FUNDAMENTOS DE SISTEMAS OPERATIVOS CON ÉNFASIS EN GNU/LINUX

Wilfredo I. Pachón López

29 de octubre de 2003

Índice general

I Fundamentos de Sistemas Operativos **15**

1. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS **17**

1.1. NOCIONES BÁSICAS SOBRE HARDWARE 17

1.1.1. REGISTROS DEL PROCESADOR 18

1.1.2. EJECUCIÓN DE INSTRUCCIONES 19

1.1.3. INTERRUPCIONES 19

1.2. ¿QUÉ ES Y QUE HACE UN SISTEMA OPERATIVO (SO)? . . . 20

1.3. CONCEPTOS BÁSICOS 21

1.3.1. PROCESOS 22

1.3.1.1. MULTIPROGRAMACIÓN 24

1.3.1.2. MULTIPROCESAMIENTO 24

1.3.1.3. PROCESAMIENTO DISTRIBUIDO 24

1.3.2. ARCHIVOS 24

1.3.3. LLAMADAS AL SISTEMA 27

1.3.4. Núcleo del Sistema (*Kernel*) 28

1.3.5. Interprete de Comandos (*shell*) 28

2. GENERALIDADES DE LOS SISTEMAS OPERATIVOS 29

2.1. TIPOS DE SISTEMAS OPERATIVOS 29

2.1.1. SISTEMAS OPERATIVOS POR LA ESTRUCTURA DEL KERNEL 29

2.1.1.1. S.O. MONOLÍTICOS 30

2.1.1.2. S.O. CON CAPAS 30

2.1.1.3. S.O. CON MÁQUINA VIRTUAL 32

2.1.1.4. MODELO MICROKERNEL 33

2.1.2. SISTEMAS OPERATIVOS POR SERVICIOS 34

2.1.2.1. MONOUSUARIO 34

2.1.2.2. MULTIUSUARIO 34

2.1.2.3. MONOTAREA 35

2.1.2.4. MULTITAREA 35

2.1.2.5. UNIPROCESO 36

2.1.2.6. MULTIPROCESO 36

2.1.3. SISTEMAS OPERATIVOS POR LA FORMA DE OFRECER SUS SERVICIOS 37

2.1.3.1. SISTEMAS OPERATIVOS DE RED 37

2.1.3.2. SISTEMAS OPERATIVOS DISTRIBUÍDOS . 38

3. GESTIÓN DE PROCESOS 41

3.1. CICLOS DE VIDA DE UN PROCESO 42

3.1.1. MODELO DE CINCO ESTADOS DE UN PROCESO . . . 43

3.1.2. ESTADOS DE UN PROCESO EN UNIX® 44

3.1.2.1. Estados de un proceso en *Linux*®(*System V*) 45

3.2. CONCURRENCIA 46

3.2.1. Exclusión mútua 52

3.2.2. Interbloqueo 52

3.2.3. Inanición 52

4. Comparativa de Sistemas Operativos 53

4.1. Sistemas Operativos 55

4.1.1. FreeBSD 55

4.1.2. GNU/Linux 55

4.1.3. Mac OS X 56

4.1.4. NetBSD 56

4.1.5. OpenBSD 57

4.1.6. Windows 98 57

4.1.7. Windows 2000 57

4.1.8. Windows XP 57

4.2. Comparación a nivel administrativo 58

4.2.1. Esquema Licenciamiento 58
4.2.1.1. FreeBSD 58
4.2.1.2. GNU/Linux 59
4.2.1.3. Mac OS X 60
4.2.1.4. NetBSD 60
4.2.1.5. OpenBSD 60
4.2.1.6. Windows 98 61
4.2.1.7. Windows 2000 61
4.2.1.8. Windows XP 63
4.2.2. Estabilidad y Desempeño 63
4.2.2.1. FreeBSD 64
4.2.2.2. GNU/Linux 64
4.2.2.3. Mac OS X 65
4.2.2.4. NetBSD 65
4.2.2.5. OpenBSD 66
4.2.2.6. Windows 98 66
4.2.2.7. Windows 2000 66
4.2.2.8. Windows XP 67
4.2.3. Facilidad de uso 67
4.2.3.1. FreeBSD, NetBSD, OpenBSD, GNU/Linux 67

