WIKIUNIX

CONTENIDO TEÓRICO

NOELIA SALES MONTES

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

TUTORIAL EN FORMATO WIKI

SOBRE SISTEMAS OPERATIVOS UNIX

CON PLATAFORMA DE PRUEBA

VERSIÓN ESTÁTICA

Noelia Sales Montes

26 de febrero de 2010

Agradecimientos

Me gustaría agradecer y dedicar este documento...

- A Pablo y Rosa, que han estado conmigo en los momentos buenos y en los no tan buenos y que me han apoyado siempre.
- A Manuel, que me sugirió este proyecto que al final ha terminado apasionándome.
- A Emilio, que cada día me hace aprender algo más del gran mundo de los wikis.
- A Fabián, que sigue aportando buenas ideas para mejorar el proyecto.
- A la OSLUCA, que ha accedido a dejar un hueco en su servidor para este proyecto y en particular a Jose, por haberme ayudado cada vez que lo necesitaba.

Licencia

Notación y formato

A continuación se expone la notación utilizada en este documento para distinguir determinados elementos:

- Al referirnos a una orden ejecutada en la terminal, utilizaremos la notación:
  `orden ejecutada`
  `más órdenes ejecutadas`
- Al referirnos a una orden ejecutada por el administrador del sistema, utilizaremos la notación:
  `orden ejecutada`
  `más órdenes ejecutadas`
- Para mostrar texto resaltado, empleamos:
  texto resaltado

WikiUNIX. Versión en PDF.

Es necesario advertir que este documento no es más que una versión estática del wiki sobre sistemas operativos UNIX.
En este documento no se encuentra toda la información del wiki (es un contenido muy extenso): tan sólo se han incluido los artículos principales y unos anexos. A continuación se muestra un listado con todos los artículos "útiles" que se pueden encontrar en dicha página.

- /etc/group
- /etc/passwd
- /etc/shadow
- /home
- AWK
- Acceso servidor utilizando SSH
- Admon. del sistema de archivos
- Admon. sistema de archivos/Ejercicios[1]
- Admon. sistema de archivos/Ejemplos[2]
- Administrador
- Agregar maquina.sh
- Anacron
- Apropos
- Ar
- Arch
- Arranque
- Arranque y parada del sistema
- Arranque y parada del sistema/Ejercicios
- Asignación de cuotas de disco
- At
- Atq
- Atrm
- Awk
- Ayuda
- Ayuda en UNIX
- Backup-remoto.sh
- Backup.py
- Backup.sh
- Banner
- Basename
- Bash
- Batch
- Bunzip2
- Bzip2
- Cal
- Caracteres comodines
- Características
- Carácter comodín
- Cat
- Cd
- Cdp
- Chage
- Chext.py
- Chfn
- Chgrp
- Chmod
- Chown
- Chpasswd
- Chroot
- Chsh
- Chvt
- Clasificación
- Clear
- Cmp
- Col
- Colrm
- Column
- Comm
- Conceptos básicos
- Conceptos básicos/Ejercicios
- Consola
- Control.py
- Copias de seguridad
- Copias de seguridad automatizadas
- Cp
- Cpio
- Cron
- Crond
- Csplit
- Cut
- Date
- Dd
- Deallocvt
- Deamon
- Demonio
- Df

[1] El nombre real de este artículo es "Administración del sitema de archivos/Ejercicios"
[2] El nombre real de este artículo es "Administración del sitema de archivos/Ejemplos"

- Diff
- Directorio casa
- Directorio de trabajo
- Directorio padre
- Dirname
- Echo
- Egrep
- Env
- Estado maquinas.sh
- Estructura/Ejercicios
- Familia UNIX
- Fichero oculto
- Find
- Free
- Gawk
- Grep
- Groupdel
- Groups
- Gzip
- Head
- History
- Horas.sh
- Id
- Igawk
- Init
- Instalación de un servidor FTP con vsftpd
- Jerarquía de directorios
- Kernel
- LILO
- Latex.sh
- Less
- Ln
- Logname
- Ls
- Mailx
- Malas prácticas en un sistema UNIX
- Man
- Marca UNIX
- Mesg
- Mkdir
- Modos de funcionamiento
- More
- Namei
- Newusers
- Nivel:Principiante
- Nivel:Usuario habitual
- Dir
- Directorio de inicio
- Directorio oculto
- Directorio raíz
- Du
- Ed
- Ejecutar script automáticamente
- Espacio montado.sh
- Estructura
- FHS
- Fgrep
- File
- Finger
- GRUB
- Gestionar espacio de swap[3]
- Groupadd
- Groupmod
- Gunzip
- Halt
- Help
- Histórico del sistema
- Hostname
- Identificador de proceso
- Info
- Instalación de software
- Ipreal.py
- Join
- Kill
- Last
- Ldd
- Liberar memoria cache
- Login
- Look
- Mail
- Main Page
- Malas prácticas y consejos útiles
- Manpath
- Mc
- Mev
- Mknod
- Monitorización de recursos
- Mv
- Newgrp
- Nice
- Nivel:Unixero profesional
- Nivel:Usuario iniciado

[3] El nombre real de este artículo es “Gestionar espacio de memoria de intercambio”

Índice general

Índice de figuras

Índice de cuadros

Parte I

Nivel principiante

Capítulo 1

Características de UNIX

En esta sección se introducen los sistemas operativos que se encuendran bajo la familia Unix, especificando sus características	
TIEMPO:	1 hora
NIVEL:	Principiante (página 3)
INFORMACIÓN EXTRAÍDA DE:	What is UNIX? ✒ The Unix Philosophy: A Brief Introduction
PRERREQUISITOS:	UNIX

Este artículo consta de las siguientes secciones:

1.1. Introducción

UNIX es un sistema operativo portable, multitarea y multiusuario, cuyo desarrollo comenzó en 1969 a manos de un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Por normal general, el término UNIX tradicional suele emplearse para describir a Unix o a un sistema operativo que cuenta con las características de UNIX Versión 7 o UNIX System V.

A continuación, vamos a profundizar en dichas características comunes.

1.2. Portabilidad

La portabilidad es uno de los conceptos clave en la programación de alto nivel. Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

El prerrequisito para la portabilidad es la abstracción generalizada entre la aplicación lógica y las interfaces del sistema. Cuando un software se puede compilar en diversas plataformas (x86, IA64, amd64, etc.), se dice que es multiplataforma. Esta característica es importante para el desarrollo de reducción costos, cuando se quiere hacer una misma aplicación.

En algunos casos el software es "independiente" de la plataforma y puede ejecutarse en plataformas diversas sin necesidad de ser compilado específicamente para cada una de ellas, a este tipo de software se le llama interpretado, donde un "interprete" traduce (propiamente interpreta) las intrucciones a tiempo de ejecución para que sean entendidas por diferentes plataformas.

1.3. Multitarea

El sistema permite que los usuarios estén ejecutando varias aplicaciones simultáneamente utilizando la técnica de tiempo compartido. Para ello se aplican los diferentes algoritmos de planificación como veremos más adelante.

1.4. Multiusuario

En general se le llama multiusuario a la característica de un sistema operativo o programa que permite proveer servicio y procesamiento a múltiples usuarios simultáneamente (tanto en paralelismo real como simulado).

En contraposición a los sistemas monousuario, que proveen servicio y procesamiento a un solo usuario, en la categoría de multiusuario se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, que comparten los mismos recursos. Actualmente este tipo de sistemas se emplean especialmente en redes, pero los primeros ejemplos de sistemas multiusuario fueron sistemas centralizados que se compartían a través del uso de múltiples dispositivos de interfaz humana (e.g. una unidad central y múltiples pantallas y teclados).

1.5. Otras características

Además de las características básicas, es interesante observar las que vamos a describir a continuación, porque son las que lo distinguen más específicamente de otras familias de sistemas.

1.5.1. Algoritmo por prioridades o multinivel

Es uno de los más complejos y eficaces. Asigna los tiempos de ejecución de la UCP según una lista de prioridades. En cada una de estas listas, el sistema operativo incluirá aquellos procesos a los que se les haya asignado esa prioridad. El tiempo de ejecución del procesador se irá destinando, en primer lugar, de forma secuencial a los procesos de mayor nivel. Terminados éstos, se ejecutarán los procesos del nivel inferior, y así sucesivamente hasta los procesos del nivel más bajo.

1.5.2. Memoria virtual

Esta técnica permite a los usuarios del sistema ejecutar programas, de tal forma que dé la sensación de que toda la memoria RAM es para ellos. Concretamente en Unix/Linux se utiliza la paginación de la memoria. Esta técnica es la que utilizan la mayoría de los sistemas operativos multiusuario. Dividen la memoria en páginas al igual que los programas y de esta forma se realiza el intercambio entre disco y RAM para ejecutar los mismos.

1.5.3. Sistema de archivos jerárquico

Utiliza un sistema de archivos[1] en forma de árbol invertido. La particularidad esencial frente a otros sistemas es que UNIX no gestiona dispositivos (como una disquetera) de forma directa. Gestiona los dispositivos como si fueran directorios, de tal forma que cuando estemos accediendo al directorio asociado a una disquetera, en realidad lo que estarás haciendo es acceder a la información contenida en el disquete.

1.5.4. Comunicación con otros ordenadores

Un sistema UNIX permite no solamente trabajar con él en una máquina, sino también conectar varios ordenadores centrales UNIX entre sí, de tal forma que cada usuario tenga acceso a la información contenida en todos ellos. La conexión se realiza a través del conjunto de protocolos y servicios que ofrece TCP/IP. Gracias a él puedes ejecutar programas en máquinas UNIX que estén a varios kilómetros de distancia entre sí; enviar correo electrónico de unos equipos a otros; realizar conversación directa entre dos usuarios, etcétera.

1.5.5. Sistemas de seguridad

Es una de las características más importantes, ya que la información a la que un usuario puede tener acceso puede limitarse de forma sencilla. De este modo, el administrador del sistema operativo, a través de palabras clave (para archivos empaquetados o comprimidos) o mediante la asignación de derechos a los usuarios, hace que la información contenida en un servidor Unix esté totalmente protegida de piratas o usuarios no deseados.

[1] Más información acerca de los sistemas de archivos en los capítulos 6 y 12

1.6. Filosofía

UNIX es mucho más un sistema operativo: se caracteriza por tener una filosofía y una forma diferente de ver las cosas. La complejidad de uso que le achacan los defensores de otros sistemas se percibe por parte de sus entusiastas como uno de sus mayores atractivos. Además trabajando en Unix se tiene la posibilidad de conocer el funcionamiento interno del sistema, modificar sus partes, añadir nueva funcionalidad y se puede confiar en su enorme fiabilidad y ausencia de cuelgues totales.

UNIX tiene una cultura propia, un estilo de programación característico y lleva consigo una potente filosofía de diseño. Entender el mundo y la comunidad que rodea a UNIX es esencial para poder adentrarse en este sistema. En ocasiones se acusa a esta comunidad de cerrada y reticente al cambio, por lo que para entrar en este mundo resulta imprescindible familiarizarse con su forma de trabajo.

La filosofía de UNIX se puede resumir en una única frase descriptiva: “Do one thing and do it well”. Sin embargo, se deben tener en cuenta una especie de normas generales tales como:

- Haz que cada programa haga una cosa y la haga bien.
- Para llevar a cabo una nueva tarea escribe un programa nuevo. No compliques uno viejo añadiendo nueva funcionalidad.
- Escribe tu programa teniendo en cuenta que su salida probablemente sea la entrada de otro programa. No llenes la salida estándar con información innecesaria ni utilices formatos raros.
- Guarda los datos en archivos de texto plano. Si necesitas seguridad, confía en los permisos.
- Usa nombres cortos y en minúscula.
- Si no es imprescindible, no pidas nada de forma interactiva: haz que el usuario suministre los datos por línea de comandos en la llamada.
- Haz partes simples conectadas mediante interfaces limpias y bien definidas.
- Céntrate en los datos. Si has elegido las estructuras adecuadas y organizado todo correctamente, los algoritmos serán evidentes.
- Claridad mejor que complejidad. La solución más simple es frecuentemente la mejor: añade complejidad solo donde sea indispensable.
- Portabilidad mejor que eficiencia.
- Piensa en paralelo. Hay otros procesos en el mundo, incluso instancias de tu mismo programa funcionando al mismo tiempo.
- Haz un programa grande sólo cuando haya quedado demostrado que no puede hacerse con uno pequeño.
- Si tu programa no tiene nada interesante que decir, que no diga nada.
- No existe una única manera de hacerlo. Cada problema tiene múltiples soluciones.
- Diseña pensando en el futuro, está más cerca de lo que piensas.

- ✦ Unix no pide por favor.

En caso de duda la regla universal a tener en cuenta es siempre la norma **KISS**: *Keep it simple, stupid!*.

Resumiendo: no compliques las cosas si pueden hacerse de una forma simple, y seguramente haya una forma muy simple de hacerlo.

Capítulo 2

Clasificación de las familias de Unix

Distinción entre las subfamilias de sistemas operativos que derivan del sistema inicial Unix	
TIEMPO:	45 minutos
NIVEL:	Principiante (página 3)
INFORMACIÓN EXTRAÍDA DE:	Unix, Linux, and variant history ✒ *Introduction to Unix* - Frank G. Fiamingo, Linda DeBula, Linda Condron - University Technology Services, The Ohio State University
PRERREQUISITOS:	Ninguno

Este artículo consta de las siguientes secciones:

2.1. Historia

Los Laboratorios Bell fueron los responsables del desarrollo de Unix como un proyecto de investigación privado dirigido por un pequeño grupo de personas que empezó en 1969. El objetivo del grupo fue diseñar un sistema operativo que cumpliera los siguientes requisitos:

- ✦ Que fuera simple y elegante.
- ✦ Que estuviera escrito en un lenguaje de alto nivel.
- ✦ Que permitiera reutilizar el código.
- ✦ UNIX tenía una relativamente pequeña parte de su código escrita en ensamblador (la que se encargaba de gestionar el hardware, más conocida como kernel y el resto del código escrito en C.

A medida que el desarrollo avanzaba se realizaban pequeñas modificaciones al código del kernel (dependiendo de la plataforma) y se desarrollaron muchas utilidades en C. A través de esta evolución el kernel y el software asociado se extendieron hasta que un sistema operativo completo se desarrolló "sobre" el kernel.

2.2. Cronología

1969

- Ken Thompson de los Laboratorios Bell escribe la primera versión de lo que se llamará posteriormente Unix. Funcionaba en una máquina DEC PDP-7.

1970

- Thompson y Dennis Ritchie portan Unix a una máquina DEC PDP-11/20, lo que propició que Ritchie diseñara y desarrollara el primer compilador de C.

1973

- Ritchie y Thompson reescriben el kernel de UNIX en C.

1974-1977

- El código de UNIX se distribuye libremente a las universidades. Como resultado, se populariza en el ámbito académico.

1978

- Se distribuye la versión 7 de UNIX. Esta versión se diseña para ser portable a varias arquitecturas hardware.
- ATT anuncia que van a empezar a cobrar por la distribución del código fuente de UNIX. Como consecuencia, la versión 7 forma la base de todas las versiones de UNIX actuales.

1979

- Ante el anuncio de ATT de su intención de comercializar UNIX, la Universidad de California (Berkeley) crea su propia variante: BSD UNIX.
- Las versiones BSD más influyentes serán BSD 4.2 (1983) y 4.3 (1987).
- El desarrollo patrocinado por DARPA de internet fue sobre BSD.

- ✦ La mayoría de las casas que comercializarán UNIX (Sun con su SunOS, DEC con su Ultrix, etc.) se basan en BSD.

1980

- ✦ Microsoft introduce XENIX.

1983

- ✦ ATT lanza su sistema comercial UNIX System V.
- ✦ Aparece la versión 4.2 de BSD que incluye una implementación completa de la familia de protocolos TCP/IP.

1987

- ✦ Se lanza la versión 3 de ATT Unix System V, que incluye STREAMS, TLI y RFS. Ésta es la versión en la que varios fabricantes de hardware como HP (HP-UX) e IBM (AIX) se basarán.
- ✦ También se lanza BSD 4.3.
- ✦ ATT y Sun deciden cooperar para unificar System V y BSD.

1990

- ✦ ATT lanza la versión 4 de System V como un nuevo estándar para la unificación de las distintas variantes de UNIX (System V, BSD y XENIX). Esta es la consecuencia de la cooperación entre ATT y SUN.
- ✦ Otros fabricantes (como DEC, HP e IBM) amenazados por esta cooperación se unen creando la "Open Software Foundation" (OSF).
- ✦ Larry Wall y Randal Schwartz publican su libro "Programming in Perl", lenguaje que acabará convirtiéndose en el estándar usado para realizar utilidades de administración de sistemas UNIX, mientras que C se usará para desarrollos de sistemas.

1991

- ✦ Aparece en el mercado OSF-1. Hasta 1995 DEC es el principal fabricante que ha adoptado OSF, aunque algunos como IBM han adoptado partes.
- ✦ Empiezan a aparecer clones Unix de libre distribución como Linux o FreeBSD.

1992

- ✦ SUN desarrolla su sistema Solaris, que es un derivado de System V release 4 con soporte para Multiproceso simétrico.

- ✦ USL lanza UNIX System V 4.2 (Destiny).

1993

- ✦ Se empieza a distribuir libremente X Window (el GUI cliente/servidor desarrollado por el MIT y de amplio uso en las estaciones de trabajo hacia 1987) a plataformas Intel (XFree86).
- ✦ Se lanza BSD 4.4.
- ✦ Novell adquiere USL.

1994

- ✦ Empieza a popularizarse Internet.
- ✦ La arquitectura Cliente/Servidor está de moda.

1995

- ✦ Linux, un clon de UNIX desarrollado como proyecto de fin de carrera de Linus Torvalds e inspirado en Minix, está siendo desarrollado.
- ✦ El código de Unix (retornando a sus orígenes) está disponible de forma gratuita.
- ✦ X/Open introduce el estándar UNIX 95.
- ✦ Novell vende UnixWare a SCO.

1997

- ✦ El Open Group introduce la versión 2 de su especificación Single UNIX, que incluye soporte para tiempo real, threads, 64 bits y soporte procesadores de más capacidad.

1998

- ✦ El Open Group introduce la familia de estándares UNIX 98 que incluye la Base, Workstation y Server.
- ✦ Sun lanza los primeros productos registrados UNIX 98.

1999

- ✦ 30 aniversario de UNIX.
- ✦ Se lanza Linux kernel 2.2. Las compañías de software cada vez desarrollan más versiones de los productos más populares para Linux.

2000

- **Primeros** gestores de volúmenes lógicos para Linux.
- Sun lanza Solaris 8.

2.3. Clasificación

A continuación se exponen las familias UNIX más significativas:

- **AT&T**: la familia que tuvo su origen en el UNIX de AT&T. Considerada la familia UNIX "puraz original. Sus sistemas operativos más significativos son UNIX System III y UNIX System V.
- **BSD**: familia originada por el licenciamiento de UNIX a Berkely. BSD se reescribió para no incorporar propiedad intelectual originaria de AT&T en la versión 4. La primera implementación de los protocolos TCP/IP que dieron origen a Internet son la pila TCP/IP BSD.
- **AIX**: Esta familia surge por el licenciamiento de UNIX System III a IBM.
- **Xenix**: familia derivada de la adquisición de los derechos originales de AT&T primero por parte de Microsoft y de esta los vendió a SCO.
- **GNU**: En 1983, Richard Stallman anunció el Proyecto GNU, un ambicioso esfuerzo para crear un sistema similar a Unix, que pudiese ser distribuido libremente. El software desarrollado por este proyecto -por ejemplo, GNU Emacs y GCC - también han sido parte fundamental de otros sistemas UNIX.
- **Linux**: En 1991, cuando Linus Torvalds empezó a proponer el núcleo Linux y a reunir colaboradores, las herramientas GNU eran la elección perfecta. Al combinarse ambos elementos, conformaron la base del sistema operativo (basado en POSIX) que hoy se conoce como GNU/-Linux. Las distribuciones basadas en el núcleo, el software GNU y otros agregados entre las que se pueden mencionar a Red Hat Linux y Debian GNU/Linux se han hecho populares tanto entre los aficionados a la computación como en el mundo empresarial. Obsérvese que Linux tiene un origen independiente, por lo que se considera un 'clónico' de UNIX y no un UNIX en el sentido histórico.

