code> <code>mv *.c prog_dir</code>
<code>less / more fich(s)</code>	Visualiza página a página un archivo. less acepta comandos vi.	<code>more muy_largo.c</code> <code>less muy_largo.c</code>
<code>ln [-s] fich acceso</code>	Crea un acceso directo a un archivo	<code>ln -s /users/mike/.profile .</code>
<code>ls</code>	Lista el contenido del directorio	<code>ls -l /usr/bin</code>
<code>pwd</code>	Muestra la ruta del directorio actual	<code>pwd</code>
<code>rm fich</code>	Borra un fichero.	<code>rm foo.c</code>
<code>rm -r dir</code>	Borra un todo un directorio	<code>rm -rf prog_dir</code>
<code>rmdir dir</code>	Borra un directorio vacío	<code>rmdir prog_dir</code>
<code>tail -count fich</code>	Muestra el final de un archivo	<code>tail prog1.c</code>
<code>vi fich</code>	Edita un archivo.	<code>vi .profile</code>

Cuadro A.1: Comandos Linux/Unix de manipulación de archivos y directorios

Comando/Sintaxis	Descripción	Ejemplos
at [-lr] hora [fecha]	Ejecuta un comando mas tarde	at 6pm Friday < miscript
cal [[mes] año]	Muestra un calendario del mes/año	cal 1 2025
date [mmddhhmm] [+form]	Muestra la hora y la fecha	date
echo string	Escribe mensaje en la salida estándar	echo "Hola mundo"
finger usuario	Muestra información general sobre un usuario en la red	finger nn@maquina.aca.com.co
id	Número id de un usuario	id usuario
kill [-señal] PID	Matar un proceso	kill 1234
man comando	Ayuda del comando especificado	man gcc man -k printer
passwd	Cambia la contraseña.	passwd
ps [axiu]	Muestra información sobre los procesos que se están ejecutando en el sistema	ps -ux ps -ef
who / rwho	Muestra información de los usuarios conectados al sistema.	who

Cuadro A.2: Comandos Linux/Unix más frecuentes

Linux	DOS	Significado
cat	type	Ver contenido de un archivo.
cd, chdir	cd, chdir	Cambio el directorio en curso.
chmod	attrib	Cambia los atributos.
clear	cls	Borra la pantalla.
ls	dir	Ver contenido de directorio.
mkdir	md, mkdir	Creación de subdirectorio.
more	more	Muestra un archivo pantalla por pantalla.
mv	move	Mover un archivo o directorio.
rmdir	rd, rmdir	Eliminación de subdirectorio.
rm -r	deltree	Eliminación de subdirectorio y todo su contenido.

Cuadro A.3: Equivalencia de comandos Linux/Unix y DOS

Bibliografía

- [osdi] ANDREW S. TANENBAUM - VRIJE UNIVERSITEIT AMSTERDAM. "Operating Systems Design and Implementation". Prentice Hall (Third Edition). 2006
- [lrhl] BILL MCCARTY. "Learning Red Hat Linux". O'Reilly (Third Edition). 2003
- [max-rpm] EDWARD C. BAILEY. "Maximun RPM - Taking the RPM Package Manager to the Limit". Red Hat (2 Edition). 2000
- [rhel-isa] REDHAT. "Introducción a la administración de sistemas". RedHat (Red Hat Enterprise Linux 4). Febrero 2005
- [ldd] JONATHAN CORBET, GREG KROAH-HARTMAN, ALESSANDRO RUBINI. "Linux: Device Drivers". O'Reilly (Versión 3). Febrero 2005
- [man] MANUALES DE LINUX. *bash*, *mount*, *mttools*, *rpm*,... Todas las páginas de manual que han sido necesarias para el documento.