4.2.3.2. Mac OS X 73
4.2.3.3. Windows 98, 2000, XP 74
4.2.4. Soporte 76
4.2.4.1. FreeBSD, NetBSD, OpenBSD 77
4.2.4.2. GNU/Linux 78
4.2.4.3. Mac OS X 78
4.2.4.4. Windows 98, 2000, XP 78
4.3. Comparación a nivel técnico 78
4.3.1. Compatibilidad con Otras Plataformas 79
4.3.2. Portabilidad 80
4.3.3. Requerimientos Hardware 80

II El Sistema Operativo GNU/Linux 81

5. INTRODUCCIÓN A GNU/LINUX 85

5.1. HISTORIA DE UNIX® Y GNU/LINUX 85
5.1.1. DE MULTICS A UNIX® 85
5.1.2. LAS VARIANTES BSD Y SISTEMA V 87
5.1.3. EL ESTÁNDAR POSIX 89
5.1.4. EL PROYECTO GNU 90
5.1.5. LINUX 93

5.1.6. GNU/LINUX 94

5.2. DISTRIBUCIONES GNU/LINUX 95

5.2.1. DEBIAN 95

5.2.2. KNOPPIX 97

5.2.3. MANDRAKE 98

5.2.4. MORPHIX 98

5.2.5. REDHAT 99

5.2.6. SLACKWARE 100

5.2.7. SUSE 100

5.3. INSTALACIÓN 100

5.3.1. CONSIDERACIONES ANTES DE LA INSTALACIÓN . . 101

5.3.1.1. SISTEMAS DE ARCHIVOS 101

5.3.1.2. DISTRIBUCIÓN DEL ESPACIO (PARTICIONAMIENTO) 101

6. COMENZANDO CON GNU/LINUX 105

6.1. EL SISTEMA DE ARCHIVOS DE UNIX 105

6.1.1. TIPOS DE ARCHIVOS 105

6.1.1.1. ARCHIVOS CORRIENTES 106

6.1.1.2. DIRECTORIOS 106

6.1.1.3. ENLACES 106

6.1.2. La ruta o path 107
6.1.2.1. Rutas absolutas 107
6.1.2.2. Rutas relativas 107
6.1.3. Manipulación de archivos y directorios . . . 108
6.1.3.1. Listado del contenido de un directorio 108
6.1.3.2. Mostrar directorio actual 109
6.1.3.3. Cambiar de directorio 109
6.1.3.4. Creación de directorios 109
6.1.3.5. Borrado de directorios 110
6.1.3.6. Copiar archivos y directorios 111
6.1.3.7. Mover o renombrar archivos 112
6.1.3.8. Borrado de archivos 114
6.1.3.9. Creación de enlaces 115
6.1.4. Seguridad en archivos y directorios 116
6.1.4.1. Cambio de propietario y grupo 116
6.1.4.2. Manipulación de permisos 117
6.2. Estructura del sistema de archivos 118
6.2.1. El sistema de archivos raíz (/) 120
6.2.1.1. /etc 125
6.2.2. La jerarquía /usr 126
6.2.3. La jerarquía /var 129