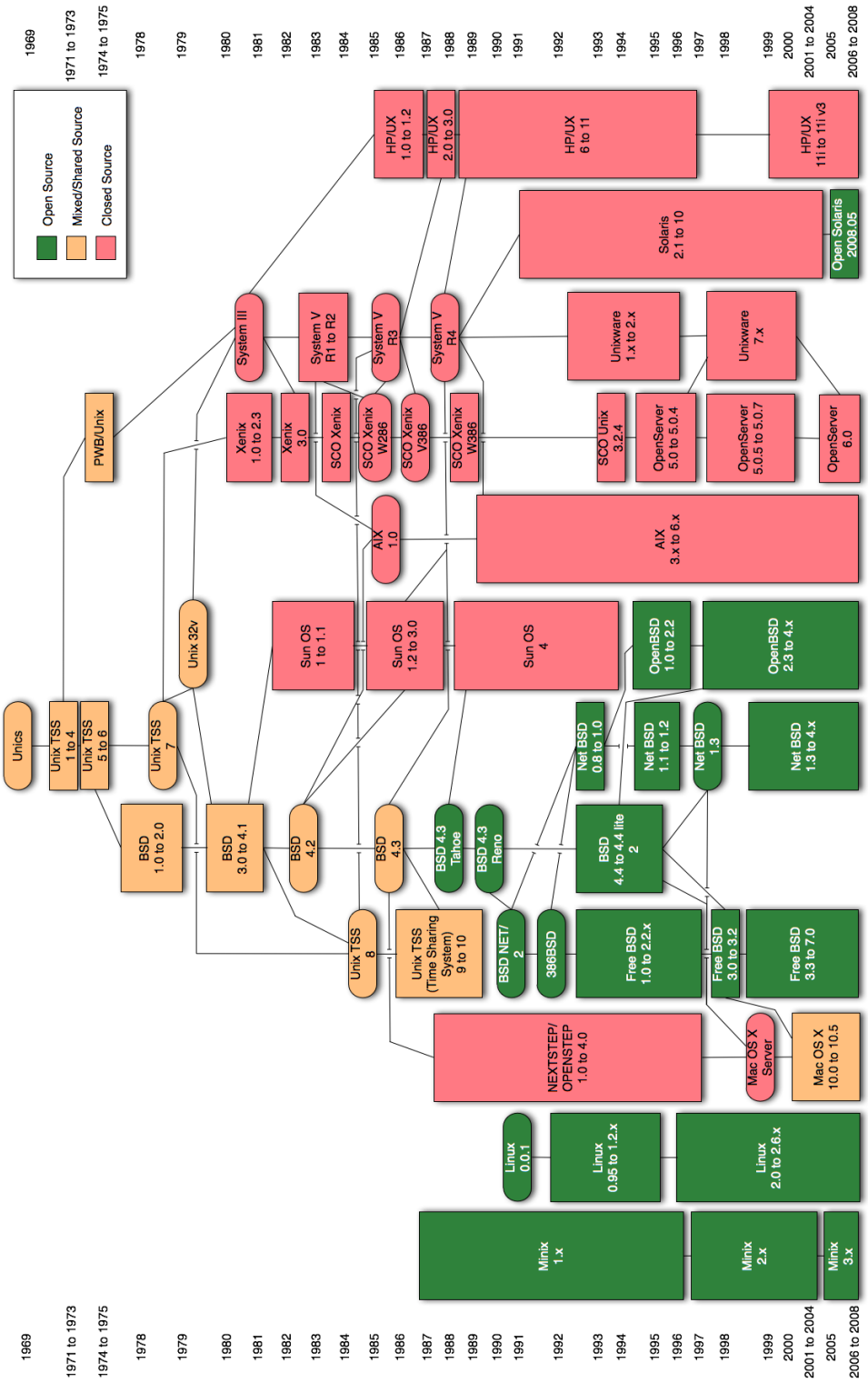

Figura 2.1: Árbol genealógico de la familia UNIX

2.4. Enlaces externos

- ✦ UNIX history en levenez.com

Capítulo 3

Estructura de los sistemas Unix

Descripción de los componentes básicos de la un sistema operativo de la familia Unix: kernel, shell e interfaz	
TIEMPO:	1 hora
NIVEL:	Principiante (página 3)
INFORMACIÓN EXTRAÍDA DE:	*The Linux Kernel* - David A Rusling (GFDL)
PRERREQUISITOS:	Ninguno

Este artículo consta de las siguientes secciones:

3.1. Introducción

La estructura de Unix se asemeja a un típico modelo de capas, donde cada capa puede comunicarse únicamente con las capas que se hallan en los niveles inmediatamente inferior y superior.

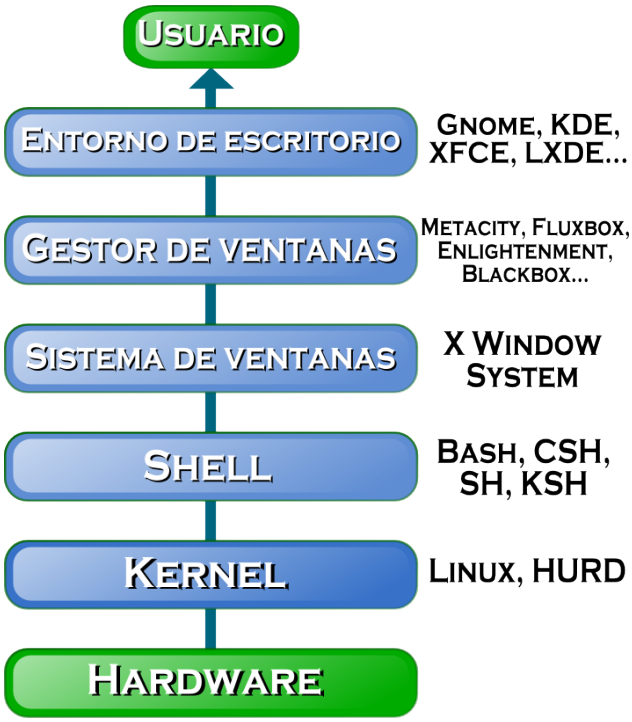

Figura 3.1: Estructura en capas del sistema operativo

3.2. Núcleo o Kernel

El **núcleo** o ***kernel*** es la parte del sistema operativo que sirve para interactuar con el hardware. Proporciona una serie de servicios que pueden ser utilizados por los programas, sin que éstos tengan que preocuparse de cómo se gestiona el hardware.

En general, el núcleo es el encargado de gestionar la memoria, mantener el sistema de archivos, del manejo de las interrupciones, manejo de errores, realización de los servicios de entrada/salida, asignación de los recursos de la UCP, gestión de periféricos de entrada/salida, entre otras funciones.

Cada programa se relaciona con la máquina a través del núcleo. Un programa realizará al núcleo las denominadas llamadas al sistema. Con estas el programa indicará, por ejemplo, que le abra un archivo, que escriba en otro, que utilice la impresora, que cambie la prioridad de ejecución de otro proceso, etcétera.

En Linux, existen dos versiones del kernel:

Versión de producción La versión de producción, es la versión estable hasta el momento. Esta versión es el resultado final de las versiones de desarrollo o experimentales. Cuando el equipo de desarrollo del núcleo experimental, decide que ha conseguido un núcleo estable y con la suficiente calidad, se lanza una nueva versión de produción o estable. Esta versión es la que se debería utilizar para un uso normal del sistema, ya que son las versiones consideradas más estables y libres de fallos en el momento de su lanzamiento.

Versión de desarrollo Esta versión es experimental y es la que utilizan los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, etc. Estos núcleos suelen ser inestables y no se deberían usar, a no ser que sepas lo que haces.

Las versiones del núcleo se numeran con 3 números, de la siguiente forma: *XX.YY.ZZ*

- XX: Indica la serie principal del núcleo. Hasta el momento solo existen la 1 y 2. Este número cambia cuando la manera de funcionamiento del núcleo ha sufrido un cambio muy importante.
- YY: Indica si la versión es de desarrollo o de producción. Un número impar, significa que es de desarrollo, uno par, que es de producción.
- ZZ: Indica nuevas versiones dentro de una versión, en las que lo único que se ha modificado son fallos de programación o bugs.

3.3. Shell

El **shell** es el intérprete de mandatos o de órdenes con el que cuenta este sistema operativo.

Actúa como interfaz de comunicación entre el usuario y el ordenador, y cuando un usuario se conecta con el servidor Unix, automáticamente se arranca un Shell para que pueda trabajar. Cada usuario conectado al servidor tendrá un Shell para su uso.

La independencia del shell respecto al *kernel* del operativo (el shell es sólo una capa de interfaz), nos permite disponer de varios de ellos en el sistema, entre ellos los siguientes:

- *Shell Bourne* (sh): Creado por Stephen Bourne, es el más utilizado en la actualidad. El prompt del sistema queda representado por el símbolo "$"("#.en caso de ser administrador).

  Este shell es el estándar de AT&T y el que se monta en casi todos los sistemas Unix/Linux.
- *C-Shell* (csh): Procedente del sistema BSD, proporciona características tales como control de trabajos, historial de órdenes, capacidades de edición, etc. Ofrece importantes características para los programadores que trabajan en lenguaje C. Su prompt de sistema queda representado con el símbolo " %". Fue desarrollado en la Universidad de Berkeley por Bill Joy a finales de los setenta y tiene unos cuantos añadidos interesantes al Bourne, como un histórico de comandos, alias, aritmética desde la línea de comandos, completa nombres de ficheros y hace control de trabajos en segundo plano. El prompt por defecto para los usuarios es ‘ %’. Los usuarios UNIX suelen preferir este shell como interactivo, pero los administradores UNIX prefieren utilizar el Bourne, ya que los scripts suelen quedar más compactos, y la ejecución suele ser más rápida. Por otro lado, una ventaja de los scripts en C shell es que, como su nombre indica, su sintaxis está basada en el lenguaje C (aunque no igual).
- *Shell job* (jsh): Incorpora algunas características de control al shell estándar del sistema.
- *Shell Korn* (ksh): Escrito por David Korn, amplia el shell del sistema añadiendo historial de órdenes, edición de la línea de ordenes y características ampliadas de programación.

- *Bourne Again shell* (Bash): Fue creado para usarlo en el proyecto GNU. BASH, por lo tanto, es un shell o intérprete de comandos GNU; éste es compatible con el shell sh. Además, incorpora algunas características útiles de ksh y csh, y otras propias, como la edición de línea de comandos, tamaño ilimitado del histórico de comandos, control de trabajos y procesos, funciones y alias, cálculos aritméticos con números enteros, etcétera. Es el shell GNU/Linux por defecto. El shell Bash (Bourne again shell) [Bas] [Coo] ha adquirido importancia desde su inclusión en los sistemas GNU/Linux como shell por defecto. Este shell forma parte del software del proyecto GNU. Es un intento de combinar los tres shell anteriores (Bourne, C y Korn), manteniendo la sintaxis del shell Bourne original. Es en el que nos vamos a fijar para desarrollar ejemplos posteriores.

3.3.1. ¿Qué shell estamos ejecutando?

Una forma rápida de conocer bajo qué shell nos encontramos como usuarios es mediante la variable "$SHELL", desde una línea de comandos con la instrucción:

```
echo \$SHELL
```

o también utilizando la órden:

```
ps
 PID     TTY       TIME     CMD
1244   pts/5   00:00:00    bash
4657   pts/5   00:00:00    ps
```

3.4. Interfaces de usuario

Éstas se definen como la parte del Sistema Unix que determina cómo interactúa el usuario con él, es decir, de qué forma el usuario introduce comandos o cualquier otra información y cómo el sistema visualiza los mensajes después de procesar tal información.

La interfaz primaria o básica del Unix es de tipo texto y hasta no hace mucho ha sido la única para el sistema. Actualmente hay interfaces gráficas como el X Windows, Open Look, GNOME o KDE. La interfaz de tipo texto es la que se muestra al cargar el shell por defecto o el deseado. Las básicas son las mismas que las explicadas en el punto 13.4.A de esta unidad, aunque hay más.

En cuanto a la interfaz gráfica X-Windows, la característica fundamental es que incorpora un modelo cliente-servidor para el modo en que las aplicaciones interactúan con los dispositivos terminales. Incorpora también un protocolo de red y varias herramientas software que pueden ser utilizadas para crear aplicaciones basadas en X Windows.

Un concepto fundamental es la separación de las aplicaciones con respecto al software que maneja la entrada y salida por Terminal. Todas las operaciones realizadas mediante la entrada o salida estándar (teclado y monitor) son manejadas por un programa que se dedica exclusivamente a ello (servidor). Las aplicaciones (clientes) envían al servidor información a visualizar, y el servidor en-

vía a las aplicaciones información referente a la entrada de usuario. Para gestionar este modelo, se utiliza el protocolo de red X. Este protocolo es un lenguaje estándar utilizado por las aplicaciones clientes para enviar instrucciones a los servidores X, y por los servidores para enviar la información transformada a los clientes (por ejemplo, el movimiento del ratón).

La interfaz OPEN LOOK ha sido diseñada por AT&T y Sun Microsystems como interfaz gráfica estándar para el sistema Unix/Linux. Esta interfaz permite ejecutar y visualizar varias aplicaciones simultáneamente en ventanas separadas sobre una misma pantalla. En general, todas las operaciones de gestión de archivos se realizan de forma gráfica gracias a la interfaz OPEN LOOK.

La interfaz KDE 3.2 para Linux SUSE, ahora con un potente gestor de información personal (PIM), incorpora, además de la función de inicio rápido y la reproducción automática de los medios introducidos (CD, DVD), KDE 3.2, un gran número de nuevas prestaciones y programas.

La interfaz GNOME 2.4 ha mejorado la usabilidad y se han incluido ayudas de accesibilidad para discapacitados, como, por ejemplo, un lector de pantalla con salida de voz o a una línea braille.

3.5. Ejercicios

I. Cambia la shell de login a tcsh sin convertirte en root.

II. Averigua la versión del kernel de tu equipo.

3.6. Enlaces externos

✦ Charting the Linux Anatomy en O'Reilly

Capítulo 4

Conceptos básicos

Definición del vocabulario básico y las palabras que se suelen emplear en el ámbito de los sistemas Unix.	
TIEMPO:	2 horas y 30 minutos
NIVEL:	Principiante (página 3)
INFORMACIÓN EXTRAÍDA DE:	*Manual Práctico de Linux. Comandos, editores y programación Shell.* - Mark G. Sobel ✒ *FAQ sobre Linux - es.comp.os.linux.** - 1999-2000 Rafael Martínez (GFDL)
PRERREQUISITOS:	Unix

Este artículo consta de las siguientes secciones:

4.1. Introducción

Una vez sepas identificar qué es un sistema Unix y qué no lo es, hay que iniciarse en la jerga habitual dentro de dichos sistemas. Se utiliza un vocabulario especial y unos conceptos sobre los cuales se sostienen las ideas principales que debemos tener claras. Para ello, vamos a aclarar algunos de estos conceptos especiales.

4.2. Vocabulario básico

- **Archivos** o **ficheros**: colecciones de información (datos relacionados entre sí), localizada o almacenada como una unidad en alguna parte de un ordenador.
  - **Archivo oculto**: Archivo que no se encuentra visible al usuario por defecto. Sueles contener información de algún tipo de configuración del equipo. En Linux, podemos distinguir estos archivos porque comienzan por un punto (.).
- **Directorio:** agrupación de archivos de datos. En UNIX, el directorio se organiza a partir del directorio **raíz** "/", el cual contiene archivos y otros directorios. Esos directorios pueden contener archivos y directorios y así sucesivamente. Esto se organiza por el sistema en una estructura llamada *árbol*.
  - **Directorio *casa*** o *de inicio*: directorio que contiene los archivos personales de algún usuario en particular. En UNIX, este directorio alberga archivos de configuración, (generalmente ocultos, por ejemplo comenzando con .), documentos, programas instalados localmente, etc. El directorio de inicio se define como parte de los datos de las cuentas de usuario.
  - **Directorio de trabajo**: directorio en el que el proceso de nuestra *shell* se encuentra en ejecución, es decir, no es un directorio real (que existe ya definido) sino una convención para hacernos creer que nos encontramos en un directorio y así simplificar nuestro trabajo.
    Informalmente podríamos decir que el directorio de trabajo es el directorio en que nos encontramos en un instante de tiempo determinado, aunque esto sea solo una simulación.
- **Ruta** o *path*: jerarquía de directorios en la que se halla un elemento del sistema de archivos. Es decir, sería como el camino que recorremos para llegar a un archivo determinado.
  - **Ruta absoluta**: señalan la ubicación de un archivo o directorio desde el directorio raíz del sistema de archivos. Se pueden distinguir de las rutas relativas fácilmente porque comienzan por "/", por ejemplo: */home/usuario/.bash_history*
  - **Ruta relativa**: señalan la ubicación de un archivo o directorio a partir de la posición actual del sistema operativo en el sistema de archivos. Por ejemplo, *usuario/.bash_history* se refiere al archivo *.bash_history* dentro del directorio *usuario* en la ubicación actual.
    En sistemas UNIX, la ruta ~ es una ruta relativa que lleva al directorio personal del usuario que ha insertado la ruta relativa,*/home/usuario*.
- **Demonio** o *daemon* (*Disk And Execution MONitor*): proceso que se ejecuta en segundo plano en lugar de ser controlado directamente por el usuario, es decir, es un proceso no interactivo. Este tipo de programas se ejecutan de forma continua (infinita): aunque se intente cerrar o matar el proceso, este continuará en ejecución o se reiniciará automáticamente, sin intervención de terceros y sin dependencia de consola alguna.
  Un ejemplo son los demonios *cronológicos* como **cron**, que realizan tareas programadas como mantenimiento del sistema en segundo plano.
- ***Login*** y ***password***: Nombre de usuario y contraseña para acceder al sistema o para identificarse cuando sea necesario.

- **Orden**, **comando** o **mandato**: instrucción que el usuario proporciona a un sistema informático, desde la línea de comandos o desde una llamada de programación.
- **Consola**, también llamada **terminal** o **línea de comandos**: es un intérprete que espera órdenes escritas por el usuario en el teclado, las interpreta y las entrega al sistema operativo para su ejecución. La respuesta del sistema operativo se muestra al usuario en la misma ventana. A continuación, el programa shell queda esperando más instrucciones.
- **Ventana de línea de comandos** o **prompt**: es lo que se muestra cuando se inicia el shell a través de la lectura de su configuración completa (en un archivo del directorio "/etc"), tras la lectura de la configuración propia del usuario (en */home/usuario/.configuration_file*).
Por defecto, en la mayoría de los shells el "prompt" (aviso) consiste en el nombre de la máquina seguido por dos puntos (:), el directorio actual y luego, un carácter que indica el tipo de usuario conectado:
  - "$" define a un usuario normal
    `usuario@usuarioPC: $`
  - "#" define al administrador, llamado "root"
    `root@usuarioPC:/home/usuario#`

4.3. Variables de entorno

Las variables de entorno son un conjunto de valores dinámicos que normalmente afectan el comportamiento de los procesos en un ordenador.

Algunas de estas variables más útiles (podemos verlas por ejemplo, mediante el comando **echo**), que pueden consultarse ya sea en la línea de comandos o dentro en la programación de shells script son:

Variable	***Valor Ejemplo***	***Descripción***
HOME	/home/usuario	Directorio raíz del usuario
LOGNAME	Usuario	Identificador del usuario en el sistema
PATH	/bin:/usr/local/bin	Caminos o rutas
SHELL	/bin/bash	Shell del usuario
PS1	$	Prompt del shell
MAIL	/var/mail/usuario	Directorio del buzón de correo
TERM	xterm	Tipo de terminal que el usuario utiliza
PWD	/home/usuario	Directorio actual en que se encuentra el usuario

Cuadro 4.1: Lo más básico

Las órdenes **env**, **set** y **printenv** muestran todas las variables de entorno junto con sus respectivos valores. **env** y **set** se usan también para asignarles valores.

```
$ env
SSH_AGENT_PID = 598
MM_CHARSET = ISO-8859-15
TERM = xterm
```

```
DESKTOP_STARTUP_ID =
SHELL = /bin/bash
WINDOWID = 20975847
LC_ALL = es_ES@euro
USER = juan
LS_COLORS = no = 00:fi = 00:di = 01;34:ln = 01;
SSH_AUTH_SOCK = /tmp/ssh-wJzVY570/agent.570
SESSION_MANAGER = local/aopcjj:/tmp/.ICE-unix/570
USERNAME = juan
PATH=/soft/jdk/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/
X11:/usr/games
MAIL = /var/mail/juan
PWD = /etc/skel
JAVA_HOME = /soft/jdk
LANG = es_ES@euro
GDMSESSION = Gnome
JDK_HOME = /soft/jdk
SHLVL = 1
HOME = /home/juan
GNOME_DESKTOP_SESSION_ID = Default
LOGNAME = juan
DISPLAY = :0.0
COLORTERM = gnome-terminal
XAUTHORITY = /home/juan/.Xauthority
_ = /usr/bin/env
OLDPWD = /etc
```

printenv permite también mostrar el valor de una variable de entorno particular si se le pasa su nombre como único parámetro.

```
printenv SHELL
/bin/bash
```

4.4. Cómo obtener la ayuda del sistema

La búsqueda de **ayuda** es una de las destrezas más importantes para el buen administrador de sistema. Una persona nunca podrá conocer todas las opciones de todas los programas. Es más, también puede olvidar algunos o desconocer otros nuevos que surgen cada día. Por ello es de especial importancia conocer los distintos modos de búsqueda de ayuda en un sistema UNIX.

4.4.1. El manual

El manual de usuario del sistema es la principal fuente de documentación que se ha usado de manera çlásica.[en] los sistemas Unix. Actualmente la mayoría de programas que funcionan en modo gráfico

no incluyen entrada en el manual, pero sí la mayoría de programas de consola y muchas bibliotecas (incluyendo las llamadas al sistema).

El manual está ordenado por secciones, conteniendo cada sección muchas entradas (que también se suelen denominar "páginas"). Lo normal es que cada programa tenga una entrada con su sintaxis, sus opciones, ejemplos de uso, entradas relacionadas, etc.

```
man <entrada>
```

El programa **man** permite, entre otras cosas, ver la página solicitada, moverse sobre la misma y realizar búsquedas. Para realizar la búsqueda, presione **/** y después teclee la palabra que busca (realmente **man** usa los mismos controles que less)

A veces, puede haber más de una entrada en el manual. Por ejemplo, puede que coincidan el nombre de una llamada al sistema con el de un comando o un fichero del sistem. En este caso, hay que especificar qué sección quiere consultarse:

```
man n <entrada>
```

siendo *n* el número de sección.

Por ejemplo, para ver la entrada de la orden passwd escribiría: `man 1 passwd` Pero para ver la descripción del fichero passwd (que almacena información de los usuarios del sistema) escribiría: `man 5 passwd`

Existen también diversas herramientas gráficas de exploración de los manuales, por ejemplo **xman** y **tkman**, así como versiones on-line del manual (por ejemplo esta de Ubuntu).

En caso de que se desee buscar alguna entrada en el manual se puede usar al orden apropos o: `man -k <término>`

4.4.2. El sistema info

El sistema de ayuda info nació para superar la principal limitación del "man": su estatismo. El sistema info nos permite navegar por el mediante enlaces similares a los de una web.

Dependiendo del autor de un programa podremos encontrarnos con página de manual pero no de info, viceversa, o el mismo contenido en ambos sistemas, etc. Por lo tanto siempre es recomendable, si no hemos obtenido información suficiente con uno de los sistemas, consultar el otro.

4.4.3. Ayuda incorporada

Además de la ayuda "tradicional"(man e info), los programas suelen incorporar ayuda de varias formas:

- Si son programas gráficos suelen tener una entrada de menú o pestaña con .Ayuda".
- Si son programas interactivos en modo texto suelen tener alguna forma de ayuda. Por ejemplo

- ➥ En el comando top basta con pulsar la letra "h". Para salir de la ayuda y después del sistema se usa la letra "q"(como suele ser habitual en la mayoría de programas interactivos de texto).
- ➥ En emacs debemos pulsar la combinación de teclas Control+h dos veces (lo que se suele escribir como C-h C-h) para obtener ayuda o C-h y la letra "t"para acceder a un tutorial (también disponible en el menú .Ayuda» "Tutorial").

- ✦ Si son programas no interactivos suelen tener alguna opción para la ayuda, normalmente –help (como chmod o ls) o a veces con -h.

4.4.4. Ayuda adicional de los paquetes

Además de la ayuda del sistema y la ayuda incorporada algunas aplicaciones incluyen documentación extra al instalarlas: manuales, tutoriales, guías de usuario, plantillas, hojas de referencia e incluso libros enteros, lista de errores detectados en el programa, descripciones de las funcionalidades añadidas al programa etc.

Normalmente, estos componentes de documentación se instalan en el directorio */usr/share/doc* (o */usr/doc* dependiendo de la distribución), donde normalmente se crea un directorio por paquete de aplicación (normalmente la aplicación puede disponer de paquete de documentación por separado). Por ejemplo:

- ✦ El excelente sistema de documentación automática doxygen incluye un manual de casi 150 páginas en */usr/share/doc/doxygen/doxygen_manual.pdf*
- ✦ Si queremos una hoja de referencia de Emacs sólo tenemos que irnos a */usr/share/emacs/22.1/etc/refcard.ps* (siendo 22.1 la versión de *Emacs* que tenemos instalada).
- ✦ Para saber los errores detectados en la versión de sudo instalada se puede consultar */usr/share/doc/sudo/BUGS*
- ✦ La lista de funciona incluidas en las últimas versiones de Inkscape está en */usr/share/doc/inkscape/changelog.gz*

4.4.5. Ayuda de escritorio o distribución

Los escritorios X, como Gnome y KDE, normalmente también traen sistemas de documentación propios con su documentación y manuales, así como información para desarrolladores, ya sea en forma de ayudas gráficas en sus aplicaciones, o en aplicaciones propias que recopilan las ayudas (por ejemplo, *devhelp* en Gnome).

Igualmente sucede con ciertas distribuciones como OpenSUSE o similares. Por ejemplo, Red Hat suele venir con unos CD de manuales en PDF o HTML que son instalables en el sistema.

4.4.6. Otras formas de ayuda

Además de todas las formas de ayuda comentadas anteriormente (todas ellas incluidas en el sistema) existen otras dos formas más: manuales impresos (que se pueden comprar, como por ejemplo los libros de la tienda de GNU) y ayuda a través de la web.

Si buscamos ayuda a través de la web recomendamos consultar primero webs oficiales del sistema, o web de grupos de usuarios que cuenten con el respaldo de la comunidad, evitando búsquedas .ª semi-ciegasçon un buscador cualquiera.

4.5. Malas prácticas y consejos útiles

Para cualquiera que no haya utilizado nunca un sistema Unix o esté acostumbrado a los malos vicios de sistemas privativos, sería conveniente echar un vistazo a la siguiente lista con cosas que **NO** se deben hacer nunca, así como una serie de consejos que pueden ser de utilidad:

I. **No** busques software gratuito ni mucho menos saltándote las licencias privadas. Existen muchísimas alternativas libres a la gran mayoría de aplicaciones privativas actuales, las cuales se pueden encontrar fácilmente en los repositorios de paquetes de tu sistema o en forjas de software como *sourceforge*.
Probablemente encuentres varias aplicaciones para hacer lo mismo: en la variedad está el gusto, busca la que más cómoda te sea.

II. **Nunca** debemos ejecutar X como usuario *root*: éste debe ser usado sólo para tareas administrativas.
Si en este caso existiera una vulnerabilidad en la aplicación, un atacante podría hacer lo que quisiera con nuestro sistema.

III. **Siempre** crear un usuario para el trabajo cotidiano. Si éste necesita habitualmente acceso a *root*, puede agregarse al grupo *wheel*. Esto hace posible que un usuario normal recurra a la orden *su* para poseer temporalmente los privilegios de administrador.

IV. **Nunca** dejes una terminal abierta estando conectado como *root*. Si no vas a tocar esa terminal en un rato, ciérrala o vuelve a tu usuario con la orden: `su <usuario>`

V. **Procura** crear *alias* que sean útiles en tu trabajo diario.
Puedes definir *alias* que invoquen una orden determinada para poder simplificar tu trabajo. Si quieres definirlos para un usuario determinado, es conveniente definirlos en el fichero *~/.bash_aliases*, teniendo en cuenta que si no existe debes crearlo y que debes asegurarte de que en el fichero *~/.bashrc* se encuentran las siguientes líneas tal y como aparecen a continuación:

ARCHIVO: .bashrc

```
# Alias definitions.
# You may want to put all your additions into a separate file like
#  /.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f  /.bash_aliases ]; then
```

```
:. /.bash_aliases
fi
```

Se podrían incluir estos alias directamente en el fichero *~/.bashrc*, pero es más adecuado utilizar *~/.bash_aliases* por distinguir más fácilmente los *alias* del sistema de los creados por el usuario.

Un ejemplo bastante útil y simple es el forzado de borrado interactivo:

ARCHIVO: ~/.bash_aliases

```
alias rm='rm -i'
```

Así, cada vez que vayamos a borrar uno o varios ficheros nos preguntará si estamos seguros de querer eliminarlo. Recuerda que en un terminal no hay papelera de reciclaje: lo que se borra desaparece para siempre.

VI. Es **conveniente** al instalar nuestro sistema definir las particiones manualmente, para crear una partición extra que se monte en */home*, además de la partición de intercambio y la de la raíz del sistema de ficheros. Así si ocurriese algún problema por el que debamos reinstalar el sistema, no tendremos que realizar *backup* alguno (al menos no de forma necesaria, nunca está de más una copia de seguridad).

VII. **Intenta** no desaprovechar lo que la consola te aporta. Debes ser eficiente:

- Utiliza la función de *autocompletar* (generalmente con la tecla Tabulador).
- Utiliza el histórico de comandos (pulsando las flechas de dirección hacia arriba y abajo).
- Utiliza *Shift+RePag*/*AvPag* para moverte hacia arriba/abajo en la pantalla de la terminal.
- Utiliza trucos de la librería *readline* para modificar la línea actual:
  - Inicio de Línea: CTRL-a
  - Fin de línea: CTRL-e
  - Carácter Atrás: CTRL-b
  - Carácter Adelante: CTRL-f
  - Palabra Adelante: ALT-f
  - Palabra Atrás: ALT-b
  - Borrar carácter: CTRL-d
  - Borrar Palabra: ALT-d

4.6. Ejercicios

I. ¿Cuántas páginas man hay en la subsección de dispositivos del manual del sistema?

II. ¿Cómo diferencio entre ruta absoluta y ruta relativa?

III. Busca qué utilidades crean y trabajan con archivos.

IV. Clasifica las rutas descritas a continuación en nombre de ruta absoluta o de ruta relativa:

- ✦ vers_nuevo
- ✦ /var/www/index.html
- ✦ ..
- ✦ /home/usuario/Doc
- ✦ ../A/B/c
- ✦ .plan.planner

Si deseas comprobar el ejercicio, introduce las rutas absolutas en el fichero absolutas.txt y las rutas relativas en relativas.txt. Luego ejecuta el script de comprobación (rutas.py) en el mismo directorio en que se encuentren dichos archivos.

Parte II

Nivel usuario iniciado

Capítulo 5

Órdenes básicas

Explicación a fondo de las órdenes más básicas que se deben conocer en un sistema Unix por su utilización común.	
TIEMPO:	2 horas y 30 minutos
NIVEL:	Usuario iniciado (página 33)
INFORMACIÓN EXTRAÍDA DE:	Unix Guru Universe ✒ Manual práctico de Linux - Mark G. Sobell (2008, Anaya, ISBN: 978-84-415-2350-0)
PRERREQUISITOS:	Vocabulario básico ✒ Soporte para pruebas

Este artículo consta de las siguientes secciones:

5.1. Introducción

Las órdenes en Unix siguen el patrón básico siguiente:

```
nombre_orden opciones argumento(s)
```

- Las opciones modifican las manera en que funciona la orden. Normalmente consisten en un guión seguido de una única letra. Por ejemplo:

  ```
  ls -l
  ```

  También pueden utilizarse varias opciones simultáneamente de forma que se indiquen cada una con un guión:

  ```
  ls -l -a
  ```

  o agrupándolas todas bajo un mismo guión:

  ```
  ls -la
  ```

- Muchas órdenes se utilizan con uno o más argumentos. Si el argumento contiene uno o más espacios en blanco es necesario encerrarlo entre comillas . Como en este caso:

  ```
  ls "directorio uno"
  ```

  o bien ”protegiendo” el espacio con una barra inversa:

  ```
  ls directorio uno
  ```

  Las órdenes se vuelven mucho más efectivas cuando se utilizan como argumento expresiones regulares.

5.2. Sinopsis de una orden

De forma general se sigue un esquema para especificar de forma simple las opciones y los argumentos necesarios para la ejecución de ésta. Esto se denomina "sinopsis".

En esta sintaxis se siguen las siguientes reglas:

- Opciones o argumentos encerrados entre “[” y “]” (corchetes) se consideran opcionales.
- Todo lo que no esté encerrado entre corchetes debe ser introducido.
- Las palabras de negrita se consideran como literales y deben ser escritas exactamente como aparecen. Esto por lo general se aplica al nombre de la orden y a sus opciones.
- Los argumentos que se muestran en cursiva deben ser remplazados por aquello que representen. Puede ser el nombre de un fichero o directorio.
- Los puntos suspensivos “...” implican que el argumento previo puede ser repetido tantas veces como se desee.

5.3. Trabajando en la terminal

5.3.1. Cambiando de directorio con *cd*

Cuando se trabaja desde la terminal[1] no importa dónde te muevas: siempre estás en un directorio. En cada momento, el ”prompt” indica en qué directorio nos encontramos y el comando ’pwd’ informa

[1] Using The Terminal - Manual Page Ubuntu

de nuestra posición actual en la jerarquía del sistema de archivos.

Ojo: cuando navegas por el sistema de archivos en la terminal siempre estarás en algún lugar.

El comando ”cd” (change directory) te permite moverte de un directorio a otro. En jerga Unix, se denomina ”cambiar de directorio de trabajo”, es decir, cambiar el directorio en el cual te encuentras y en el cual puedes actuar o utilizar archivos y directorios sin especificar un camino.

Vamos a probar cómo funciona:

- Abriendo una nueva terminal, sin desplazarnos a ningún otro directorio, mostramos el contenido del directorio casa ejecutando la órden ’ls’.

  ```
  usuario@usuarioPC:~$ ls
  Documentos Escritorio examples.desktop Imágenes Música
  Plantillas Público Vídeos
  ```

- Para movernos al directorio .Escritorio", introduce lo siguiente:

  ```
  usuario@usuarioPC:~$ cd Escritorio
  usuario@usuarioPC:~\Escritorio$
  ```

  El "promptçambia para reflejar el directorio de trabajo actual.

  Si necesitas moverte dentro de una carpeta cuyo nombre contiene más de una palabra, encierra su nombre entre comillas o bien inserta "\"(barra inversa) antes de cada espacio. Por ejemplo, si queremos ir a un supuesto directorio "Mis documentos"tendremos que introducir:

  ```
  usuario@usuarioPC:~$cd "Mis documentos"
  ```

  o bien:

  ```
  usuario@usuarioPC:~$cd Mis\documentos
  ```

  Volviendo a donde estábamos, para ver qué contiene, introduce ’ls’:

  ```
  usuario@usuarioPC:~\Escritorio$ % ls
  user.Desktop
  ```

- Prueba a moverte dentro de los directorios con ’cd’ y ver su contenido, utilizando ’ls’. En este momento, probablemente ya no sabes dónde te encuentras en tu sistema de archivos. Utiliza el comando ”pwd” para saber tu posición:

  ```
  usuario@usuarioPC:~/Documentos/DocA$ pwd
  /home/usuario/Documentos/DocA
  ```

- Ahora puedes utilizar el comando *cd* para desplazarte de vuelta en la jerarquía ejecutando *cd* sin especificar el camino completo de la carpeta a la que quieres volver. Por ejemplo, si quieres volver al directorio */home*, introduce:

  ```
  user@Equipo:~$ cd /home
  user@Equipo:/home$
  ```

  El ”prompt” muestra que estás en el directorio *home*. Si quieres volver al directorio "DocA.nterior, puedes teclear lo siguiente:

  ```
  usuario@usuarioPC:~/home$ cd /usuario/Documentos/DocA
  usuario@usuarioPC:~/Documentos/DocA$
  ```

- Si intentas moverte dentro de un directorio que no existe, o si tecleas erróneamente el nombre del directorio, verás el siguiente mensaje:

```
usuario@usuarioPC:~/Documentos/DocA$ cd Intento
bash: cd: Intento: No existe el fichero ó directorio.
```

Atajo a casa

Puede parecer que utilizar la órden 'cd' requiere teclear mucho y en su uso básico, puede ser verdad. De cualquier forma, 'cd' también ofrece muchos atajos que permiten navegar rápida y fácilmente por el sistema de archivos. El atajo más útil es teclear solamente 'cd':

```
usuario@usuarioPC:~/Documentos/DocA:~$cd
usuario@usuarioPC:~$
```

Así, *cd* nos devuelve a nuestro directorio casa (*/home/usuario* en nuestro caso).

Otro dato interesante es la utilización del carácter ~ (virgulilla) como atajo directo al directorio casa del usuario. El sentido de este atajo es que cualquier sistema Unix considera que los usuarios por defecto trabajan únicamente en su directorio casa o en subdirectorios de éste. Así, este atajo te permite moverte fácilmente por estos directorios. Si se quiere ir al directorio "Documentos"que está dentro de tu directorio casa, se puede utilizar:

```
usuario@usuarioPC:~$ cd ~/Documentos
usuario@usuarioPC:~/Documentos$
```

Utilizando este atajo se evita tener que introducir la ruta completa del directorio al que se desea ir (en este caso, */home/usuario/Documentos*).

Subiendo/bajando

Cada directorio en tu sistema de archivos contiene como mínimo dos entradas de forma automática. Se trata de . (punto) y .. (punto punto).

Puedes comprobar esto consultando cualquier directorio ya creado o creando un nuevo:

```
usuario@usuarioPC:~/Documentos$ mkdir Prueba
usuario@usuarioPC:~/Documentos$ cd Prueba
usuario@usuarioPC:~/Documentos/Prueba$ ls –a
. ..
```

Especificando el parámetro *-a* a la órden *ls*, se muestra el contenido de este directorio, incluidos los archivos ocultos. Así puedes confirmar que los únicos archivos en este nuevo directorio son . (punto) y .. (punto punto).

Estos archivos son muy útiles. El primero, . (punto), representa el actual directorio de trabajo, y puede ahorrarte tiempo en ciertos comandos relacionados con archivos dentro de este directorio. El segundo, .. (punto punto), es un atajo al directorio origen o padre: el directorio inmediatamente superior en la jerarquía del sistema de archivos.

Gracias a la existencia de estos archivos, en cualquier directorio se puede hacer lo siguiente:

```
usuario@usuarioPC:~/Documentos/Prueba$ cd ..
usuario@usuarioPC:~/Documentos$
```

Como puedes ver en el *prompt*, al teclear *cd* .. regresamos al directorio padre de "Prueba", "Documentos". Siempre que te muevas a un subdirectorio utilizando el comando *cd*, puedes usar *cd* .. para volver atrás. Pero también te puedes mover hacia arriba en el sistema de archivos desde el actual directorio hasta alcanzar su límite superior (el directorio raíz, /"):

```
usuario@usuarioPC:~/Documentos$ cd ..
usuario@usuarioPC:~$ cd ..
usuario@usuarioPC:/home$ cd ..
usuario@usuarioPC:/$
```

Ahora ya podemos desplazarnos hacia arriba y abajo en la jerarquía del sistema de archivos.

Ojo: No existe un atajo para volver al punto desde el que empiezas a moverte (a no ser que éste sea el directorio casa), por lo que tendrás que desplazarte paso a paso, o bien, indicando la ruta completa del directorio para regresar al lugar inicial.

5.4. Ejecución de procesos

Para ejecutar un proceso desde la terminal tan sólo debemos realizar la llamada:

```
<nombre_proceso>
```

Por ejemplo, si quisiéramos ejecutar el editor emacs:<div style="background-color:

```
emacs
```

Así estaríamos ejecutando dicho proceso en primer plano (*foreground*). Si queremos continuar trabajando con la terminal desde la que hemos llamado al proceso tenemos que ejecutarlo en segundo plano (*background*):

```
<nombre_proceso>&
```

En nuestro ejemplo:

```
emacs &
```

Para devolver un proceso que se encuentre en segundo plano al primero utilizamos la orden **fg**:

```
fg <nombre_proceso>
```

```
fg emacs
```

5.5. Ahorrando tiempo

5.5.1. Ejecutar varias órdenes

- Ejecutar varias órdenes en secuencia: separándolas con ";"(punto y coma)

  ```
  cd /home ; ls ; cd
  ```

- Ejecutar varias órdenes como si fueran una sola: separándolas con ";"(punto y coma) y rodeándolas entre "{ז "}"(llaves). Más adelante veremos la diferencia con la forma anterior.

  ```
  {cd /home ; ls ; cd ;}
  ```

- Órdenes en más de una línea: colocar "\"(barra inversa) al final de la línea actual y pulsar *Intro*, apareciendo una nueva línea que comienza por »"donde deberemos terminar de escribir la orden

  ```
  cd \
  >/home
  ```

5.5.2. Flujo de datos en la terminal

Cada proceso en entornos Unix suele disponer de tres archivos abiertos al comienzo de su ejecución: la entrada estándar (*stdin*), la salida estándar (*stdout*) y la salida de errores estándar (*stderr*). Se denominan *estándar* porque suelen estar asignados a descriptores de ficheros conocidos, de manera que un programa siempre tomará los datos de entrada por el descriptor cero, enviará los resultados por el uno y mostrará los errores por el dos.

Nota: Esto no se utiliza en sistemas gráficos como X Window o en demonios.

El shell del sistema es el encargado, generalmente, de abrir esta entrada y conectarla con el terminal al que está asignado el programa, puesto que es el programa más usual para comunicarse con un operador.

Ojo. Aunque por defecto estén asociados así, la entrada estándar *NO* es el teclado y la salida estándar *NO* es la pantalla.

- Redireccionar las salidas:
  - *stdin*: con “<”

    ```
    orden <fichero
    ```

  - *stdout*: con “>” (reescribe el fichero, en caso de que contenga algo se pierde) y “>>” (añade contenido al final del fichero)

    ```
    orden >fichero
    orden >> fichero
    ```

  - *sterr*: con “2>”

    ```
    orden 2>fichero
    ```

- Redireccionar *stdout* y *stderr* a un mismo archivo (las siguientes formas son equivalentes, pero se suele usar la segunda por rapidez):

```
orden >archivo 2>&1
orden >& archivo
```

- Redireccionar salida de varios procesos: agrupar comandos con llaves, separados por ;

5.5.3. Tuberías

Las tuberías (*pipes*) son mecanismos de comunicación específicos para todos los sistemas Unix. Una tubería, simbolizada por "|"(barra vertical), permite asignar la salida estándar de una orden a la entrada estándar de otra, de la misma forma en que una tubería permite la comunicación entre la entrada estándar de una orden y la salida estándar de otra.

En el siguiente ejemplo, la salida estándar del listado se envía a *sort*, el cual ordena los resultados en orden alfabético.

```
ls –al | sort
```

Esto permite conectar una cierta cantidad de órdenes a través de sucesivas tuberías. En el siguiente ejemplo, se listan todos los archivos del directorio actual, selecciona las líneas que contienen la palabra "zip*z* cuenta la cantidad total de líneas:

```
ls –l | grep zip | wc –l
```

5.5.4. Trabajar con varios archivos

Existen formas para *expandir archivos* dentro de una misma línea de órdenes.

I. Se pueden especificar uno a uno como una lista de caracteres entre "{*z* "}"(llaves) y separados por comas (**sin espacios**).

II. Con expresiones regulares se pueden denotar varios ficheros cuyo nombre cumpla unas características especiales. Para ello se utilizan *metacaracteres* (operadores), entre los cuales se encuentra:

   - “∗” denota varias apariciones (o ninguna) de la expresión que lo precede
   - “?” denota una aparición de la expresión que lo precede
   - “+” denota una o más apariciones de la expresión que lo precede
   - “[” y “]” sirven para especificar rangos de valores o conjunto

5.5.5. Esquema general

Redireccionamiento de E/S de Shell	
Redirección	***Significado***
>fichero	dirige salida estándar o archivo
>> fichero	anexa salida estándar o archivo
<fichero	toma salida estándar o archivo
2>fichero	dirige salida de errores
>& fichero	dirige salida estándar y de errores
p1 | p2	conecta salida estándar de *p1* con la entrada de *p2*

5.6. Resumen

Las órdenes más comunes que han sido explicadas en este apartado son:

Orden	**Descripción**
pwd	Averigua el directorio actual
cd	Cambia de directorio
ls	Lista el contenido de un directorio
mkdir	Crea un nuevo directorio
cat	Muestra el contenido de un archivo
cp	Copia un archivo
mv	Mueve un archivo
rm	Elimina un archivo

5.7. Ejercicios

ASPECTOS PEDAGÓGICOS: Permiten el aprendizaje de órdenes elementales y favorecen la asimilación de trabajo en la terminal. Se trabajará con órdenes muy simples para el procesamiento básico de archivos y directorios, tales como listados, copias, movimiento, creación y borrado entre otras.
PREPARACIÓN PREVIA: Para hacer estos ejercicios debes acceder al sistema con el usuario por defecto *usuario* (recuerda que la contraseña es *unix*). A continuación:

✦ Ve al directorio casa del usuario actual. Comprueba que existe un directorio llamado **unix** y dentro de éste un subdirectorio llamado **ordenes_basicas**. Para ello ejecuta:
`ls unix`

Debe aparecer el directorio **ordenes_basicas**. En caso de que no exista alguno de estos elementos, puedes hacer varias cosas:

1. Descargar el archivo comprimido con la jerarquía de elementos de comprobación que se encuentra en la portada del wiki y descomprimirlo en el directorio casa del usuario actual.

2. Apagar la máquina virtual y restaurar el *snapshot* que funcionaba bien.
3. Apagar la máquina virtual y borrarla eliminando el disco duro virtual. A continuación, descargar el disco duro y volver a configurar la máquina de nuevo.

✦ Ve al subdirectorio **unix/ordenes_basicas**. Limpia el sistema antes de realizar los demás ejercicios para prevenir posibles incongruencias:
`python ordenes.py -c`
o
`python ordenes.py -clean`

1. Crea un directorio llamado **nuevo**.
2. Copia el fichero **b.txt** del directorio **antiguo** en el nuevo directorio que acabas de crear.
3. Duplica el fichero **b.txt** en la misma ruta pero con el nombre **bueno.txt**.
4. Copia el directorio **antiguo/num** al directorio **nuevo**.
5. Vuelve al directorio inicial (**ordenes_basicas**) y ejecuta el primer control intermedio:
`python ordenes.py -p 1`
o
`python ordenes.py -partial 1`

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Copiar ficheros, duplicarlos, crear directorios	Mover ficheros, borrarlos,...