A. BIOGRAFIAS **133**

A.1. DIJKSTRA, EDSGER 134

A.2. KERNIGHAN, BRIAN 136

A.3. RITCHIE, DENNIS 137

A.4. STALLMAN, RICHARD 138

A.5. TANENBAUM, ANDREW 140

A.6. THOMPSON, KENNETH 141

A.7. TORVALDS, LINUS 142

B. Traducción a Español de la GNU Free Documentation License 1.1 (GFDL) **145**

Indice de Tablas

1.1. Llamadas al sistema comunes en UNIX® 27

3.1. Interacción entre procesos 50

4.1. Precios Windows 2000 62

6.1. Opciones de mkdir 110

6.2. Opciones de rmdir 111

6.3. Opciones de cp 112

6.4. Opciones de mv 114

6.5. Opciones de rm 115

6.6. Valores Binarios/Octales de Permisos 118

6.7. Categorias de archivos 120

6.8. Jerarquia / 125

6.9. Jerarquia /usr 129

6.10. Jerarquia /var 132

Índice de figuras

1.1. Arbol de procesos en un sistema GNU/Linux 23

1.2. Un sistema de archivos típico 26

2.1. *Edsger Dijkstra* 31

2.2. Modelo de capas del sistema operativo UNIX® 31

2.3. Salá de computo de *Digital Domain* con el *cluster* para el renderizado de Titanic. 39

3.1. Representación gráfica del modelo de 5 estados 44

3.2. Rutina ejemplo (concurrencia) 47

4.1. Entorno Gráfico en UNIX 68

4.2. FreeBSD con GNOME 69

4.3. NetBSD con FluxBox 69

4.4. OpenBSD con KDE 70

4.5. GNU/Linux con AfterStep 70

4.6. GNU/Linux con Enlightenment 71

4.7. GNU/Linux con FluxBox 71

4.8. GNU/Linux con GNOME 72

4.9. GNU/Linux con IceWM 72

4.10. GNU/Linux con KDE 73

4.11. Mac OS X 74

4.12. Windows XP 75

4.13. Windows 2000 75

4.14. Windows 98 76

5.1. *Dennis Ritchie* y *Ken Thompson* en una PDP-11, trabajando en *UNIX* 87

5.2. *Richard Stallman* en el *Linux World* (China, Agos/2000) . . . 90

5.3. *Linus Torvalds* 93

5.4. Particionamineto de un disco (en una instalación de Mandrake GNU/Linux 9.1) 104

6.1. Arbol del sistema de archivos 119

6.2. Jerarquía */usr* 126

6.3. Jeraquía */var* 130

Parte I

Fundamentos de Sistemas Operativos

Capítulo 1

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

Antes de entrar en materia con el estudio de los sistemas operativos, es conveniente hacer un repaso (o mejor aún un **refuerzo**) sobre algunas nociones básicas que se deben conocer acerca del hardware de todo computador.

1.1. NOCIONES BÁSICAS SOBRE HARDWARE

De una manera un tanto simplista (es decir sin entrar en detalles) podemos ver que todo computador está compuesto por 3 componentes:

- **Procesador**: Se le conoce también con el nombre de **Unidad Central de Procesamiento (UCP)** o en inglés ***Central Process Unit*** (***CPU***), es el encargado de la coordinación de toda acción del computador, y del procesamiento de los datos. Análogamente se puede ver como el cerebro humano.

- **Memoria primaria**: En esta son almacenados todos los datos con los que trabajará el procesador, además de los programas a ejecutarsen, indistintamente la denominan también **memoria principal**.
- **Unidades de Entrada/Salida** (E/S): Estos dispositivos transportan los datos de la memoria o el procesador hacia el entorno exterior, el cual puede ser cualquier dispositivo como una impresora o un disco.

Estos diferentes dispositivos se pueden comunicar entre si gracias a la ayuda de los sistemas de interconexión.

1.1.1. REGISTROS DEL PROCESADOR

Aunque siempre se habla de un procesador como una estructura monolítica, en realidad está compuesto por diferentes módulos, siendo uno de los principales los registros. Estos cumplen funciones de almacenamiento de datos aunque de una manera muchisimo más rápida que si se utilizara la memoria principal.

Los registros del procesador se dividen según su función en dos grandes bloques:

- **Registros de control y estado**: Estos son de uso casi exclusivo del procesador para su funcionamiento interno (control de operaciones) y por algunas rutinas especiales en los sistemas operativos (para el control de la ejecución de los programas).
- **Registros de usuario:** Estos registros pueden ser utilizados por los programas de usuarios que requieran la utilización de pequeños espacios de almacenamiento pero de gran velocidad, sin embargo es de aclarar que no todos los lenguajes de programación permiten la creación de programas que puedan interactuar con estos, tan sólo algunos pocos (por ejemplo **C**).

1.1.2. EJECUCIÓN DE INSTRUCCIONES

Todo programa es una sucesión de instrucciones que le indican al procesador la acción que este debe realizar, pero estas no son pasadas de forma directa del programa al procesador, antes de esto se llevan a cabo algunos pasos intermedios.

Al ejecutar un programa este es «cargado» en la memoria principal, la cual pasará a solicitud del procesador una a una cada instrucción almacenada y este se encargará de su ejecución y de una nueva petición a la memoria primaria para la siguiente instrucción, aunque pueda parecer un proceso lento en realidad es ejecutado en fracciones de segundo por lo cual al ser humano le parece que se lleva a cabo de forma instantanea, aunque para ajustarse a la realidad esta es una visión simplificada del proceso, ya que conlleva una serie de pasos más en los cuales entran a jugar un papel tracendental algunos registros del procesador, pero esta explicación en detalle se deja para su consulta en textos más avanzados ya que se sale un poco del ámbito de profundidad de este curso.