6. Borra el fichero 3.txt dentro de nuevo/num.
7. Ve al directorio nuevo/num.
8. Ejecuta la orden pwd y guarda el resultado en el fichero pwd.res en el directorio inicial (ordenes_basicas).
9. Vuelve al directorio ordenes_basicas y crea un directorio doble con dos subdirectorios internos: primero y segundo.
10. Mueve el fichero 1.txt de nuevo/num a primero.
11. Enlaza el fichero enlacea2.txt de segundo al fichero 2.txt de nuevo/num. **Ojo:** El enlace debe ser duro, no simbólico (aunque puedes probar a hacer algún enlace simbólico para ver que es igual de simple).
12. Lista el directorio nuevo y guarda el resultado en el fichero nuevo.res.
13. Vuelve al directorio inicial (**ordenes_basicas**) y ejecuta el segundo control intermedio:
`python ordenes.py -p 2`
o
`python ordenes.py -partial 2`

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Copiar ficheros, duplicarlos, crear directorios, mover ficheros, borrarlos,...	Limpieza final del directorio.

14. Borra todos los ficheros y directorios nuevos que has creado desde que empezaste los ejercicios:

- El directorio nuevo.
- El fichero pwd.res.
- El fichero nuevo.res.
- El directorio doble.

Tras la realización de estos ejercicios debes tener clara la utilización y el manejo de las siguientes órdenes:

ls ✎ cp ✎ cd ✎ mv ✎ rm

Si aún sigues teniendo dudas puedes:

- Releer el artículo **órdenes_básicas** de nuevo al completo, o tan sólo las secciones en las que tengas dudas.
- Volver a desarrollar los ejercicios de nuevo.
- Revisar los artículos creados en el wiki para cada una de las órdenes.
- Revisar el manual de las órdenes implicadas (recuerda que tienes los enlaces en los artículos del wiki de cada una y además en está el manual del sistema).

Capítulo 6

Sistema de archivos

Información de los directorios base que conforman el sistema de archivos de los sistemas Unix	
TIEMPO:	3 horas
NIVEL:	Usuario iniciado (página 33)
INFORMACIÓN EXTRAÍDA DE:	Organizacion de los directorios en Linux ✒ Introducción a Linux. Directorios y archivos en Wikilibros
PRERREQUISITOS:	Jerarquía de directorios ✒ Soporte para pruebas

Este artículo consta de las siguientes secciones:

6.1. Introducción

Todo en el sistema Unix son archivos, por lo que el sistema de archivos es indispensable para el éxito y utilidad del sistema Unix.

Es la parte del núcleo más visible por los usuarios; se encarga de abstraer propiedades físicas de diferentes dispositivos para proporcionar una interfaz única de almacenamiento: el archivo. Cada sistema Unix tiene su sistema de archivos nativo (por ejemplo, **ext2** en Linux, **UFS** en Solaris o **EFS** en IRIX), por lo que para acceder a todos ellos de la misma forma el núcleo de Unix incorpora una capa superior denominada VFS (Virtual File System) encargada de proporcionar un acceso uniforme a diferentes tipos de sistema de ficheros.

6.2. Conceptos básicos sobre archivos

El sistema no impone estructura alguna a los archivos, ni asigna significado a su contenido; el significado de los bytes depende únicamente de los programas que interpretan el archivo. Esto es así tanto para archivos en disco como para dispositivos periféricos.

Cada byte de un archivo contiene un número de tamaño suficiente para representar un carácter. El código empleado en la mayoría de los sistemas Unix es ASCII (*Código Norteamericano Estándar para Intercambio de Información*), pero algunos equipos, entre los que sobresalen los de IBM, usan un código llamado EBCDIC (*Código Extendido de Intercambio Decimal Codificado en Binario*).

Los programas obtienen los datos de un archivo por medio de una llamada al sistema (una rutina del núcleo) llamada read . Cada vez que se invoca a read , esta regresa la siguiente porción de un archivo, la siguiente línea de texto tecleada en la terminal , por ej. read también indica cuantos bytes trae el archivo, por lo que al final del archivo es identificado en el momento en que read dice " se traen cero bytes". Si se hubieran quedado algunos bytes en el archivó, read los hubiera leído. En realidad , tiene sentido no representar el final de un archivo con algún valor en especial, ya que, como se menciono anteriormente , el significado de los bytes depende de como se vaya a interpretar el archivo. Pero todos los archivos tienen un final, y como todos los archivos deben accesarse por medio de read , regresar un cero es una manera de representar el final de un archivo (independientemente de cualquier representación) sin introducir ningún carácter especial.

6.2.1. Tipos de archivo

En un sistema Unix típico existen tres tipos básicos de archivos:

- **ficheros planos**: son secuencias de bytes que a priori no poseen ni estructura interna ni contenido significante para el sistema: su significado depende de las aplicaciones que interpretan su contenido.
- **directorios**: archivos cuyo contenido son otros ficheros de cualquier tipo (planos, más directorios, o ficheros especiales).
- **ficheros especiales** (dispositivos): ficheros que representan dispositivos del sistema.
  - dispositivos orientados a carácter
  - dispositivos orientados a bloque

La principal diferencia entre ambos es la forma de realizar operaciones de entrada/salida: mientras que los dispositivos orientados a carácter las realizan byte a byte (esto es, carácter a carácter), los orientados a bloque las realizan en bloques de caracteres.

Nota: Generalmente, al hablar de ficheros nos solemos referir a todos ellos si no se especifica lo contrario.

6.2.2. Sistemas de ficheros

Respecto a los sistemas de ficheros soportados debemos destacar la gran variedad de ellos, actualmente podemos encontrar (entre otros):

- **Sistemas asociados a GNU/Linux**: como el estándar *ext2* y el *ext3*, evolución del anterior con concepto de *journaling* (soporte de log de operacionesrealizadas en el sistema de fichero que puede permitir su recuperación en caso de algún desastre que lo haga inconsistente).
- **Compatibilidad con entornos no GNU/Linux**: *msdos*, *vfat*, *ntfs*, acceso a los diferentes sistemas de *fat16*, *fat32* y *ntfs*. En ciertos casos está limitado a lectura, pero existen soluciones en espacio de usuario (mediante FUSE, un componente que permite escribir sistemas de ficheros en espacio de usuario). También se dispone de compatibilidad a otros entornos como Mac con *hfs* y *hfsplus*.
- **Sistems asociados a soportes físicos**: como CD/DVD como los *iso9660* y *udf*.
- **Sistemas usados en diferentes Unix**: ofrecen generalmente mejor rendimiento (a veces a costa de mayor consumo de recursos, en CPU por ejemplo), como *JFS2* (IBM), *XFS* (SGI) o *ReiserFS*.
- **Sistemas de ficheros en red**: *NFS*, Samba (*smbfs*, *cifs*) permiten acceder a sistemas de ficheros disponibles en otras máquinas de forma transparente por red.
- **Sistemas distribuidos en red**: como *GFS*, *Coda*.
- **Pseudo Sistemas de ficheros**: como *procfs* (*/proc*) o *sysfs* (*/sys*).

6.3. Permisos

Cada archivo tiene un conjunto de permisos asociados con él, los cuales determinan qué puede hacerse con el archivo y quien puede hacerlo.

Existe un usuario especial en cada sistema UNIX, llamado *superusuario*, quien puede leer o modificar cualquier archivo en el sistema. La clase de acceso especial root posee privilegios de superusuario: esta la emplean los administradores del sistema cuando llevan a cabo el mantenimiento del mismo. Existe también un comando llamado su que otorga un status de superusuario si se conoce la contraseña de root. Por eso no es recomendable guarda ningún material confidencial en el sistema de archivos.

Existen tres tipos de permisos para cada archivo:

- ✦ lectura (*r*): examinar su contenido
- ✦ escritura (*w*): modificar su contenido
- ✦ ejecución (*x*)

Se puede aplicar un permiso diferente a cada persona. Como dueño (*user*) de un archivo, el lector posee un conjunto de permisos de lectura, escritura y ejecución. Su grupo (*group*) tiene otro conjunto. Los demás (*others*) tienen un tercer conjunto.

Nota: Los permisos para borrar archivos son independientes del archivo mismo. Si el usuario tiene permiso de escritura en un directorio, puede borrar archivos contenidos en éste, incluso los que estén protegidos contra escritura.

6.3.1. Notación simbólica

El esquema de notación simbólica se compone de 10 caracteres, donde el primer carácter indica el tipo de fichero:

Valor	***Descripción***
-	Fichero regular
d	Directorio
b	Fichero especial como dispositivo de bloque
c	Fichero de carácter especial
l	Enlace simbólico
p	Tubería nombrada (FIFO)
s	Socket

Entonces el esquema completo de la notación sería:

Tipo de fichero							***Permisos***																	
							User						**Group**						**Others**					
							Read		Write		Exec		Read		Write		Exec		Read		Write		Exec	
-	d	b	c	l	p	s	r	-	w	-	x	-	r	-	w	-	x	-	r	-	w	-	x	-

6.3.2. Notación octal

La notación octal se compone de valores de tres a cuatro dígitos en base 8. Con la notación octal de tres dígitos cada número representa un componente diferente de permisos a establecer: clase de *usuario*, clase de *grupo* y clase de *otros* respectivamente. Cada uno de estos dígitos es la suma de sus bits que lo componen (en el sistema numeral binario). Como resultado, bits específicos se añaden a la suma conforme son representados por un numeral:

- ✦ El bit de ejecución añade 1 a la suma.

- El bit de escritura añade 2 a la suma
- El bit de lectura añade 4 a la suma.

Estos valores nunca producen combinaciones ambiguas y cada una representa un conjunto de permisos específicos, que se pueden observar en la siguiente tabla:

Valor	***Notación simbólica***	***Permisos***
0	-	Ninguno
1	x	Ejecución
2	w	Escritura
3	wx	Escritura, ejecución
4	r	Lectura
5	rx	Lectura, ejecución
6	rw	Lectura, escritura
7	rwx	Lectura, escritura, ejecución

6.3.3. Permisos adicionales

Acabamos de hablar de la forma de tres dígitos, pero hay otra de cuatro dígitos. Bajo este esquema el estándar de tres dígitos anterior se convierte en los últimos tres dígitos del conjunto. El dígito restante (el primero) representa los permisos adicionales.

Nota: En los casos en que este primer dígito no puede omitirse porque tenga que mostrarse el conjunto de cuatro completo, se establece cero para dicho valor.

- **Permiso SUID** (o *bit setuid*): cuando se ha establecido ejecución, el proceso resultante asumirá la identidad del propietario.
- **Permiso SGID** (o *bit setgid*): cuando se ha establecido ejecución, el proceso resultante asumirá la identidad del grupo al que pertenece el propietario. Cuando se aplica a un directorio, todos los nuevos ficheros creados debajo de este directorio heredarán el grupo propietario de este mismo directorio.

  Nota: Cuando no se ha establecido setgid, el comportamiento predefinido es asignar el grupo del usuario al crear nuevos elementos.
- **Bit *sticky***: un usuario solo podrá modificar y eliminar ficheros y directorios subordinados dentro de un directorio que le pertenezca. Los directorios a los cuales se les ha establecido bit *sticky* restringen las modificaciones de los usuarios. Así cada usuario mantiene el control total sobre sus propios ficheros pudiendo crear nuevos ficheros; sin embargo, solo puede adjuntar o añadir contenido a los ficheros de otros usuarios. Se utiliza en directorios como */tmp* y */var/spool/mail*.

  Nota: En ausencia del bit *sticky* se aplican las reglas generales y el derecho de acceso de escritura por sí solo permite al usuario crear, modificar y eliminar ficheros y directorios subordinados dentro de un directorio.

Cuando un fichero no tiene permisos de ejecución en alguna de las clases y se le es asignado un permiso especial, éste se representa con una letra mayúscula.

Permiso	***Afectado***	***Ejecuta***	***No ejecuta***
SETUID	Usuario	s	S
SETGID	Grupo	s	S
Sticky	Otros	t	T

Al igual que en el formato de tres dígitos, el primer dígito del conjunto de cuatro es también la suma de los bits que lo componen:

- El bit *sticky* añade 1 a la suma.
- El bit *setgid* añade 2 a la suma.
- El bit *setuid* añade 4 a la suma.

Así en formato octal, tendríamos:

Valor	***Notación simbólica***	***Permisos***
1	— — –t	Bit *sticky*
2	— -s- —	Bit SETGID
3	— -s- –t	Bit SETGID y *sticky*
4	s– — —	Bit SETUID
5	s– — –t	Bit SETUID y *sticky*
6	s– -s- —	Bit SETUID y SETGID
7	s– -s- –t	Bit SETUID, SETGID y *sticky*

6.3.4. Órdenes

Para controlar los permisos que queremos establecer en ficheros y directorios, nos serán útiles las siguientes órdenes (en todos los casos puede resultar más cómodo el modo recursivo *-R*).

Nota: Recuerda que el caracter *x* denota permiso de ejecución en el caso de los ficheros y permiso de acceso (búsqueda) para los directorios.

Si queremos ser más específicos en cuanto a qué ficheros/directorios queremos modificar en este sentido. podemos combinar estas órdenes con otras como **find**.

chown

Sirve para cambiar propietario (y grupo) de los archivos y directorios de nuestro sistema.

La sintaxis es: `chown [OPCIÓN]... [PROPIETARIO][:[GRUPO]] FICHERO...`

Por ejemplo:

```
chown www-data:www-data /var/www/index.html chown -R www-data:www-data
/var/www/*
```

chgrp

Sirve para cambiar el grupo propietario de los ficheros.

La sintaxis es: `chgrp [OPCIÓN]... GRUPO ARCHIVO...`

chmod

Sirve para cambiar permisos específicos (rwx) de los archivos, pudiendo especificarse el modo tanto en notación simbólica como en notación octal (simplemente introduciendo el valor numérico).

La sintaxis es:

chmod [OPCIÓN]... MODO... FICHERO...

La notación simbólica se especifica de la forma **[ugo[[Categoría:Programas]]*([-+=]([rwxXs[[Categoría:Program**
siendo:

- Clases de usuarios:
  - **u**: usuario propietario
  - **g**: grupo propietario
  - **o**: otros
  - **a**: todos
- Permisos básicos:
  - **r**: lectura
  - **w**: escritura
  - **x**: ejecución/acceso
- Caracteres especiales:
  - **+**: añade un permiso
  - **-**: elimina un permiso
  - **=**: especifica un nuevo modo sobreescribiendo el anterior

Por ejemplo:

1. Dar permisos de ejecución a todos los usuarios (no es necesario especificar *a*):

   ```
   chmod +x script.py
   ```

II. Dar todos los permisos a todos los usuarios en notación octal:

```
chmod 777 visible.txt
```

III. Dar todos los permisos al propietario, de ejecución al grupo y ninguno a los demás:

```
chmod u+rwx,g+x,o= fich.txt
```

6.4. Nodo-i

Un archivo está compuesto de:

- un nombre
- contenido
- información administrativa como permiso
- fechas de modificación

La **información administrativa** está almacenada en el *nodo-i*(también se suele llamar *inodo*), junto con datos esenciales para el sistema tales como su longitud, la región del disco en la que se encuentra almacenado el contenido del archivo y otros elementos.

Existen tres fechas en un inodo:

- la fecha en la que se hizo la última modificación (escrita) al contenido del archivo
- la fecha en la que dicho contenido fue usado (leído o ejecutado) por última vez
- la fecha en la que el inodo fue alterado por última vez, por ejemplo, para definir los permisos

El nombre de archivo en un directorio se llama enlace (*link*), ya que une un nombre en la jerarquía de directorio al inodo y, en consecuencia, a los datos. El mismo numero-i puede aparecer en más de un directorio.

Por lo tanto, el inodo es un registro que almacena información sobre el archivo determinado y contiene:

I. Identificación de usuario y de grupos de archivos.

II. Instantes del ultimo acceso y de la ultima modificación.

III. Contador con él numero de HORD-LINK al archivo.

IV. El tipo de archivo.

V. 15 apuntadores a bloques de disco.

Los primeros 12 bloques apuntan a bloques directos, o sea que se puede referenciar inmediatamente a 12 directorios de bloques de datos de archivos (ya que existe una copia del inodo en memoria principal, mientras el archivo está abierto).

Los siguientes tres apuntan a bloques indirectos (del tamaño del bloque grande):

- El primero es la dirección de un bloque indirecto simple (bloques de direcciones de bloques de datos)
- El segundo apunta a un bloque indirecto doble (bloque de direcciones de bloques que apuntan a bloques de datos)
- El tercero apunta a un bloque indirecto triple (no se lo necesita)

6.4.1. Acceso

Si el primer carácter del nombre del camino es `/`.[es] el directorio raiz, sino el directorio de partida es el proceso actual. El final es el nombre de un archivo, se realiza el proceso en busca de este nombre y si no se lo encuentra, se emite un mensaje de error.

Para los archivos que no estén en disco se designan controladores apropiados para manejar su entrada/salida.

6.4.2. Mapeado

Se utiliza para indexar en una tabla de archivos abiertos el proceso actual. Cada entrada en la tabla contiene unos apuntados a una estructura de archivos, que a su vez apunta al inodo.

La estructura del inodo es un copia en memoria que hay en disco, con campos extras.

6.5. Ejercicios

ASPECTOS PEDAGÓGICOS: Permiten la modificación de permisos y propietarios de los ficheros. Son muy útiles para la comprensión a fondo del sistema de permisos que existe en este tipo de sistemas y para valorar un buen control de éstos, así como qué importancia tienen.
PREPARACIÓN PREVIA: Para hacer estos ejercicios debes acceder al sistema con el usuario por defecto *usuario* (recuerda que la contraseña es *unix*). A continuación:

- Ve al directorio casa del usuario actual. Comprueba que existe un directorio llamado **unix** y dentro de éste un subdirectorio llamado **archivos**. Para ello ejecuta:
  ```
  ls unix
  ```
  Debe aparecer el directorio **archivos**. En caso de que no exista alguno de estos elementos, puedes hacer varias cosas:

1. Descargar el archivo comprimido con la jerarquía de elementos de comprobación que se encuentra en la portada del wiki y descomprimirlo en el directorio casa del usuario actual.
2. Apagar la máquina virtual y restaurar el *snapshot* que funcionaba bien.
3. Apagar la máquina virtual y borrarla eliminando el disco duro virtual. A continuación, descargar el disco duro y volver a configurar la máquina de nuevo.

✦ Ve al subdirectorio **unix/archivos**. Limpia el sistema antes de realizar los demás ejercicios para prevenir posibles incongruencias:
```
sudo python archivos.py -c
```
o
```
sudo python archivos.py -clean
```

1. Consulta qué usuarios y grupos posee el directorio **inicio** y todo su contenido.

   Ojo: Apúntalo si es necesario o guarda los resultados en un fichero de texto, puesto que esta información te será útil más adelante.

2. Consulta qué permisos posee el directorio **inicio** y todo su contenido.

3. Consulta cuál es la máscara de permisos actual.

   Pista: Existe una orden para visualizar y modificar la máscara de permisos de usuario (piensa en inglés).

4. Modifica la máscara de permisos para que tan sólo puedan leer y modificar ficheros el usuario y grupo propietarios.

   Ojo: El resto de permisos deben estar desactivados, no son irrelevantes.

5. Crea un archivo *foo.bar* en el directorio **archivos** y comprueba que la máscara del sistema funciona.

6. Vuelve al directorio inicial (**archivos**) y ejecuta el primer control intermedio:
```
sudo python archivos.py -p 1
```
o
```
sudo python archivos.py -partial 1
```

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Máscara de permisos	Cambiar permisos

7. Cambia el usuario y el grupo propietarios del directorio **inicio/root** y todo su contenido por el administrador.

8. Cambia los permisos de los ficheros (únicamente los ficheros, no los directorios) que se encuentran en **inicio** y en **inicio/valores** para que todos los usuarios puedan leer y ejecutar los ficheros, pero ninguno pueda modificarlos.

9. Cambia el grupo propietario del directorio **inicio/tmp** y todo su contenido por *www-data*.

10. Dale permisos de lectura (únicamente) a todos los usuarios al fichero *passwd.dat* que se encuentra en **inicio/tmp/root**.

11. Cambiar el usuario propietario de los ficheros contenidos en **inicio/valores** por *invitado* y el grupo propietario por *users*.

12. Añade pemisos de ejecución para el usuario propietario al fichero *foo.bar*.

13. Vuelve al directorio inicial (**archivos**) y ejecuta el segundo control intermedio:
    `sudo python archivos.py -p 2`
    o
    `sudo python archivos.py -partial 2`

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Máscara de permisos y cambio de permisos y usuarios/grupos propietarios.	Limpieza final del directorio: restaurar permisos y propietarios.

14. Borra el fichero *foo.bar* que ya no es necesario.

15. Restaura el usuario y el grupo propietario para el directorio **inicio** y todo su contenido.

16. Restaura los permisos básicos para el directorio **inicio** y todo su contenido.

17. Restaura la máscara de permisos de usuario.

Tras la realización de estos ejercicios debes tener clara la utilización y el manejo de las siguientes órdenes:

chmod ✒ chown ✒ chgrp ✒ otro

Si aún sigues teniendo dudas puedes:

- ✦ Releer el artículo **Sistema de archivos** de nuevo al completo, o tan sólo las secciones en las que tengas dudas.
- ✦ Volver a desarrollar los ejercicios de nuevo.
- ✦ Revisar los artículos creados en el wiki para cada una de las órdenes.
- ✦ Revisar el manual de las órdenes implicadas (recuerda que tienes los enlaces en los artículos del wiki de cada una y además en está el manual del sistema).

Parte III

Nivel usuario habitual

Capítulo 7

Editor VI

Referencia breve a la utilización del editor VI, que está presente en (casi) cualquier sistema Unix por defecto.	
TIEMPO:	2-3 horas
NIVEL:	Usuario habitual (página 57)
INFORMACIÓN EXTRAÍDA DE:	An Introduction to Display Editing with Vi ✒ Manual del editor VI (español) ✒ Mastering the VI editor
PRERREQUISITOS:	Conceptos básicos ✒ Terminal

Este artículo consta de las siguientes secciones:

7.1. Introducción

Vi(Visual) es un simple editor de texto, que no lo formatea en absoluto, pues no centra ni justifica párrafos pero permite mover, copiar, eliminar o insertar caracteres por medio del búfer permaneciendo la información ahí hasta que los cambios en el archivo se hayan guardado o bien hasta que termine la ejecución de la aplicación sin haber guardado las modificaciones.

7.2. Modos

En el editor VI se pueden elegir diferentes modos en función de la acción que se realizará al recibir una instrucción. Estos modos se pueden agrupar distinguir en dos: modo edición y modo comando, los cuales veremos a continuación más en profundidad.

7.2.1. Modo texto

El modo texto contiene a su vez otros modos, bastante similares entre sí pero esta compuesto por varios modos que son muy similares pero con pequeñas diferencias. Estos modosestos modos son el modo insertar, agregar, abierto y reemplazar. Sin profundizar demasiado en ellos, podemos decir que lo que escribas en alguno de estos modos será para la edición del texto del archivo.

7.2.2. Modo comando

El editor comienza en modo comando, en el cuál puedes tanto mover el cursor como borrar y pegar texto. El modo inserción comienza introduciendo un comando de inserción o modificado de texto. [ESC] devuelve al editor a modo comando (desde el que te puedes salir por ejemplo tecleando :q!). La mayoría de los comandos se ejecutan tan pronto como los tecleas a excepción de los comandos "dos puntos" los cuáles se ejecutan cuando pulsas la tecla RETURN.

7.3. Acciones

7.3.1. Salir

- ✦ Salir, guardando los cambios **:x**
- ✦ Salir (si no se han realizado cambios) **:q**
- ✦ Salir (fuerza, aunque no se haya salvado) **:q!**

7.3.2. Inserción de texto

- ✦ Insertar antes del cursor, antes de la línea **i, I**

- Añadir después del cursor, al final de la línea **a, A**
- Añadir una línea por debajo, por encima **o, O**
- Remplazar un carácter, muchos caracteres **r, R**

7.3.3. Moverse

- Izquierda, abajo, arriba, derecha **h,j,k,l**
- Siguiente palabra, palabra delimitada por blanco **w, W**
- Principio de palabra, de palabra delimitad. por blanco **b, B**
- Final de palabra, de palabra delimitada por blanco **e, E**
- Sentencia hacia atrás, hacia delante **(,)**
- Párrafo hacia atrás, hacia delante **{, }**
- Principio, al final de la línea **0, $**
- Principio, al final del fichero **1G, G**
- Linea n **nG o :n**
- Hacia atrás, hacia delante hasta el carácter c **fc, Fc**
- Parte superior, media y baja de la pantalla **H,M,L**

7.3.4. Borrar texto

Casi todos los comandos de borrado se realizan tecleando **d** seguido de un comando de movimiento. Por ejemplo, **dw** borra una palabra. Otros pocos comandos de borrado son:

- Carácter de la derecha, de la izquierda **x, X**
- Hasta el final de línea **D**
- Línea **dd**
- Línea **:d**

7.3.5. Pegar texto

Al igual que en el borrado, casi todos los comandos de pegado se realizan tecleando **y** seguido de un comando de movimiento. Por ejemplo, **y$** pega hasta el final de línea. Otros dos comandos de pegado son:

- Línea **yy**
- Línea **:y**

7.3.6. Cambiar texto

El comando de cambio es un comando de borrado que deja al editor en modo inserción. Se realiza tecleando **c** seguido de un comando de movimiento. Por ejemplo **cw** cambia una palabra. Otros comandos de cambio son:

- Hasta el final de la línea **C**
- Línea **cc**

7.3.7. Pegar texto

- Pegar después de la posición o después de la línea **p**
- Pegar antes de la posición o antes de la línea **P**

7.3.8. Buffers

Se puede especificar el nombre de un buffer antes de cualquier borrado, cambio, copiado o pegado. El prefijo general tiene la forma **"c** donde c podría ser cualquier letra minúscula. Por ejemplo, **"adw** borra una palabra y la guarda en el buffer a. Podría ser esta palabra devuelta al texto con un comando de pegado adecuada, por ejemplo **"ap**.

7.3.9. Marcas

Las marcas nominales pueden ser colocadas sobre cualquier línea del fichero. Cualquier letra minúscula puede ser el nombre de una marca. Las marcas podrían también ser utilizadas como límites para rangos. Poner la marca c en esta línea mc

- Ir a la marca c **'c**
- Ir al primer carácter no blanco de la marca c **'c**

7.3.10. Búsqueda de Cadenas

- Buscar hacia delante **/cadena**
- Buscar hacia atrás **?cadena**
- Repetir la búsqueda en la misma, distinta dirección **n, N**

7.3.11. Remplazar

La Función de búsqueda y remplazamiento se realiza con el comando **:s**. Se usar normalmente en combinación con rangos o el comando **:g** (más abajo):

- Remplaza patrón con cadena **:s/patrón/cadena/opción**
- Opciones: varias en la misma línea, confirmación **g, c**
- Repetir el último comando **:s&**

7.3.12. Expresiones Regulares

- Cualquier carácter único excepto el salto de línea **.** (punto)
- Cero o más repeticiones ∗
- Cualquier carácter del conjunto **[...]**
- Cualquier carácter que no sea del conjunto **[^ ...]**
- Principio, final de línea **^, $**
- Principio, final de palabra **\<, \>**
- Agrupación **\(...\)**
- Contenido del agrupamiento n **\n**

7.3.13. Números

Casi todos los comandos pueden ser precedidos por un número que especifica cuántas veces va a ser realizado. Por ejemplo **5dw** borrará 5 palabras y **3fe** moverá el cursor hacia delante hasta la tercera ocurrencia de la letra e. Incluso las inserciones pueden ser repetidas de forma conveniente con este método, pudiéndose insertar la misma línea 100 veces.

7.3.14. Rangos

Los rangos pueden preceder a la mayoría de los comandos “dos puntos” y hacer que dichos comandos se ejecuten sobre un intervalo de líneas determinado. Por ejemplo :3,7d eliminará las lineas de la 3 a la 7. Los rangos son combinados frecuentemente con el comando :s para realizar una substitución en varios líneas, como con :.,$s/patrón/cadena/g para hacer una substitución desde la línea actual hasta el final del fichero.

- Líneas de la n a la m (ambas inclusive) **:n,m**
- Línea actual **:.**

- Última línea **:$**
- Marcador c **:'c**
- Todas las líneas del fichero **:%**
- Todas las líneas que encajen con el patrón **:g/patrón/**

7.3.15. Ficheros

- Escribir a fichero (el actual si no se especifica fichero) **:w fichero**
- Leer el fichero después de la línea actual **:r fichero**
- Siguiente fichero **:n**
- Fichero anterior **:p**
- Editar fichero **:e fichero**
- Remplazar la línea con la salida del programa **!!programa**

7.3.16. Otros

- Cambiar entre mayúsculas y minúsculas ~
- Unir líneas **J**
- Repetir el último comando de cambio de texto **.**
- Deshacer el último cambio, de la línea actual **u, U**

Capítulo 8

Órdenes

Compendio de las órdenes de un sistema Unix, con su modo de utilización y un enlace al manual del sistema	
TIEMPO:	6 horas
NIVEL:	Usuario habitual (página 57)
INFORMACIÓN EXTRAÍDA DE:	Documentación básica y manual del sistema ✒ Ubuntu Manpages
PRERREQUISITOS:	Órdenes básicas ✒ Soporte para pruebas

Este artículo consta de las siguientes secciones:

8.1. Introducción

Normalmente se suelen emplear muy a menudo unas pocas órdenes (sobre todo en relación a la cantidad de programas disponibles), sin embargo, no está de más ver una referencia algo más completa de las órdenes clásicas de Unix, con algunos ejemplos de uso bastante útiles.

A continuación, tan sólo se expone un listado de las órdenes. Si quieres saber más información acerca de ellas utiliza las herramientas del sistema o entra en el wiki.

8.2. Referencia general de órdenes

8.2.1. Órdenes varias

passwd *clear* *banner* *echo* *set* *env* *su* *history*

8.2.2. Órdenes de conexión / desconexión

login *poweroff* *reboot* *halt*

8.2.3. Órdenes de comunicación entre usuarios

write *mesg* *mail* *mailx* *wall* *rmail*

8.2.4. Órdenes de control de terminal

stty

8.2.5. Órdenes de información

General

man *help* *whereis* *whatis*

Fecha y hora

date *cal*

De usuarios

who ❧ finger ❧ logname ❧ id ❧ tty

Del sistema

uname

Procesos

ps

Uso de disco

df ❧ du

8.2.6. Operaciones con archivos

Archivos

ar ❧ cpio ❧ tar

Acceso a archivos

chmod ❧ chgrp ❧ newgrp ❧ umask

Borrado

rm

Búsqueda

find ❧ grep ❧ egrep ❧ fgrep

Cambio de nombre

mv

Clasificación

file

Comparación

cmp ❧ comm ❧ diff

Copia

cp ❧ cat ❧ ln

Edición

ed ❧ vi ❧ sed ❧ red

Formato

pr ❧ dd ❧ col

Listado

ls ❧ dir ❧ vdir

Mantenimiento del sistema de archivos

touch ❧ sum ❧ uniq

Movimiento

mv

Proceso de archivos por campo

awk ❧ cut

Proceso de archivos por línea

sed ❧ awk ❧ tr ❧ pr ❧ nl

Contador de palabras

wc

Separación de archivos

cut ❧ split ❧ csplit

Unión

cat ❧ paste ❧ join

Visualización

cat ❧ more ❧ tail ❧ head ❧ pr ❧ nl ❧ od

8.2.7. Operaciones con directorios

cd ❧ pwd ❧ ls ❧ dir ❧ vdir ❧ mkdir ❧ rmdir ❧ ln ❧ dirname

8.2.8. Ordenación

sort

8.2.9. Procesos

ps ❧ sh ❧ at ❧ atq ❧ atrm ❧ batch ❧ nohup ❧ kill ❧ tee ❧ trap ❧ nice ❧ time

8.3. Ejemplos

Órden	***Descripción***
`apropos <palabra>`	Ver órdenes relacionadas con <palabra>

continúa en la siguiente página...

continúa desde la página anterior...

Órden	***Descripción***
`which <orden>`	Ver la ruta completa de <orden>
`time <orden>`	Medir el tiempo que tarda <orden>en ejecutarse
`nice info`	Ejecutar orden (**info** en este caso) con prioridad baja
`renice 19 -p $$`	Dar prioridad baja al shell elegido. Usar para tareas no interactivas.

Cuadro 8.1: Ejemplos de órdenes: generales

8.3.1. Navegación

Órden	***Descripción***
`cd -`	Volver al directorio anterior
`cd`	Ir al directorio casa
`(cd <directorio>&& <orden>)`	Ir a <directorio>, ejecutar <orden>y volver al directorio inicial
`pushd .`	Guardar el directorio actual en la pila. Si ejecutamos posteriormente **popd** volveremos al mismo.

Cuadro 8.2: Ejemplos de órdenes: navegación

8.3.2. Búsquedas

Órden	***Descripción***
`alias l='ls -l -color=auto'`	Crear un alias para listar el directorio al introducir **l**
`ls -lrt`	Listar archivos por fecha
`find -name '*.[ch]'`	Buscar <expre>en este directorio y subdirectorios
`find -type f -print0`	Buscar <ejemplo>en todos los archivos regulares en este directorio y subdirectorios
`find -maxdepth 1 -type f`	Buscar <ejemplo>en todos los archivos regulares de este directorio
`find -type f ! -perm -444`	Buscar archivos sin permiso general de lectura
`find -type d ! -perm -111`	Buscar directorios sin permiso general de acceso
`locate -r 'file [ ]*\ .txt'`	Buscar nombres en índice en cache (la expresión regular es igual a $*file*.txt$)
`look <ref>`	Búsqueda rápida (ordenada) de prefijo en diccionario
`grep -color <ref>/usr/share/dict/pal`	Resaltar ocurrencias de expresión regular en diccionario

continúa en la siguiente página...

continúa desde la página anterior...

Órden	***Descripción***

Cuadro 8.3: Ejemplos de órdenes: búsquedas

8.3.3. Archivos

Órden	***Descripción***
`gpg -c <fichero>`	Encriptar <fichero>
`gpg <fichero>.gpg`	Desencriptar <fichero>
`tar -c <directorio>/ | bzip2 ><directorio>.tar.bz2`	Crear archivo compacto de <directorio>
`bzip2 -dc <directorio>.tar.bz2 | tar -x`	Extraer archivo compacto (usar **gzip** en vez de **bzip2** para archivos *tar.gz*)
`tar -c <dir>/ | gzip | gpg -c | ssh <usuario>@<equipo>'dd of=<dir>.tar.gz.gpg'`	Crear compactado encriptado de <dir>en <equipo>
`find <dir>/ -name '*.txt' | tar -c -files-from=- | bzip2 >dir_txt.tar.bz2`	Crear compactado de subconjunto de <dir>y subdirectorios
`find <dir>/ -name '*.txt' | xargs cp -a -target-directory=dir_txt/ -parents`	Copiar subconjunto de <dir>y subdirectorios
`( tar -c <dir_a_copiar>) | ( cd /<dir>/ && tar -x -p )`	Copiar (con permisos) directorio <dir_a_copiar>a directorio <dir>
`( cd <dir_a_copiar>&& tar -c . ) | ( cd /<dir>/ && tar -x -p )`	Copiar (con permisos) contenido del directorio <dir_a_copiar>a directorio <dir>
`( tar -c <dir_a_copiar>) | ssh -C <usuario>@<equipo>'cd /<dir>/ && tar -x -p'`	Copiar (con permisos) directorio <dir_a_copiar>a directorio remoto <dir>
`dd bs=1M if=/dev/hda | gzip | ssh <usuario>@<equipo>'dd of=hda.gz'`	Respaldo de disco duro en equipo remoto

Cuadro 8.4: Ejemplos de órdenes: archivos

8.3.4. Descargas con wget

Órden	***Descripción***
`(cd cmdline && wget -nd -pHEKk http://<dir_web>.html)`	Guardar en directorio actual una versión navegable de una página web
`wget -c http://<dir_archivo>`	Retomar descarga de un archivo parcialmente descargado

continúa en la siguiente página...

continúa desde la página anterior...

Órden	***Descripción***
`wget -r -nd -np -l1 -A '*.jpg' http://<dir>`	Descargar una serie de archivos en el directorio actual
`wget ftp://<dir_equipo>/archivo[1-9].iso`	El protocolo FTP permite globalizaciones directas
`wget -q -O- http://www.pixelbeat.org/timeline.html | grep 'a href' | head`	Procesando directamente la salida
`echo 'wget <url>' | at 01:00`	Descargar la URL a 1AM al directorio actual
`wget --limit-rate=20k <url>`	Hacer descargas de baja prioridad (en este caso, no exceder los 20KB/s)
`wget -nv --spider --force-html -i <url>`	Revisar los enlaces de una página
`wget --mirror <url>`	Actualizar eficientemente una copia local de una página web (útil si usamos **cron**)

Cuadro 8.5: Ejemplos de órdenes: descargas con wget

8.3.5. Redes

Órden	***Descripción***
`ethtool <interfaz>`	Mostrar estado de la interfaz
`ip link show`	Listar interfaces
`ip link set dev eth0 name wan`	Renombrar *eth0* a *wan*
`ip addr add 1.2.3.4/24 brd + dev eth0`	Agregar ip y máscara
`ip link set dev interface up`	Subir (o bajar) la interfaz
`ip route add default via 1.2.3.254`	Establecer 1.2.3.254 como valor por omisión para la puerta de enlace
`tc qdisc add dev lo root handle 1:0 netem delay 20msec`	Agregarle 20ms de espera al dispositivo de retorno (para hacer pruebas)
`tc qdisc del dev lo root`	Quitar la espera agregada antes
`host <IP_o_url>`	Obtener la dirección IP para el dominio o al revés
`hostname -i`	Obtener la dirección IP local
`netstat -tupl`	Listar los servicios de internet de un sistema
`netstat -tup`	Listar las conexiones activas desde/hacia un sistema

Cuadro 8.6: Ejemplos de órdenes: redes

8.3.6. Espacio en disco

Órden	*Descripción*
`ls -lSr`	Mostrar archivos, de menor a mayor
`du -s * | sort -k1,1rn | head`	Mostrar usuarios de disco principales en el directorio actual
`df -h`	Mostrar espacio libre de disco
`df -i`	Mostrar *nodos-i* libres
`fdisk -l`	Mostrar tamaños y tipos de particiones de disco
`rpm -q -a -qf '%10{SIZE}\t%{NAME}\n' | sort -k1,1n`	Listar todos los paquetes por tamaño instalado (Bytes) de distribuciones RPMs
`dpkg-query -W -f='${Installed-Size;10}\t${Package}\n' | sort -k1,1n`	Listar todos los paquetes por tamaño instalado (Kbytes) de distribuciones deb
`dd bs=1 seek=2TB if=/dev/null of=ext3.test`	Crear un gran archivo de prueba (sin ocupar espacio)

Cuadro 8.7: Ejemplos de órdenes: espacio en disco

8.3.7. Información del sistema

Órden	*Descripción*
`hdparm -i /dev/hda`	Ver informe sobre partición hda
`hdparm -tT /dev/hda`	Hacer una prueba de velocidad de lectura en partición hda
`badblocks -s /dev/hda`	Hallar bloques ilegibles en partición hda
`mount | column -t`	Ver particiones montadas en el sistema (y alinear la salida)
`cat /proc/partitions`	Ver todas las particiones registradas en el sistema
`grep MemTotal /proc/meminfo`	Ver el total de RAM que registra el sistema
`grep "model name/proc/cpuinfo`	Ver informe de CPU(s)
`lspci -tv`	Ver informe de PCI
`lsusb -tv`	Ver informe de USB

Cuadro 8.8: Ejemplos de órdenes: información del sistema

8.4. Ejercicios

ASPECTOS PEDAGÓGICOS: Permiten el aprendizaje de órdenes más complejas y a veces poco utilizadas comparadas con la utilidad que tienen.
PREPARACIÓN PREVIA: Para hacer estos ejercicios debes acceder al sistema con el usuario por defecto *usuario* (recuerda que la contraseña es *unix*). A continuación:

- ✦ Ve al directorio casa del usuario actual. Comprueba que existe un directorio llamado **unix** y dentro de éste un subdirectorio llamado **ordenes**. Para ello ejecuta:
  `ls unix`

Debe aparecer el directorio **ordenes**. En caso de que no exista alguno de estos elementos, puedes hacer varias cosas:

1. Descargar el archivo comprimido con la jerarquía de elementos de comprobación que se encuentra en la portada del wiki y descomprimirlo en el directorio casa del usuario actual.
2. Apagar la máquina virtual y restaurar el *snapshot* que funcionaba bien.
3. Apagar la máquina virtual y borrarla eliminando el disco duro virtual. A continuación, descargar el disco duro y volver a configurar la máquina de nuevo.

✦ Ve al subdirectorio **unix/ordenes**. Limpia el sistema antes de realizar los demás ejercicios para prevenir posibles incongruencias:
```
sudo python ordenes.py -c
```
o
```
sudo python ordenes.py -clean
```

1. Deseamos hacer una contabilidad de la información de ayuda incluida en nuestro sistema dentro del directorio */usr/share/doc*.

   Para ello almacene en un fichero llamado *stats-pdf*, el nombre de todos los ficheros con extensión **pdf** que estén ubicados dentro de él.

   Realice la operación recursivamente.

2. Después almacene en un fichero llamado stats-txt en el nombre de todos los ficheros con extensión **txt** que ocupen más de 100 bytes que estén ubicados dentro de él.

   Realice la operación recursivamente.

3. Vuelve al directorio inicial (**ordenes**) y ejecuta el primer control intermedio:
   ```
   sudo python ordenes.py -p 1
   ```
   o
   ```
   sudo python ordenes.py -partial 1
   ```

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Utilidad de la orden find	Utilidad de las órdenes tar y grep

4. También sería bueno tener una copia de todos los ficheros alojados en */usr/share/doc/lsb-release*. Hágala con tar + gz y almacene el resultado en un fichero llamado *backup.tgz* dentro del directorio actual.

   Ojo: Trasládate al directorio */usr/share/doc/lsb-release* en lugar de utilizar *tar* con la ruta absoluta, o la comprobación del ejercicio no saldrá correctamente.

5. Después restaure la información en un directorio llamado "prueba".

6. Vuelve al directorio inicial (**ordenes**) y ejecuta el segundo control intermedio:
   ```
   sudo python ordenes.py -p 2
   ```
   o
   ```
   sudo python ordenes.py -partial 2
   ```

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Utilidad de la orden tar	Utilidad de las órdenes grep

7. Queremos analizar el proceso de arranque de nuestro sistema. Para ello analizaremos la información en */var/log/dmesg* y generaremos ficheros que lo resuman en el directorio *tmp* de la raíz del sistema de ficheros:

   - Copie todas las líneas que contengan **Device** o **device** a un fichero llamado "dispositivos.txt".
   - Almacene todas las líneas que tienen la cadena *eth* pero no la palabra **driver** en red.txt"
   - Almacene todas las líneas que tengan la cadena *PCI* en cualquier combinación de mayúsculas y minúsculas en "pecei.txt"

Tras la realización de estos ejercicios debes tener clara la utilización y el manejo de las siguientes órdenes:

tar ✒ find ✒ grep

Si aún sigues teniendo dudas puedes:

- Releer el artículo **Órdenes avanzadas** de nuevo al completo, o tan sólo las secciones en las que tengas dudas.
- Volver a desarrollar los ejercicios de nuevo.
- Revisar los artículos creados en el wiki para cada una de las órdenes.
- Revisar el manual de las órdenes implicadas (recuerda que tienes los enlaces en los artículos del wiki de cada una y además en está el manual del sistema).

Capítulo 9

Instalación de software

En esta sección se trata de explicar cómo se instalan aplicaciones en sistemas UNIX y más específicamente en distribuciones de GNU Linux. Se basa sobre todo en instalación de paquetes DEB y RPM.	
TIEMPO:	2 horas
NIVEL:	Usuario habitual (página 57)
INFORMACIÓN EXTRAÍDA DE:	DEB en Wikipedia ✒ RPM en Wikipedia ✒ Cómo instalar programas en ubuntu-es ✒ Cómo instalar un RPM en Fedora
PRERREQUISITOS:	Conceptos básicos ✒ Órdenes básicas ✒ Órdenes avanzadas

Este artículo consta de las siguientes secciones:

9.1. Introducción

Antes de explicar las pautas a seguir, es necesario saber que un paquete de software no es más que código que se distribuye e instala conjuntamente. Nosotros vamos a trabajar comúnmente con dos tipos de paquetes:

RPM significa *RedHat Package Manager*, un administrador capaz de instalar, actualizar, desinstalar, verificar y solicitar programas previamente empaquetados. Estos programas empaquetados reciben el nombre de RPMs y están en un formato de paquetes estándar para Linux.