1.1.3. INTERRUPCIONES

La comunicación entre el procesador y la memoria principal es a gran velocidad, sin embargo la mayoría de programas deben interactuar con algún tipo de dispositivo de E/S, durante este proceso la ejecución del programa se ve interrumpida ya que la comunicación con dichos dispositivos es significativamente más lenta que con la memoria, en algunas ocaciones esta operación puede ser incluso más lenta ya que puede requerir la intervención humana y obviamente el tiempo de reacción se reducirá de manera drástica. Un ejemplo típico de estas situaciones y que muy seguramente a todos nos es familiar se presenta cuando estamos utilizando un procesador de textos para redactar un documento, el aplicativo debe estar pendiente de cada acción desde el teclado o el ratón (o algún otro dispositivo que pueda estar involucrado) y cuando se produzca, reponder de manera adecuada según corresponda, por más rápido que sea el

digitador siempre será más lento que el procesador e incluso que algunos dispositivos por lo cual entre cada pulsación del teclado el procesador puede (y debe) estar realizando otras acciones lo que implica que el sistema operativo soporte interrupciones.

Algunos sistemas operativos ejecutan todos los trabajos en forma de lotes (*batch*), es decir no podían hacer ningún otro trabajo hasta tanto el programa en ejecución no terminase su trabajo, por lo cual cuando este debía por ejemplo leer de un CD o un disquette, pareciera que el programa y hasta la máquina completa se bloquease por un tiempo y después «milagrosamente» se recuperase. Para solucionar este problema la mayoría de S.O. modernos utilizan el concepto de **interrupción.**

Este concepto se basa en el trabajo «colaborativo», es decir el procesador sede el control de la E/S a un módulo que se encarga de ejcutar este tipo de operaciones y esperar hasta que estas se completen y cuando esto suceda avisarle al procesador (que se encuentra mientras tanto realizando otras instrucciones -bien sea del mismo programa o de algún otro-) que puede continuar con las operaciones subsiguientes que quedaron pendientes cuando la petición de E/S.

1.2. ¿QUÉ ES Y QUE HACE UN SISTEMA OPERATIVO (SO)?

En términos generales un sistema operativo se puede definir como el componente de software encargado de la interrelación de los programas con la máquina y de este conjunto con el usuario.

Esta definición es apropiada para dar una idea de la importancia de un sistema operativo y de la necesidad de este, sin embargo, como se dijo es una visión muy simplificada por lo cual se hace necesario ayudar un poco más a entender ¿para que sirve? o ¿como se utiliza? un S.O.

Una de las principales funciones de un S.O. es el de servir de **máquina virtual** o **extendida**, esta funcionalidad es la que le permite al programador abstraer la capa de hardware y utilizar una serie de llamadas al sistema operativo para que este sea el encargado de tratar a bajo nivel con los componentes físicos, si no existiese esta funcionalidad todo programador que en una aplicación necesite grabar en un disco duro o disquette y/o imprimir en una impresora, debería escribir el código que le permita hacer esto pero de forma individual para cada dispositivo al que desee dar soperte, por lo tanto si el programador sólo incluye funcionalidad para las impresoras marca A y B, si el usuario tiene una de marca C no podría utilizar las funciones de impresión, lo mismo sucedería con los dispositivos de almacenamiento y en general se iría complicando entre más dispositivos sea necesario utilizar y entre mayor sea la diversidad de estos dispositivos; es por este motivo que en la definición «rápida» se dice que provee una interface entre las aplicaciones y el hardware.

La otra gran tarea de un sistema operativo es servir de «agente del orden», es decir es quien controla el acceso y la utilización de los recursos del sistema y los distribuye de forma tal que los más voraces no lo consuman todo dejando a los demás sin estos, además también controla quién hace uso de estos recursos y en que momento; sin esta función del sistema operativo todos los procesos podrían por ejemplo enviar al mismo tiempo peticiones de escritura/lectura a un disco y por consiguiente no se podría garantizar que se complete con exito ninguna de las peticiones ya que no habría forma de controlar en que momento este se encuentra ocupado cumpliendo la misión encomendada por un proceso. Por tanto un sistema operativo debe poder conocer cuando un recurso está siendo utilizado y en que momento está libre y dependiendo de esto permitir o no su uso.

1.3. CONCEPTOS BÁSICOS

Antes de poder continuar explorando el increible mundo de los sistemas operativos es recomendable hacer una pausa para hablar brevemente sobre algunos

conceptos o términos que se han venido mencionando en las secciones previas y que debido a su gran importancia deberemos continuar viendo.