DEB es la extensión del formato de paquetes de software de Debian y derivadas, y el nombre más usado para dichos paquetes. Como Debian, su nombre proviene de Deborah Murdock, exesposa del fundador de la distribución Ian Murdock.

9.2. Instalación de paquetes DEB

Para instalar paquetes se utiliza **apt** desde la terminal y **synaptic**/**kynaptic** en modo gráfico (que no es más que una interfaz para facilitar el uso de *apt*; puedes acceder a ella a través de *Sistema >Administración >Gestor de Paquetes Synaptic*).

apt posee la capacidad de descargar los paquetes de los repositorios e instalarlos automáticamente. La lista de repositorios se encuentra en el archivo */etc/apt/sources.list*.

9.2.1. Instalación teniendo conexión a Internet

Antes de instalar cualquier paquete, es recomendable actualizar las definiciones de los paquetes, para disponer de las versiones más recientes a instalar. Esto se consigue ejecutando:

```
sudo apt-get update
```

Para actualizar nuestro sistema, apt puede descargar e instalar automáticamente todas las actualizaciones disponibles:

```
sudo apt-get upgrade
```

<font color="#483D8B»**Nota.**Si deseáramos actualizar nuestro sistema hasta una nueva versión de la distribución, podemos ejecutar:

```
sudo apt-get dist-upgrade
```

Si deseamos instalar una o varias aplicaciones determinadas, tan sólo tendremos que ejecutar lo siguiente, sustituyendo "paquete"por el listado de paquetes a instalar:

```
sudo apt-get install paquete ...
```

Recuerda que tabulando al escribir el nombre de los paquetes podemos servirnos del autocompletado para comprobar qué paquete nos queremos instalar. Otra solución es buscar en la lista de paquetes disponibles en caso de no saber el nombre del paquete:

```
sudo apt-cache search paquete
```

9.2.2. Instalación sin conexión a internet

Primero debes asegurarte de que esté instalado *dpkg*. En caso contrario debes descargarte e instalar el paquete *dpkg-dev*. A continuación creamos la carpeta donde vamos a ubicar el repositorio, por ejemplo:

```
mkdir ~/repo
```

Copiamos dentro los paquetes *.deb* que queremos instalar y creamos el script que hará que puedan ser leídos por *apt*:

```
cd /bin/
```

```
sudo nano autorepo
```

El archivo debe contener lo siguiente:

ARCHIVO: /bin/autorepo

```
#!/bin/bash
sudo dpkg-scanpackages repo /dev/null | gzip -9c> repo/Packages.gz
sudo dpkg-scansources repo /dev/null | gzip -9c> repo/Sources.gz
```

Una vez guardado el contenido del archivo, cambiamos los permisos para que pueda ser ejecutado:

```
sudo chmod +x autorepo
```

Ejecutamos el *script* desde el directorio padre del directorio que hará las veces de repositorio:

```
cd
```

```
autorepo
```

Debe aparecer un listado con el contenido de la carpeta del repositorio. Comprueba que se han creado dos paquetes dentro: *Packages.gz* y *Sources.gz*.

Ahora editamos el archivo *sources.list* para agregar nuestro repositorio:

```
sudo nano /etc/apt/sources.list
```

ARCHIVO: /etc/apt/sources.list

```
# Repositorio local: ~/repo
deb file:///home/usuario/ repo/
```

Ojo. El espacio que hay entre */home/usuario* y *repo/* no es un error. Debe ser así.

Para terminar, actualizamos los repositorios:

```
sudo apt-get update
```

Y ya podemos instalar los paquetes.

```
sudo apt-get install paquete...
```

9.3. Instalación de paquetes RPM

Para seguir uno de los procesos siguientes necesitaremos tener descargado el paquete a instalar.

9.3.1. Utilizando RPM

Simplemente necesitaremos ejecutar como usuario *root* la orden **rpm** más la opción *-i* o *–install* seguido de la ruta exacta del paquete:

```
sudo rpm –i /ruta_del_paquete/paquete.rpm
```

Así no aparecerá nada en la terminal. Si añadimos más opciones:

```
sudo rpm –ivh paquete–rpm
```

podremos ver con más detalle el proceso.

9.3.2. Utilizando YUM

Tendremos que ejecutar como *root* la orden **yum** más la opción de instalación local y la ruta exacta donde se encuentre el paquete:

```
yum –nogpgcheck localinstall /ruta_del_paquete/paquete.rpm
```

Si se trata de un paquete no firmado con una llave GPG o ésta no se encuentra registrada en nuestro sistema, necesitaremos también agregar la variable *–nogpgcheck* la cual indica que no se revise si se encuentra firmada o no, o si se encuentra la llave GPG en nuestro sistema. Si no se agrega esta opción el paquete **no se podrá instalar**.

Nota. La **ventaja** de utilizar *yum* en lugar de *rpm* es que si existen dependencias necesarias para instalar el paquete, automáticamente las buscará en los repositorios y las instalará junto con el paquete. Además éste será registrado en la base de datos de YUM.

9.4. Instalación de código fuente

En caso de que no se encuentren disponibles o no podamos instalar la aplicación mediante ninguno de los procesos anteriores, deberemos descargarnos el código fuente. Una vez descargado y descomprimido deberemos acceder a la carpeta donde se encuentren los ficheros de compilación y ejecutar:

```
./configure
make
make install
```

Ojo: Este no es el proceso adecuado, son preferibles los anteriores. Sólo debemos seguir este método si no existe otra alternativa.

Parte IV

Nivel unixero profesional

Capítulo 10

Arranque y parada del sistema

Contiene explicaciones de los niveles de la forma de comportarse de estos sistemas en su inicio y su apagado, así como los niveles las formas en que puede hacerse.	
TIEMPO:	4 horas
NIVEL:	Unixero profesional (página 81)
INFORMACIÓN EXTRAÍDA DE:	Proceso de arranque en Wikipedia ✒ *Arranque del sistema: Administración de linux. Una guía básica.* - Pedro Pablo Fábrega (Creative Commons) ✒ Guía de Instalación de Debian GNU/Linux - Capítulo 5: Arranque del sistema de instalación (LGPL)
PRERREQUISITOS:	Conceptos básicos ✒ Órdenes avanzadas ✒ Kernel ✒ Estructura

Este artículo consta de las siguientes secciones:

Unix es un sistema operativo complejo, y tanto arrancar como detener el sistema es más complicado que emplear un interruptor. Una de las funciones del administrador del sistema es la de arrancarlo y detenerlo de forma correcta. Es importante que el administrador comprenda bien este proceso, debido a que durante el arranque el sistema es especialmente vulnerable.

Cuando se necesita apagar la máquina que está ejecutando un sistema Unix es necesario detener el sistema. Además hay varias razones por las que el administrador necesita detenerlo y arrancar en modo monousuario, para tener el máximo control sobre él, o bien, que sólo estén presentes algunas características del sistema. Entre estas razones se encuentran:

- ✦ Añadir un nuevo hardware a la configuración del sistema.
- ✦ Actualizar algunas aplicaciones.
- ✦ Posibilidad de corrupción en el sistema de ficheros.
- ✦ Sospecha de un fallo hardware.
- ✦ Realizar copias de seguridad del sistema u otras tareas administrativas.

10.1. Niveles de ejecución

Los sistemas Unix tienen varios modos diferentes de operación llamados estados del sistema o niveles de ejecución. Estos niveles hacen posible al administrador limitar la actividad en el sistema cuando efectúe ciertas tareas administrativas.

Un nivel de ejecución se puede definir como un método software para configurar el sistema, indicando qué procesos ejecutar, modo monousuario o multiusuario, dispositivos, etc.

En Unix existen siete niveles de ejecución numerados del 0 al 6. Estos son:

Nivel	***Función***	***Descripción***
0	Parada	Finaliza servicios y programas activos, así como desmonta sistemas de archivos activos y para CPU.
1	Monousuario	Finaliza la mayoría de servicios, permitiendo sólo la entrada del administrador (*root*). Se usa para las tareas de mantenimiento y corrección de errores críticos.
2	Multiusuario sin red	No se inician servicios de red, permitiend sólo entradas locales en el sistema.
3	Multiusuario	Inicia todos los servicios excepto los gráficos asociados a X Window.
4	Multiusuario	Suele usarse para realizar pruebas.
5	Multiusuario X	Igual que el nivel 3, pero con soporte X para la entrada de usuarios (login gráfico).
6	Reinicio	Para todos los programas y servicios. Reinicia el sistema.

Cuadro 10.1: Niveles de ejecución

El nivel 1 se utiliza para las tareas administrativas, pues sólo se permitirá la entrada al *root*. Los niveles 2 al 5, permitirán diferenciar distintas configuraciones del sistema, permitiendo iniciar X Window, dispositivos hardware, etc. Por este motivo, la configuración de los niveles de ejecución es una tarea importante para el administrador del sistema.

10.2. Proceso de arranque del sistema

El proceso de iniciar el sistema desde un estado de parada o apagado se denomina *bootstrapping* o *booting*. Durante esta fase se carga en memoria una pequeña parte de código denominada bootstrap loader, y se inicia el sistema operativo. Este código puede estar almacenado en un disquete o bien en el disco duro, en el MBR.

Si arrancamos desde el disco duro, el código en el MBR examina la tabla de particiones, identifica la partición activa, lee el sector de arranque de la partición, y ejecuta el código almacenado en ese sector de arranque.

Este código se encarga de leer el núcleo del sistema operativo del disco e inicializarlo.

Una vez cargado el núcleo en memoria se realiza una serie de pasos:

- ✦ Reconocer el hardware.
- ✦ Montar los sistemas de ficheros indicados en el fichero */etc/fstab*, empezando por el raíz.
- ✦ Lanzar el proceso **init** que realizará las acciones asociadas al nivel de ejecución escogido.

10.2.1. El proceso init

El proceso **init** es el primero que se ejecuta cuando se produce el arranque del sistema, encargándose de inicializar el sistema, creando y controlando el resto de los procesos. De este modo, el proceso **init** se convierte en el padre de todos los procesos.

Durante el arranque del sistema, el proceso **init** examina el fichero */etc/inittab* en busca de un nivel de ejecución por omisión (initdefault). Si no hay ninguno presente, lo pregunta. El proceso init utiliza el fichero */etc/inittab* para determinar cómo será inicializado el sistema, ejecutando las órdenes indicadas en este fichero.

El proceso **init** se ejecuta en segundo plano, y se queda esperando la terminación de todos los procesos que se creen en el sistema, o bien recibir una señal de error, o a que se cambie el nivel de ejecución mediante las órdenes init, telinit, halt shutdown.

Cuando se le indica un cambio de nivel de ejecución, **init** envía una señal de terminación a todos sus procesos hijos que no deban ejecutarse en el nuevo nivel. A continuación se inicia el nuevo nivel de ejecución lanzando los procesos indicados en el fichero */etc/inittab*.

10.2.2. Configuración de los niveles de ejecución

La existencia de diferentes niveles de ejecución permite configurar el sistema según nuestras necesidades, ejecutando los procesos necesarios. Su objetivo es activar o desactivar los distintos servicios del sistema.

El fichero */etc/inittab* controla el proceso de inicialización del sistema, proporcionando al proceso **init** las instrucciones para crear y ejecutar los procesos asociados a los distintos niveles de ejecución. Para ello, init busca en este fichero las entradas correspondientes al nivel de ejecución.

Cada línea del fichero posee cuatro campos separados por el carácter ':', como se muestra a continuación:

identificador:nivel:acción:orden

- identificador: Cadena de 1-4 caracteres que identifica la línea.
- nivel: Especifica los niveles de ejecución en los que la línea va a ser procesada.
  - **0** Parada
  - **1** Monousuario
  - **2345** Multiusuario
  - **6** Rearranque
  - **sS** Monousuario
  - **aAbBcC** Otros
- acción: Especifica la acción a realizar.
- orden: Especifica el proceso que se va a ejecutar.

ARCHIVO: /etc/inittab

```
# El nivel de ejecución por omisión
id:2:initdefault:
# Fichero de inicialización del sistema
# en tiempo de arranque.
si::sysinit:/etc/init.d/rcS
# Qué realizar en modo monousuario
:S:wait:/sbin/sulogin
# Indica los procesos a ejecutar en los distintos
# niveles de ejecución
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Acción a realizar cuando se pulsa CTRL-ALT-DEL
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Acción a realizar cuando se pulsa ALT-Cursor
kb::kbrequest:/bin/echo "Keyboard Request edit
```

```
/etc/inittab to let this work."
# Qué hacer cuando la fuente de energía del
# sistema falla o vuelve.
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
# Iniciar las consolas virtuales
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
# Iniciar las terminales en el puerto serie
T1:23:respawn:/sbin/getty -L ttyS1 19200 vt100
# Iniciar un módem
T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Asociados al fichero */etc/inittab* existen una serie de directorios que sirven para configurar los distintos niveles. El directorio */etc/init.d* contiene los distintos scripts que se ejecutan cuando se cambia de nivel de ejecución. Estos ficheros son referenciados por enlaces simbólicos en los directorios */etc/rcN.d*, siendo N el nivel de ejecución. Por tanto, cada nivel posee su directorio */etc/rcN.d*.

Cuando se inicializa el sistema en un nivel determinado, se ejecutan todos los ficheros que existen en el directorio corespondiente. Los nombres de estos Sxxnombre o Kxxnombre, donde xx son dos números ficheros son de la forma que indica el orden en que se ejecuta, nombre es el nombre del script que se ejecuta, S indica que se inicia un servicio y K que se finaliza. Cuando se cambia de nivel, se ejecutarán primero los scripts que comienzan por K, seguido por los que comienzan por S.

Además, existe un directorio especial, */etc/rc.boot*, que contiene los scripts que se ejecutan cuando la máquina arranca. Se utiliza principalmente para la inicialización de dispositivos hardware, etc.

10.2.3. Cambio de nivel de ejecución

La existencia de distintos niveles de ejecución permite que el sistema, durante su funcionamiento, pueda pasar de uno a otro según nuestras necesidades.

Durante las transiciones al modo monousuario, los sistemas de ficheros no son desmontados. Sólo se desmontan durante la transición al estado de parada del sistema.

Acciones a realizar sobre un proceso en el nivel de ejecución:

- **respawn**: Reiniciar el proceso cuando termine la ejecución. Es la acción a realizar para las consolas virtuales (getty).
- **wait**:El proceso se ejecuta una vez empezado a ejecutar el nivel de ejecución, y el proceso *init* esperará su terminación.

- **once**: El proceso se ejecutará una única vez.
- **boot**: El proceso se ejecutará durante el arranque del sistema. En este caso, el campo nivel será ignorado.
- **bootwait**: El proceso se ejecutará durante el arranque del sistema, mientras *init* espera su terminación.
- **off**: No realiza nada.
- **ondemand**: Se ejecutará sólo cuando se entre en ese nivel, pero no cuando ocurra un cambio de nivel.
- **initdefault**: Se especifica el nivel de ejecución por omisión en el arranque del sistema. Si no existe, pedirá el nivel de arranque.
- **sysinit**: El proceso se ejecutará durante el arranque del sistema.
- **ctrlaltdel**: Se ejecutará cuando el usuario pulse la combinación de teclas Ctrl + Alt + Del.
- **powerwait**: Se ejecutará cuando reciba una senal que indica un fallo en la fuente de energía. init esperará la terminación del proceso para continuar.
- **powerfail**: Igual al anterior, pero *init* no esperará la terminación.
- **powerokwait**: Se ejecutará cuando la fuente de energía esté de nuevo funcionando.

La orden shutdown

La orden **shutdown** se utiliza para detener, rearrancar o pasar a modo monousuario el sistema. Sólo el superusuario puede ejecutarla.

La sintaxis es:

```
shutdown [opciones] tiempo [mensaje]
```

donde *tiempo* especifica cuando se va a producir el cambio. Se puede usar la palabra *now* para un cambio inmediato, o especificar un tiempo futuro en uno de los siguientes formatos:

- **+n**: Especifica los minutos.
- **hh:mm**: Especifica el tiempo absoluto.

El argumento mensaje se le enviará a todos los usuarios conectados al sistema notificándoles, a intervalos de tiempo cada vez menores, que se va a cambiar de nivel.

Cuando se usa sin opciones, la orden **shutdown** pasa el sistema a modo monousuario. La opción -r rearranca el sistema y la opción -h detiene el sistema.

Ejemplo:

```
shutdown +10 .El sistema se detendrá para rutinas de mantenimiento"
```

La orden halt

La orden **halt** puede ser usada para detener el sistema. Esta orden sólo puede ser utilizada por el superusuario.

La sintaxis es: `halt [opciones]`

La orden halt detiene todos los procesos, sincroniza los discos, registra la parada escribiendo una entrada en el fichero de informes (*/var/log/wtemp*) del sistema, desmonta los sistemas de ficheros y luego para el procesador.

La orden reboot

La orden **reboot** detiene todos los procesos, sincroniza los discos, registra el rearranque escribiendo una entrada en el fichero de informes del sistema (*/var/log/wtemp*). Sin embargo, no proporciona un tiempo de espera ni informa a los usuarios sobre la parada y el arranque del sistema. Por tanto, esta orden no debería ser utilizada si hay usuarios conectados. En ese caso, se debe utilizar la orden shutdown -r.

Se debe tener privilegios de superusuario para utilizar esta orden.

La sintaxis es: `reboot [opciones]`

Nota: Si se hacen cambios en el software del sistema o en ficheros de configuración que se ejecutan cuando éste se inicia, debemos rearrancarlo para que estos cambios tengan efecto.

La orden init

La orden **init** permite realizar cualquier transición entre niveles de ejecución.

Esta orden sólo la puede ejecutar el superusuario.

La sintaxis es:

```
init nivel
```

donde *nivel* especifica el nuevo nivel que se quiere alcanzar:

- **0**: Detiene el sistema.
- **1**: Cambia al modo monousuario.
- **2345**: Cambia al modo multiusuario.
- **6**: Rearranca el sistema.
- **sS**: Cambia al modo monousuario.
- **qQ**: Examina las entradas en /etc/inittab.

- **aAbBcC**: Procesa en el fichero /etc/inittab las entradas que correspondana los niveles de ejecución a, A, b, B, c o C.

La orden telinit

La orden **telinit** está asociada a la orden **init** para especificarle el cambio de un nivel de ejecución. Por tanto, presenta el mismo formato que ella, pues esos parámetros se le pasan a init para realizar la acción apropiada.

Esta orden sólo puede ser utilizada por el superusuario, o bien por ciertos usuarios con los privilegios adecuados.

La sintaxis es: `telinit nivel`

Entrada al modo multiusuario

Una vez que el proceso init ejecuta las acciones indicadas en el directorio */etc/rcN.d* relacionadas con el nivel de ejecución que se quiere alcanzar, si éste es el modo multiusuario (niveles 2 a 5), inicia las consolas virtuales indicadas en el fichero */etc/inittab*, ejecutando la orden getty.

La orden getty muestra el mensaje que aparece en el fichero */etc/issue* y a continuación ejecuta la orden login que se encarga de pedir el nombre y la contraseña al usuario para permitirle la entrada al sistema. Para ello, comprueba en el fichero */etc/passwd* la validez de los datos introducidos. Si son válidos, ejecuta el shell de entrada del usuario. Si no son válidos termina su ejecución.

init es el proceso padre de todos, creando nuevos procesos vía fork y después ejecutando el programa adecuado con exec, incluso los programas getty y login.

Una vez finalizada la ejecución del proceso de usuario se avisa al proceso init para que vuelva a reiniciar la consola virtual que permita la entrada de un nuevo usuario, esto se consigue especificando la acción respawn en la línea correspondiente a la orden getty en el fichero */etc/inittab*.

10.2.4. Proceso de arranque

Al encender la computadora las primeras operaciones las realiza la *BIOS* (*Sistema Básico de Entrada/Salida* o *Basic Input-Output System*). En esta etapa se realizan operaciones básicas de hardware. El proceso de arranque será diferente dependiendo de la arquitectura del procesador y la *BIOS*.

Una vez que el hardware se reconoce y se deja listo, la *BIOS* carga en memoria el código ejecutable del **gestor de arranque** y le pasa el control. Hay variedad de *BIOS* que permiten al usuario definir en qué dispositivo/partición se encuentra dicho gestor.

10.2.5. Gestores de arranque

Un cargador de arranque (*boot loader* en inglés) es un programa diseñado exclusivamente para cargar un sistema operativo en memoria. La etapa del cargador de arranque es diferente de una plataforma a otra.

Como en la mayoría de arquitecturas este programa se encuentra en el MBR, el cual es de 512 bytes, no es suficiente para cargar en totalidad un sistema operativo. Por eso, el cargador de arranque consta de varias etapas.

Para las plataformas *x86*, el *BIOS* carga la primera etapa del cargador de arranque (típicamente una parte de LILO o GRUB). El código de esta primera etapa se encuentra en el sector de arranque (o *MBR*). La primera etapa del cargador de arranque carga el resto del cargador de arranque.

Los cargadores de arranque modernos típicamente preguntan al usuario cual sistema operativo (o tipo de sesión) desea inicializar.

GRUB

GRUB se carga y se ejecuta en 4 etapas:

I. La primera etapa del cargador la lee el *BIOS* desde el *MBR* (*master boot record*, primer sector de arranque).

II. La primera etapa carga el resto del cargador (segunda etapa). Si la segunda etapa está en un dispositivo grande, se carga una etapa intermedia (llamada etapa 1.5), la cual contiene código extra que permite leer cilindros mayores que 1024 o dispositivos tipo LBA (*logical block addressing* o *dirección lógica de bloques*).

III. La segunda etapa ejecuta el cargador y muestra el menú de inicio de *GRUB*. Aquí se permite elegir un sistema operativo junto con parámetros del sistema.

IV. Cuando se elige un sistema operativo, se carga en memoria y se pasa el control.

GRUB soporta métodos de arranque directo, arranque *chain-loading*, *LBA*, *ext2* y hasta ün pre-sistema operativo totalmente basado en comandos". Tiene tres interfaces: un menú de selección, un editor de configuración y una consola de línea de comandos.

Dado que *GRUB* entiende los sistemas de archivos *ext2* y *ext3* y además provee una interfaz de línea de comandos, es más fácil rectificar o modificar cuando se malconfigura o se corrompe. La nueva versión 2 de *GRUB*, soporta sistema de archivos *ext4*.

LILO

LILO es más antiguo. Es casi idéntico a *GRUB* en su proceso, excepto que no contiene una interfaz de línea de comandos. Por lo tanto todos los cambios en su configuración deben ser escritos en el *MBR*,

y reiniciar el sistema. Un error en la configuración puede arruinar el proceso de arranque a tal grado de que sea necesario usar otro dispositivo que contenga un programa que sea capaz de arreglar ese defecto.

De forma adicional, *LILO* no entiende sistema de archivos, por lo tanto no hay archivos y todo se almacena en el *MBR* directamente.

Cuando el usuario selecciona una opción del menú de carga de *LILO*, dependiendo de la respuesta, carga los 512 bytes del *MBR* para otros sistemas, o la imagen del *kernel Linux*.

10.3. Ejercicios

ASPECTOS PEDAGÓGICOS: Permiten asimilar el comportamiento del sistema al iniciarse y cerrarse. También sirven para distinguir las características de los niveles de arraque.
PREPARACIÓN PREVIA: Para hacer estos ejercicios debes acceder al sistema con el usuario por defecto *usuario* (recuerda que la contraseña es *unix*). A continuación:

- Ve al directorio casa del usuario actual. Comprueba que existe un directorio llamado **unix** y dentro de éste un subdirectorio llamado **arranque**. Para ello ejecuta:
  `ls unix`

  Debe aparecer el directorio **arranque**. En caso de que no exista alguno de estos elementos, puedes hacer varias cosas:

  1. Descargar el archivo comprimido con la jerarquía de elementos de comprobación que se encuentra en la portada del wiki y descomprimirlo en el directorio casa del usuario actual.
  2. Apagar la máquina virtual y restaurar el *snapshot* que funcionaba bien.
  3. Apagar la máquina virtual y borrarla eliminando el disco duro virtual. A continuación, descargar el disco duro y volver a configurar la máquina de nuevo.

- Ve al subdirectorio **unix/arranque**. Limpia el sistema antes de realizar los demás ejercicios para prevenir posibles incongruencias:
  `sudo python arranque.py -c`
  o
  `sudo python arranque.py -clean`

1. Compruebe cuál es el modo de arranque por defecto de su sistema.

2. Deseamos configurar el nivel de ejecución 4 para pruebas del sistema. Por eso deseamos que en dicho nivel:

   - No se ejecuten los servicios “cups” ni “pcmciautils”
   - El servicio “anacron” se ejecute inmediatamente después de la ejecución de “atd” (no modifique “atd” para ello, sólo “anacron”)

3. Vuelve al directorio inicial (**arranque**) y ejecuta el primer control intermedio:
   `sudo python arranque.py -p 1`
   o
   `sudo python arranque.py -partial 1`

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Manejo y modificación de programas a ejecutar en los distintos niveles de ejecución.	Modificación del nivel por defecto.

4. Establezca el nivel de ejecución 4 como modo por defecto de su sistema.

5. Asegúrese de que al pulsar *Ctrl+Alt+Del* en dicho nivel no se reinicie el sistema, sino que se almacena la hora de pulsación en el fichero */var/log/ctrl.log* (cree este fichero si no existe y asegúrese de que sólo puede leerlo o modificarlo el *root*).

Tras la realización de estos ejercicios debes tener clara la utilización y el manejo de las siguientes órdenes:

indeterminado

Si aún sigues teniendo dudas puedes:

- Releer el artículo **Arranque y parada del sistema** de nuevo al completo, o tan sólo las secciones en las que tengas dudas.
- Volver a desarrollar los ejercicios de nuevo.
- Revisar los artículos creados en el wiki para cada una de las órdenes.
- Revisar el manual de las órdenes implicadas (recuerda que tienes los enlaces en los artículos del wiki de cada una y además en está el manual del sistema).

Capítulo 11

Usuarios y grupos

Administración y control de los usuarios y los grupos dentro de un sistema Unix	
TIEMPO:	4-5 horas
NIVEL:	Unixero profesional (página 81)
INFORMACIÓN EXTRAÍDA DE:	*Guadalinex 2004. Manual sobre Guadalinex Ciudadano versión.* Capítulo 11: Linux avanzado - Daniel López Avellaneda (GFDL)
PRERREQUISITOS:	Soporte para pruebas ✒ Órdenes avanzadas

Este artículo consta de las siguientes secciones:

11.1. Introducción

Puesto que Unix es un sistema multiusuario, permite que varios usuarios usen el sistema simultáneamente. Se hace necesario que cada usuario no pueda acceder a los documentos de los demás, además no todos los usuarios deberían poder instalar programas, modificar ficheros importantes del sistema u otras cosas importantes.

Por ello se establece nombre y contraseña para cada usuario. Además existe el administrador del sistema o superusuario, cuyo login es root, que tiene acceso a todo y permisos para todo.

Las contraseñas son almacenadas de forma cifrada, por lo que es imposible acceder a ellas (si las guardara simplemente en un fichero de texto, sería muy fácil acceder y romper el sistema de seguridad).

Dentro de las cuentas asociadas a usuarios podemos encontrar diferentes tipos:

- Cuenta de administrador, con identificador *root*: sólo es (o debería ser) utilizada para las operaciones de administración. El usuario root es el que dispone demás permisos y acceso completo a la máquina y a los archivos de configuración. Por lo tanto, también es el que más daño puede causar por errores u omisiones. Es mejor evitar usar la cuenta de *root* como si fuese un usuario más, por lo que se recomienda dejarla sólo para operaciones de administración.
- Cuentas de usuarios: cuentas normales para cualquier usuario de la máquina tienen los permisos restringidos al uso de ficheros de su cuenta, y a algunas otras zonas particulares (por ejemplo, los temporales en */tmp*), así como a utilizar algunos dispositivos para los que se le haya habilitado.
- Cuentas especiales de los servicios: **lp**, **news**, **wheel**, **www-data**... cuentas que no son usadas por personas, sino por servicios internos del sistema, que los usa bajo estos nombres de usuario. Algunos de los servicios también son usados bajo el nombre de root.

11.2. Información de usuarios

Toda la información sobre usuarios, grupos y contraseñas se guarda en los archivos:

- **/etc/passwd**: información sobre usuarios
- **/etc/group**: información sobre grupos
- **/etc/shadow**: contraseñas cifradas
- **/etc/gshadow**: contraseñas cifradas de los grupos, aunque generalmente no se utiliza este fichero

En estos archivos de texto, se almacena la información línea a línea (cada una es un usuario o un grupo) y dentro de cada línea hay varios campos separados por ":".

11.2.1. El fichero /etc/passwd

ARCHIVO: /etc/passwd

usuario : x : UID : GID : comentarios : directorio_home : shell

donde cada uno de los campos se corresponde con:

- **usuario**: identificador de usuario en el sistema.
- **x**: contraseña, pero en su lugar aparece una 'x', ya que la contraseña se encuentra cifrada en */etc/shadow*
- **UID**: número de identidad del usuario. Es un valor único. Suele ser mayor o igual que 100, ya que el 0 se reserva para el *root* y del 1 al 99 para tareas del sistema.
- **GID**: número de identidad de grupo. Al igual que ocurre con el UID, el cero se reserva para el grupo *root*.
- **comentarios**: nombre real u otros comentarios sobre el usuario (puede contener espacios).
- **directorio_home**: directorio de casa del usuario
- **shell**: intérprete de comandos (normalmente bash). En caso de que se encuentre en este campo */bin/false*, el usuario no podrá ejecutar ningún comando del sistema.

Nota: Si aparecen dos símbolos de dos puntos seguidos (::) significa que el campo está vacío.

CURIOSIDAD

El fichero */etc/passwd* solía contener las palabras de paso de los usuarios de forma encriptada, pero el problema estaba en que cualquier usuario podía ver el fichero, y en su momento se diseñaron cracks que intentaban encontrar en forma bruta la palabra de paso, mediante la palabra de paso encriptada como punto de partida (palabra codificada con el sistema crypt).

Para evitar esto, hoy en día ya no se colocan las palabras de paso en este archivo, sólo una "x"que indica que se encuentran en otro fichero, que es sólo de lectura para el usuario root, */etc/shadow*, que veremos a continuación.

11.2.2. El fichero /etc/group

ARCHIVO: /etc/group

grupo : x : GID : lista_usuarios

donde cada uno de los campos se corresponde con:

- **grupo**: identificador de grupo en el sistema.
- **x**: contraseña, aunque lo más común es que este campo aparezca vacío (el grupo no necesita contraseña).
- **GID**: número de identidad de grupo.
- **lista_usuarios**: lista de usuarios que pertenecen al grupo separados por comas. También se pueden añadir usuarios de otros grupos, con lo que tendrían todos los privilegios de este grupo.

Para añadir un nuevo usuario a un grupo, basta con agregarlo en la lista de usuarios (sin olvidar poner la coma de separación entres usuarios).

La lista de usuarios del grupo puede estar presente o no, ya que la información ya está en */etc/passwd*, no suele ponerse en */etc/group*.

11.2.3. El fichero /etc/shadow

ARCHIVO: /etc/shadow

usuario : contraseña_cifrada : d1 : d2 : d3 : d4 : d5 : d6 : reservado

donde cada uno de los campos es:

- **usuario**: es el login o nombre de usuario (el mismo que en */etc/passwd*)
- **x**: contraseña: aparece una x; la contraseña se encuentra cifrada en */etc/shadow*.
- **d1**: nº de días desde el 01/01/1970 hasta último cambio de la contraseña.
- **d2**: nº de días que deben pasar hasta que se pueda cambiar la contraseña.
- **d3**: nº de días que deben pasar para que caduque la contraseña y deba ser cambiada.
- **d4**: nº de días de antelación con los que avisará el sistema de la caducidad de la contraseña.
- **d5**: nº de días con contraseña caducada antes de deshabilitar la cuenta.
- **d6**: nº de días desde el 01/01/1970 y el día en que se deshabilitó la cuenta.

Además, las claves de encriptación pueden ser más difíciles, ya que ahora puede utilizarse un sistema denominado md5 (suele aparecer como opción a la hora de instalar el sistema) para proteger las palabras de paso de los usuarios.

Veremos más detalles al respecto en la unidad dedicada a la seguridad.

11.2.4. El fichero /etc/gshadow

Al igual que el fichero */etc/shadow* de las contraseñas encriptadas para usuarios, también se puede usar un fichero */etc/gshadow* de contraseñas encriptadas para grupos.

Se suele usar para permitir el acceso al grupo, a un usuario que no es miembro del grupo. Ese usuario tendría entonces los mismos privilegios que los miembros de su nuevo grupo.

11.2.5. Otros ficheros

Otros ficheros interesantes son los del directorio */etc/skel*, donde se hallan los ficheros que se incluyen en cada cuenta de usuario al crearla.

Podemos tener unos scripts de configuración que se ejecutaban al entrar o salir de la cuenta. En el directorio *skel* se guardan los .esqueletos"que se copian en cada usuario al crearlo. Suele ser responsabilidad del administrador crear unos ficheros adecuados para los usuarios, poniendo los *path* necesarios de ejecución, inicialización de variables de sistema, variables que se necesiten para el software, etc.

11.3. Gestión de Usuarios y Grupos

A continuación vamos a ver una serie de órdenes útiles para esta gestión:

11.3.1. useradd

La orden **useradd** añade un usuario al sistema.

La sintaxis es:

```
useradd [opciones] <usuario>
```

Algunas de las opciones más útiles son:

- **-c <comentario>**: Incluir comentario (normalmente nombre completo del usuario).
- **-d <directorio_casa>**: Permite especificar su directorio de inicio.
- **-m**: Crea directorio de inicio */home/usuario* y le pone todos los subdirectorios y ficheros que haya en */etc/skel*.

El nuevo usuario creado no tiene contraseña. Para asignarle una contraseña, tecleamos desde terminal:

```
passwd <usuario><contraseña>
```

Nota: Cualquier usuario puede cambiar su contraseña tecleando *passwd*.

11.3.2. userdel

La orden **userdel** borra la cuenta de un usuario.

La sintaxis es:

`userdel [-r] <usuario>`siendo:

- **-r**: Borra también su directorio home.

Si usamos la opción **-r** borrará el directorio casa del usuario, pero no borra el resto de ficheros que tenga en otras carpetas. Estos ficheros tendrán como dueño un usuario inexistente, por lo que podría crear conflictos en el sistema. Por ello resulta aconsejable deshabilitar una cuenta en lugar de eliminarla. Otra opción es usar la orden *deluser* (que permite incluso hacer copia de seguridad de los ficheros del usuario).

11.3.3. usermod

La orden **usermod** modifica las opciones de un usuario.

La sintaxis es:

`usermod [opciones] <usuario>`

Algunas de las opciones más útiles son:

- **-L**: Lock. Bloquea la contraseña (deshabilita la cuenta).
- **-U <directorio_casa>**: Unlock. Desbloquea la contraseña.

11.3.4. groupadd

La orden **groupadd** añade un grupo al sistema.

La sintaxis es:

`groupadd <nombre_grupo>`

11.3.5. groupmod

La orden **groupmod** modifica un grupo.

La sintaxis es:

`groupmod [-n] <nombre_grupo>`

donde:

- **-n <nuevo_nombre>**: Cambia el nombre del grupo

11.3.6. groupdel

<div class="plainlinks"style=çlear: both;float: right; border: 5px groove yellowgreen; margin-left: 5px; padding-left: 7px; padding-right: 7px; background-color: #CCFF99;» Manual: groupdel La orden **groupdel** borra un grupo del sistema.

La sintaxis es:

`groupdel <nombre_grupo>` **Nota:** Si el grupo es primario para algún usuario no se puede borrar.

11.3.7. Otras órdenes

- **chage**: cambia información sobre la caducidad de la contraseña
- **chfn**: cambia la información del campo çomentario"de un usuario, presente en el comentario del fichero */etc/passwd*.
- **chsh**: cambia el shell de inicio del usuario.
- **chpasswd**: cambiar las contraseñas a un gran número de usuarios.
- **whoami**: nos dice qué usuario somos
- **groups**: nos dice en qué grupos estamos
- **id**: nos muestra ID y grupos
- **login**: nos permite cambiar de usuario
- **su**: Cambio de identidad.
- **newusers**: permite crear varias cuentas desde la información introducida en un fichero.
- **newgrp**: nos permite cambiar a otro grupo (necesitamos saber la contraseña)
- **passwd**: cambia la contraseña de un usuario. Puede ejecutarse como usuario, y entonces pide la contraseña antigua y la nueva. En el caso de hacerlo, el *root* tiene que especificar el usuario al que va a cambiar la contraseña (si no, estaría cambiando la suya) y no necesita la contraseña antigua.
- **sg**: nos permite ejecutar comandos de otro grupo
- **who**: lista los usuarios que hay en el sistema
- **w**: lista los usuarios que hay en el sistema y lo que están haciendo
- **write**: escribir mensaje a otro usuarios
- **wall**: escribir mensaje a todos los usuarios

- **mesg**: permitir o no que te escriban mensajes

CURIOSIDAD

Respecto a la administración de usuarios y grupos, lo que hemos comentado aquí hace referencia a la administración local de una sola máquina. En sistemas con múltiples máquinas que comparten los usuarios suele utilizarse otro sistema de gestión de la información de los usuarios. Estos sistemas, denominados genéricamente sistemas de información de red, como NIS, NIS+ o LDAP, utilizan bases de datos para almacenar la información de los usuarios y grupos, de manera que se utilizan máquinas servidoras, donde se almacena la base de datos, y otras máquinas clientes, donde se consulta esta información.

Esto permite tener una sola copia de los datos de los usuarios (o varias sincronizadas), y que éstos puedan entrar en cualquier máquina disponible del conjunto administrado con estos sistemas. Además estos sistemas incorporan conceptos adicionales de jerarquías, y/o dominios/zonas de máquinas y recursos, que permiten representar adecuadamente los recursos y su uso en organizaciones con diferentes estructuras de organización de su propio personal y sus secciones internas.

Podemos comprobar si estamos en un entorno de tipo NIS si en las líneas *passwd* y *group* del archivo de configuración */etc/nsswitch.conf* aparece *compat*, si estamos trabajando con los ficheros locales, o bien *nis* o *nisplus* según el sistema con que estemos trabajando. En general, para el usuario simple no supone ninguna modificación, ya que la gestión de las máquinas le es transparente, y más si se combina con ficheros compartidos por NFS que permite disponer de su cuenta sin importar con qué máquina trabaje. La mayor parte de los comandos anteriores pueden seguir usándose sin problema bajo NIS o NIS+, son equivalentes a excepción del cambio de contraseña, que en lugar de *passwd*, se suele hacer con *yppasswd* (NIS) o *nispasswd* (NIS+); aunque suele ser habitual que el administrador los renombre (por un enlace) a *passwd*, con lo cual los usuarios no notarán la diferencia.

11.4. Utilidades prácticas

11.5. Ejercicios

ASPECTOS PEDAGÓGICOS: Permiten el aprendizaje de órdenes relacionadas con la gestión y administración tanto de usuarios como de grupos. También sirve para asimilar el comportamiento habitual de un adminitrador a la hora de cambiar su rol y comprobar quién está trabajando en el sistema.
PREPARACIÓN PREVIA: Para hacer estos ejercicios debes acceder al sistema con el usuario por defecto *usuario* (recuerda que la contraseña es *unix*). A continuación:

- Ve al directorio casa del usuario actual. Comprueba que existe un directorio llamado **unix** y dentro de éste un subdirectorio llamado **usuarios**. Para ello ejecuta:
  `ls unix`

  Debe aparecer el directorio **usuarios**. En caso de que no exista alguno de estos elementos, puedes hacer varias cosas:

1. Descargar el archivo comprimido con la jerarquía de elementos de comprobación que se encuentra en la portada del wiki y descomprimirlo en el directorio casa del usuario actual.
2. Apagar la máquina virtual y restaurar el *snapshot* que funcionaba bien.
3. Apagar la máquina virtual y borrarla eliminando el disco duro virtual. A continuación, descargar el disco duro y volver a configurar la máquina de nuevo.

✦ Ve al subdirectorio **unix/usuarios**. Limpia el sistema antes de realizar los demás ejercicios para prevenir posibles incongruencias:
```
sudo python usuarios.py -c
```
o
```
sudo python usuarios.py -clean
```

1. Cree un grupo de usuarios denominado “progs”.
2. Cree un usuario llamado “juan” cuyo grupo primario sea “progs”.
3. Cree un grupo de usuarios demonimado “externos”.
4. Cree un usuario llamado “luis” cuyo grupo primario sea “externos” y grupo secundario “progs”.
5. Vuelve al directorio inicial (**usuarios**) y ejecuta el primer control intermedio:
```
sudo python usuarios.py -p 1
```
o
```
sudo python usuarios.py -partial 1
```

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Modificación del rol del usuario y comprobaciones	Trabajo de creación y modificación de nuevos usuarios y grupos

6. Cree un directorio llamado “desarrollo” en /srv (comprueba si existe este directorio en la máquina virtual)
7. Ponga los permisos adecuados para que luis pueda leer, escribir y acceder a él, pero que juan sólo pueda acceder y leer su contenido.
8. Queremos que a partir de ahora todas las cuentas de usuario que se creen en el sistema tengan un directorio llamado “repos”.
9. Vuelve al directorio inicial (**usuarios**) y ejecuta el segundo control intermedio:
```
sudo python usuarios.py -p 2
```
o
```
sudo python usuarios.py -partial 2
```

DEBES HABER APRENDIDO A:	QUEDA POR APRENDER:
Creación y modificación de nuevos usuarios y grupos	Modificación de datos de usuario

10. Asegúrese de que todos los usuarios cambian su contraseña cada 30 días como máximo
11. Cambie la contraseña del usuario luis, ponga como contraseña “SuJJh_4”.

12. Cambie el shell por defecto de juan, que use /bin/zsh.

Tras la realización de estos ejercicios debes tener clara la utilización y el manejo de las siguientes órdenes:

useradd ✒ userdel ✒ usermod ✒ groupadd ✒ groupmod ✒ groupdel ✒ newusers ✒ newgrp ✒ su ✒ who

Si aún sigues teniendo dudas puedes:

- ✦ Releer el artículo **Usuarios y grupos** de nuevo al completo, o tan sólo las secciones en las que tengas dudas.
- ✦ Volver a desarrollar los ejercicios de nuevo.
- ✦ Revisar los artículos creados en el wiki para cada una de las órdenes.
- ✦ Revisar el manual de las órdenes implicadas (recuerda que tienes los enlaces en los artículos del wiki de cada una y además en está el manual del sistema).

Capítulo 12

Administración del sistema de archivos

Esta sección está dedicada a un nivel más complejo de los sistemas de archivos y a la forma de trabajar de un administrador de sistemas Unix respecto a ellos	
TIEMPO:	4 horas
NIVEL:	Unixero profesional (página 81)
INFORMACIÓN EXTRAÍDA DE:	LSB ✒ GAWK Documentation ✒ *Effective AWK Programming* - Arnold Robbins (O'Reilly, 2001, ISBN: 0-596-00070-7) ✒ Sed & Awk - Dale Dougherty and Arnold Robbins (O'Reilly, 1997, ISBN: 1-56592-225-5)
PRERREQUISITOS:	Sistema de archivos

Este artículo consta de las siguientes secciones:

12.1. Linux Standard Base

La **Base Estándar para Linux** (*Linux Standard Base*, abreviado LSB), es un proyecto conjunto de varias Distribuciones de Linux bajo la estructura organizativa del Free Standards Group con el objeto de crear y normalizar la estructura interna de los sistemas operativos derivados de Linux. La LSB está basada en la Especificación POSIX, la Especificación Única de UNIX (Single UNIX Specification) y en varios otros estándares abiertos, aunque extiende éstos en ciertas áreas.

De acuerdo a la definición de la propia LSB:

El objetivo de la LSB es desarrollar y promover un conjunto de estándares que aumentarán la compatibilidad entre las distribuciones de Linux y permitirán que los programas de aplicación puedan

> *ser ejecutados en cualquier sistema que se adhiera a ella. Además, la LSB ayudará a coordinar esfuerzos tendentes a reclutar productores y proveedores de programas que creen productos originales para Linux o adaptaciones de productos existentes.*

Mediante un proceso de certificación es posible obtener la conformidad a la LSB de un producto. Dicha certificación la lleva a cabo el Open Group en colaboración con el Free Standards Group (Grupo de Estándares Libres).

Como ejemplo, la LSB específica: bibliotecas estándar, un conjunto de órdenes y utilerías que extienden el estándar POSIX, la estructura jerárquica del sistema de archivos, los niveles de ejecución, y varias extensiones al sistema gráfico X Window.

12.2. Puntos de montaje

Aparte del sistema de ficheros principal (/) y de sus posibles divisiones en particiones extras (*/usr /var*, */tmp*, */home*), cabe tener en cuenta la posibilidad de dejar puntos de montaje preparados para el montaje de otros sistemas de ficheros, ya sea particiones de disco u otros dispositivos de almacenamiento.

En las máquinas en que GNU/Linux comparte la partición con otros sistemas operativos, mediante algún sistema de arranque (LILO o GRUB), pueden existir varias particiones asignadas a los diferentes operativos. Muchas veces es interesante compartir datos con estos sistemas, ya sea para leer sus ficheros o modificarlos. A diferencia de otros sistemas (que sólo tienen en cuenta sus propios datos y sistemas de ficheros, y en los cuales en algunas versiones no se soportan algunos de sus propios sistemas de ficheros), GNU/Linux es capaz de tratar, como hemos visto, con una cantidad enorme de sistemas de ficheros de diferentes operativos y poder compartir la información.

12.2.1. Explicación

Si en los PC personales hemos instalado GNU/Linux, seguramente encontraremos más de un operativo, por ejemplo, otra versión de GNU/Linux con *ext2* o *ext3* de sistema de ficheros y otros. Por ejemplo, sistema de ficheros FAT, FAT32 (o vfat para Linux) o NTFS (ntfs para Linux).

Nuestro sistema GNU/Linux puede leer datos (o sea ficheros y directorios) de todos estos sistemas de ficheros y escribir en la mayoría de ellos.

En el caso de NTFS, hasta ciertos momentos, existieron problemas en la escritura que se encontraba en forma experimental en la mayoría de *drivers* de *kernel*. Esto se debía principalmente a las diferentes versiones que van apareciendo del sistema de ficheros, ya que existen dos versiones principales llamadas NTFS y NTFS2, y algunas extensiones como los llamados volúmenes dinámicos, o los sistemas de ficheros encriptados. Acceder con según que drivers presentaba ciertas incompatibilidades, que podrían causar corrupciones de datos o errores en el sistema de ficheros.

Gracias a *FUSE* (un módulo integrado en el *kernel* a partir de la versión 2.6.11) se ha permitido un desarrollo más flexible de sistemas de ficheros, directamente en espacio de usuario (de hecho FUSE actúa como un “puente” entre las peticiones del kernel, y el acceso que se hace desde el *driver*).

Gracias a las posibilidades de FUSE, se tiene un soporte más o menos completo de NTFS, en especial desde la aparición del *driver* (basado en FUSE) ntfs-3g (http://www.ntfs-3g.org), y la combinación con las utilidades *ntfsprogs*.

Para que se puedan leer o escribir los datos, la partición tiene que estar disponible dentro de nuestro sistema de ficheros raíz (/). Por lo tanto, hay que llevar a cabo un proceso de "montaje"del sistema de ficheros en algún punto de nuestro árbol de directorios. Se seguirá el mismo proceso si se trata de un dispositivo de almacenamiento.

Dependiendo de la distribución, se usan unos u otros, o también los podemos crear nosotros. Normalmente suelen existir o bien como subdirectorios de la raíz, por ejemplo */cdrom*, */win*, */floppy*, o bien son subdirectorios dentro de */mnt*, el punto estándar de montaje (aparecen como */mnt/cdrom*, */mnt/floppy*...), o el directorio */media* preferido últimamente por las distribuciones. Según el estandard FHS, */mnt* se debería usar para montajes temporales de sistemas de archivo, mientras */media* se utilizaría para montar dispositivos removibles.

El proceso de montaje se realiza mediante la orden **mount** con el siguiente formato:

```
mount -t filesystem-type device mount-point
```

El tipo de sistema de ficheros puede ser:

- msdos (fat)
- vfat (fat32)
- ntfs (ntfs)
- iso9660 (para cdrom)

El dispositivo es la entrada correspondiente en el directorio */dev* a la localización del dispositivo:

- Los IDE tenían */dev/hdxy* donde:
  - **x** es *a* (1 master), *b* (1 slave), *c* (2 master) o d (2 slave)
  - **y** es el número de partición
- Los SCSI son */dev/sdx* donde:
  - x es a,b,c,d ... (según el ID SCSI asociado 0,1,2,3,4 ...).

12.2.2. Ejemplos

- Montar el CD-ROM (si es el IDE que está en el segundo IDE de forma máster) en el punto */mnt/cdrom*.

```
mount -t iso9660 /dev/hdc /mnt/cdrom
```

- Montar el CD-ROM; */dev/cdrom* se usa como sinónimo (es un enlace) del dispositivo donde está conectado:

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

- Montar el disquete */dev/fd0H1440*. Sería la disquetera A en alta densidad (1.44 MB), también puede usarse */dev/fd0*:

```
mount -t vfat /dev/fd0H1440 /mnt/floppy
```

Si estas particiones son más o menos estables en el sistema (o sea, no cambian frecuentemente) y las queremos utilizar, lo mejor será incluir los montajes para que se hagan en tiempo de ejecución, al iniciar el sistema, mediante la configuración del fichero */etc/fstab*:

ARCHIVO: /etc/fstab

```
# /etc/fstab: Información estática del sistema de ficheros
#<Sis. ficheros><Punto montaje><Tipo><Opciones><volcado><pasada>
/dev/hda2 / ext3 errors = remount,ro 0 1
/dev/hdb3 none swap sw 0 0
proc /proc proc defaults 0 0
/dev/fd0 /floppy auto user,noauto 0 0
/dev/cdrom /cdrom iso9660 ro,user,noauto 0 0
/dev/sdb1 /mnt/usb vfat user,noauto 0 0
```

Por ejemplo, esta configuración incluye algunos de los sistemas estándar, como la raíz en */dev/hda2*, la partición de *swap* que está en *hdb3*, el sistema *proc* (que utiliza el *kernel* para guardar su información). Y el disquete, el CD-ROM, y en este caso un disco USB de tipo Flash (que se detecta como un dispositivo SCSI). En algunos casos, se especifica auto como tipo de sistema de ficheros. Esto permite que se autodetecte el sistema de ficheros. Si se conoce, es mejor indicarlo en la configuración, y por otra parte, el *noauto* en las opciones permite que no sea montado de forma automática siempre, sino bajo petición (o acceso).

Si tenemos esta información en el fichero, el proceso de montaje se simplifica mucho, ya que se hará o bien en ejecución, en arranque, o bien bajo demanda (para los *noauto*). Y puede hacerse ahora simplemente pidiendo que se monte el punto de montaje o el dispositivo:

```
mount /mnt/cdrom

mount /dev/fd0
```

dado que el sistema ya tiene el resto de la información.

El proceso contrario, el desmontaje, es bastante sencillo. Tan sólo hay que ejecutar la orden *umount* con punto o dispositivo:

```
umount /mnt/cdrom
```

```
umount /dev/fd0
```

En el caso de medios extraíbles, tipo CD-ROM (u otros), puede usarse *eject* para la extracción del soporte físico:

```
eject /dev/cdrom
```

o, en este caso, sólo:

```
eject
```

Las órdenes *mount* y *umount* montan o desmontan todos los sistemas disponibles. En el fichero */etc/mtab* se mantiene una lista de los sistemas montados en un momento concreto, que se puede consultar utilizando:

```
mount
```

12.3. Utilidades prácticas

Anexo acerca de AWK en la página 119.

Capítulo 13

Instalación de un servidor FTP

En esta sección se expone una iniciación al trabajo con servidores. Se incluye el proceso de instalación de un servidor FTP media vsftpd.	
TIEMPO:	90 minutos
NIVEL:	Unixero profesional (página 81)
INFORMACIÓN EXTRAÍDA DE:	Web oficial de vsftpd
PRERREQUISITOS:	Conceptos básicos ✒ Instalación de software ✒ Órdenes avanzadas

Este artículo consta de las siguientes secciones:

13.1. Introducción

FTP es un protocolo para el envío y recepción de ficheros. Se basa en una estructura cliente-servidor cuyo comportamiento está definido en el RFC 959.

Básicamente consiste en un sistema servidor que atiende peticiones de clientes. Estos, tras identificarse, pueden enviar o recibir ficheros según la configuración del sistema.

13.2. Funcionamiento FTP

13.2.1. Cliente FTP

Existen multitud de clientes FTP libres, como *gftp, wsftp* o incluso el mismo navegador Mozilla Firefox.

Su instalación es sencilla, y se pueden probar para descargar ficheros de servidores FTP de descarga públicos como ftp.cica.es, ftp.rediris.es, ftp.suse.com, etc.

13.2.2. Servidores FTP

Existen muchos servidores FTP para sistemas UNIX: unos destacan por su sencillez para la configuración, otros por determinadas opciones que lo distinguen del resto, etc. Para nuestro ejemplo vamos a usar uno de los que más crecimiento tiene en los últimos tiempos: vsftp (Very Secure FTP).

Como se puede ver en su web oficial, muchos servidores importantes (Debian, Red Hat, etc) confían en *vsftp* por su robustez y facilidad de uso.

13.3. Instalación de vsftp

El servidor *vsftp* viene empaquetado en un sólo paquete de nombre *vsftpd*. Consulte la entrada de instalación de software si necesita instrucciones detalladas para su sistema. Para sistemas basados en Debian (como el usado de soporte para pruebas en el wiki) basta con escribir:

```
sudo apt-get install vsftpd
```

13.4. Puesta en marcha

Por defecto, el sistema se configura para estar activado como servicio por defecto en los modos de funcionamiento 2, 3, 4 y 5. Si desea realizar otra configuración consulte la sección sobre arranque y parada del sistema.

13.4.1. Comprobación

Cuando el servidor esté en marcha puedes probarlo desde el mismo equipo accediendo con un cliente ftp a tu mismo equipo (con la dirección IP 127.0.0.1). Debe de conectar y darte un mensaje de vsFTPd.

13.5. Configuración

La configuración del servidor se realiza en el fichero */etc/vsftpd.conf*. Dicho fichero incluye muchos comentarios (las líneas que comienzan por #) que explican la utilidad de muchas directivas.

Además, el sistema incorpora al manual del sistema una página de ayuda sobre el programa (**vsftp**) y otra sobre el ficheros de configuración (*vsftpd*). Y en el directorio */usr/share/doc/vsftpd/* hay un subdirectorio EXAMPLE con ficheros de configuración preparados para diversas tareas.

Recuerde que tras cualquier cambio que hagamos será necesario reiniciarlo para que los cambios tengan efecto.

13.6. Directivas básicas

A continuación aparecen una serie de directivas soportadas por *vsftpd.conf* acompañadas por comentarios que las explican:

ARCHIVO: vsftpd.conf

```
# Especifica el mensaje de bienvenida al servidor
ftpd_banner=Bienvenido al servidor FTP de la OSLUCA

# Permite la conexión de usuarios anónimos
anonymous_enable=YES

# Permite la conexión al FTP de los usuarios de la máquina servidora.
# Cada uno entrará en su directorio casa
local_enable=YES

# Permite que se realicen subidas de ficheros por FTP
write_enable=YES

# Registra en el log de vsftpd la actividad del servidor
xferlog_enable=YES

#Restringe la navegación de cada usuario a su directorio casa del sistema
chroot_local_user=YES
```

Capítulo 14

Seguridad

Esta sección contiene una introducción a algunas herramientas útiles que ayudan a que nuestro sistema sea más seguro ante ataques y/o eventualidades	
TIEMPO:	
NIVEL:	Unixero profesional (página 81)
INFORMACIÓN EXTRAÍDA DE:	Información sobre seguridad
PRERREQUISITOS:	Principiante ✒ Usuario iniciado ✒ Usuario habitual

Este artículo consta de las siguientes secciones:

14.1. Administración de sistemas

14.1.1. SSH

Secure Shell (SSH)[1][2], es un software cuya principal función es permitir la conexión remota segura a sistemas a través de canales inseguros, aunque también se utiliza para la ejecución de órdenes en ese

[1] Página oficial de OpenSSH
[2] SSH en Guía Ubuntu

sistema remoto o transferir archivos desde o hacia él de manera fiable.

14.2. Utilidades de cifrado

14.2.1. GPG

GPG o GNU Privacy Guard[3][4] es una herramienta para cifrado y firmas digitales, que viene a ser un reemplazo del PGP (Pretty Good Privacy) pero con la principal diferencia que es software libre licenciado bajo la GPL. GPG utiliza el estándar del IETF denominado OpenPGP.

14.2.2. TrueCrypt

TrueCrypt[5] es una aplicación para cifrar y ocultar en el ordenador datos que el usuario considere reservados empleando para ello diferentes algoritmos de cifrado como AES, Serpent y Twofish o una combinación de los mismos. Permite crear un volumen virtual cifrado en un archivo de forma rápida y transparente.

14.3. Herramientas de monitorizacion de redes

14.3.1. iptables

Netfilter es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estados del procesamiento. Netfilter es también el nombre que recibe el proyecto que se encarga de ofrecer herramientas libres para cortafuegos basados en Linux.

El componente más popular construido sobre Netfilter es iptables, una herramientas de cortafuegos que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT) para IPv4 o mantener registros de log. El proyecto ofrecía compatibilidad hacia atrás con ipchains hasta hace relativamente poco, aunque hoy día dicho soporte ya ha sido retirado al considerarse una herramienta obsoleta. El proyecto Netfilter no sólo ofrece componentes disponibles como módulos del núcleo sino que también ofrece herramientas de espacio de usuario y librerías.

iptables es el nombre de la herramienta de espacio de usuario mediante la cual el administrador puede definir políticas de filtrado del tráfico que circula por la red. El nombre iptables se utiliza frecuentemente de forma errónea para referirse a toda la infraestructura ofrecida por el proyecto Netfilter. Sin embargo, el proyecto ofrece otros subsistemas independientes de iptables tales como el connection tracking system o sistema de seguimiento de conexiones, o que, que permite encolar paquetes para que sean tratados desde espacio de usuario. iptables es un software disponible en prácticamente todas las distribuciones de Linux actuales.

[3] Página oficial de GPG
[4] Manual de instalación y uso
[5] Página oficial de TrueCrypt

14.3.2. TCP Wrappers

TCP Wrappers se encarga de definir una serie de redes o máquinas autorizados a conectar con nosotros. Cualquier administrador que desee un mínimo de seguridad ha de instalar TCP Wrappers en sus equipos; incluso algunos Unices como Linux o BSDI lo ofrecen por defecto al instalar el operativo.

14.4. Otras herramientas

14.4.1. Tripwire

La herramienta Tripwire es un comprobador de integridad para archivos y directorios de sistemas Unix: compara un conjunto de estos objetos con la información sobre los mismos almacenada previamente en una base de datos, y alerta al administrador en caso de que algo haya cambiado.

14.4.2. Nessus

En 1998 surgió Nessus, un analizador de vulnerabilidades gratuito, de código fuente libre. Este programa detecta vulnerabilidades de seguridad en sistemas Unix y redes, desde fallos conocidos en el software hasta políticas incorrectas.

14.4.3. Crack

Crack, desarrollado por el experto en seguridad Alec Muffet, es el 'adivinador' de contraseñas más utilizado en entornos Unix; actualmente se encuentra en su versión 516.5, que funciona correctamente en la mayoría de clones del sistema operativo (Linux, Solaris, OSF...). Ejecutar periódicamente Crack sobre el archivo de contraseñas de sus sistemas es algo muy recomendable para cualquier administrador mínimamente preocupado por la seguridad, sin importar que se utilicen mecanismos para obligar a los usuarios a elegir passwords aceptables.

Parte V

Apéndices y licencia

AWK

AWK es un lenguaje de programación que fue diseñado con el objetivo de procesar datos basados sobre texto y una de las primeras herramientas en aparecer en Unix. Su nombre deriva de la primera letra de los apellidos de sus autores Alfred Aho, Peter Weinberger y Brian Kernighan.

Utiliza listas en un índice ordenado por cadenas clave (listas asociativas) y expresiones regulares. Es un lenguaje ampliamente utilizado para la programación de guiones ejecutables pues añade funcionalidad a las tuberías en los sistemas operativos tipo POSIX. Está incluido en las instalaciones básicas de prácticamente todas las distribuciones de GNU/Linux.

Estructura de los programas escritos en AWK

La orden *awk* utiliza un fichero o emisión de ordenes y un fichero o emisión de entrada. El primero indica como procesar al segundo. El fichero de entrada es por lo general texto con algún formato que puede ser un fichero o bien la salida de otro mandato.

La sintaxis general utilizada para el mandato awk sigue el siguiente patrón:

```
awk 'expresión-regular { orden }'
```

Cuando se utiliza el mandato, éste examina el fichero de entrada y ejecuta la orden cuando encuentra la expresión regular especificada.

El siguiente modelo ejecutaría la orden al inicio del programa y antes de que sean procesados los datos del fichero de entrada:

```
awk 'BEGIN { orden }'
```

El siguiente modelo ejecutaría la orden al final del programa y después de que sean procesados los datos del fichero de entrada:

```
awk 'BEGIN { orden }'
```

El siguiente modelo ejecutaría la orden por cada una de las líneas del fichero de entrada:

```
awk '{ orden }'
```

Ejemplos de uso de AWK

A continuación vamos a ir realizando ejemplos con esta orden, mientras se van comentando diversas características.

- Especificar que al inicio se imprima en la salida la frase "Hola mundoz terminar el procesamiento:

```
awk 'BEGIN { print "Hola mundo"; exit }'
Hola mundo
```

  Vamos a utilizar un fichero *prueba.txt* para las pruebas siguientes:

```
echo –e Çolumna1\tColumna2\tColumna3\tColumna4\n» ejemplo.txt
cat ejemplo.txt
Columna1 Columna2 Columna3 Columna4
```

- Mostrar únicamente la columna 1 y la columna 3:

```
awk '{ print $1, $3}' ejemplo.txt
Columna1 Columna3
```

- Mostrar únicamente la columna 3 y la columna 1 y en ese orden:

```
awk '{ print $3, $1}' ejemplo.txt
Columna3 Columna1
```

  Editamos los datos de un fichero *ejemplo.txt* del siguiente modo:

```
echo –e "Dato1\tDato2\tDato3\tDato4\n">> ejemplo.txt
echo –e "Dato5\tDato6\tDato7\tDato8\n">> ejemplo.txt
echo –e "Dato9\tDato10\tDato11\tDato4\12">> ejemplo.txt
```

```
cat ejemplo.txt
Columna1 Columna2 Columna3 Columna4
Dato1 Dato2 Dato3 Dato4
Dato5 Dato6 Dato7 Dato8
Dato9 Dato10 Dato11 Dato4
```

- Mostrar la columna 1 y la columna 3 del siguiente modo:

```
awk '{ print $1, $3}' ejemplo.txt
Columna1 Columna3
Dato1 Dato3
Dato5 Dato7
Dato9 Dato11
```

- Mostrar solo la línea cuya columna contenga la expresión regular *Dato5*:

```
awk '/Dato5/ { print }' ejemplo.txt
Dato5 Dato6 Dato7 Dato8
```

- Mostrar solo la línea cuya columna contenga la expresión regular *Dato5* y además las columnas 1 y 4:

```
awk '/Dato5/ { print $1, $4}' ejemplo.txt
Dato5 Dato8
```

- Mostrar solo las líneas con más de 35 caracteres del fichero */etc/crontab*:

```
awk 'length >35' /etc/crontab
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

- Mostrar solo las líneas con menos de 35 caracteres en el fichero */etc/crontab*:

```
awk 'length <35' /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
```

Creamos un fichero *usuario.txt* con el siguiente contenido: ARCHIVO: usuario.txt

```
Fulano Algo
Calle Mengana 123
Colonia Perengana
Ciudad de Zutano, C.P. 123456
```

Para que reconozca cada línea como un registro completo, en lugar de considerar cada palabra como una columna, se utiliza '**BEGIN { FS="\n"; RS=}**', donde:

I. el valor de FS (*Field Separator* o *separador de campo*) se establece como un retorno de carro

II. el valor de RS (*Record Separator* o *separador de registro*) se establece como una línea vacía

- Imprimir los valores de cada registro (cada línea) separados por una coma y un espacio:

```
awk 'BEGIN { FS="\n"; RS= } { print $1 ", "$2 ", "$3 ", "$4 }' usuario.txt
Fulano Algo, Calle Mengana 123, Colonia Perengana, Ciudad de Zutano,
C.P. 123456
```

Información: El mandato *awk* puede contar líneas, palabras y caracteres.

- Establecer que el valor de *w* sea igual al número de campos (*New Field* o *NF*), *c* sea igual la longitud de cada campo, e imprimir el número de campos, el valor de *w* y el valor de *c*:

```
awk '{ w += NF; c += length} \
END { print \
Çampos: "NR , "\nPalabras: "w, "\nCaracteres: ç }' \
usuario.txt Campos: 4
Palabras: 12
Caracteres: 74
```

Generamos el fichero *numeros.txt* con el siguiente contenido, donde las columnas serán separadas por un tabulador:

ARCHIVO: numeros.txt

```
1 2 3 4
5 6 7 8
9 10 11 12
```

Información: El mandato *awk* puede realizar operaciones matemáticas.

- Establecer que *s* es igual a la suma del valor de los campos de la primera columna del fichero numeros.txt, e imprimir el valor de **s**:

```
awk '{ s += $1 } END { print s }' numeros.txt
15
```

- Lo mismo, pero con los valores de la columna 2:

```
awk '{ s += $2 } END { print s }' numeros.txt
18
```

- Contar la frecuencia de palabras (para ello se establece que el valor para FS sea igual a expresiones regulares que van desde la *a* a la *z* y desde la *A* a la *Z*, se establece que el valor de la variable *i* es igual a 1 y menor al número de campos):

```
awk 'BEGIN { FS="[ â-zA-Z]+"} \
{ for (i=1; i<=NF; i++) words[tolower($i)]++ } \
END { for (i in words) print i, words[i] }' /etc/crontab
7
bin 3
run 5
etc 4
sbin 3
bash 1
weekly 1
daily 1
cron 4
usr 2
path 1
shell 1
parts 5
home 1
mailto 1
monthly 1
hourly 1
root 6
```

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a

licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
See `http://www.gnu.org/copyleft/.`

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.