1.3.1. PROCESOS

La gran mayoría de sistemas operativos modernos tiene su base fundamental o su pilar central en este concepto, un proceso es la forma en que un sistema operativo trata a un programa durante su ejecución. Como ya se había mencionado, para que el procesador pueda trabajar con un programa es necesario que exista algo de información adicional (como los registros de operación en el procesador, los datos de entrada del programa, etc) acompañando al código objeto (o ejecutable), por tanto cuando se dice que proceso es la forma en que el sistema trata a todo programa durante su ejecución, estamos haciendo referencia a todo este conjunto.

Cuando un sistema operativo arranca este crea algunos procesos principales que seran los encargados de crear todos los demás procesos, como a su vez cualquier proceso puede crear otros procesos (denominados «hijos»), obtenemos una estructura jerárquica en forma de arbol donde todo proceso (execpto el o los iniciales del S.O.) tiene un padre (o antecesor) y a su vez tiene la capacidad de tener uno o más hijos, a este conjunto conformado por un padre y todos sus hijos (y recursivamente los hijos de estos) se le denomina familia de procesos. En el capítulo 3 se trata más en profunidad otros temas referentes a los procesos.

La figura 1.1 nos presenta un ejemplo real de un arbol de procesos (tomado con la aplicación *pstree*) en un servidor *Debian GNU/Linux*, como se puede ver todos los procesos tienen un padre común (*init*) y cada proceso tiene uno o más hijos (el proceso padre se encuentra a la extrema izquierda de cada línea y los hijos unidos por la línea punteada), que a su vez pueden tener también otros hijos.

Figura 1.1: Arbol de procesos en un sistema GNU/Linux

```
init-+-amavis-milter---amavis-milter---amavis-milter
     |-apache---10*[apache]
     |-apache-ssl-+-5*[apache-ssl]
     |            `-gcache
     |-autolog
     |-bash---dselect---install---apt-get---dpkg-+-dpkg-deb
     |                                           `-dpkg-deb---2*[dpkg-deb]
     |-bash---pstree
     |-cron
     |-2*[daemon---less]
     |-4*[getty]
     |-inetd
     |-keventd
     |-khubd
     |-5*[kjournald]
     |-lockd---rpciod
     |-master-+-pickup
     |        `-qmgr
     |-4*[mount]
     |-mysqld_safe---mysqld---mysqld---2*[mysqld]
     |-8*[nfsd]
     |-nmbd
     |-popa3d
     |-portmap
     |-2*[portsentry]
     |-postmaster-+-postmaster---postmaster
     |            `-2*[postmaster]
     |-rpc.mountd
     |-rpc.statd
     |-smbd
     |-squid---squid-+-7*[ncsa_auth]
     |               `-squid---16*[squid]
     |-sshd
     |-syslog-ng
     `-update-menus
```

Dentro de esta introducción a los procesos vale la pena reseñar otros subtemas de vital importancia dentro de los procesos:

- Multiprogramación
- Multiprocesamiento
- Procesamiento distribuido

1.3.1.1. MULTIPROGRAMACIÓN

Intimamente ligado al concepto de procesos, la multiprogramación permite administrar varios procesos en una sóla CPU, alternando la ejecución de estos a través de controles de tiempo, asignando límites temporales de ejecución a cada proceso y alternandolos a medidad que este limite es alcanzado.

1.3.1.2. MULTIPROCESAMIENTO

Este es otro importante concepto dentro de la administración de procesos, sin embargo a diferencia de la multiprogramación esta gestión se basa en la distribución de los procesos en multiples procesadores.

1.3.1.3. PROCESAMIENTO DISTRIBUIDO

En este caso la administración de los procesos se complica un poco más al tener que distribuir la carga de procesamiento entre diferentes CPUs pero con el aditivo especial de que estas se encuentran en otras máquinas, la esencia del procesamiento distribuido es manejar todos los recursos de que dispone el conjunto distribuido como si fuesen una sóla máquina.

1.3.2. ARCHIVOS

Junto a los procesos el otro gran componente de un sistema operativo es el sistema de archivos. todo proceso requiere para su ejecución que los datos que va a utilizar se encuentren físicamente en algún sitio; lo que hace el sistema sistema de archivos precisamente es permitir que esta información se pueda organizar de una manera lógica y sencilla. Todo sistema operativo debe brindar las herramientas (en forma de llamadas al sistema o instrucciones) para que el sistema de archivos sea funcional, entre las principales se pueden encontrar:

- Las de creación y destrucción de archivos.

- Las de apertura y cierre de los archivos.

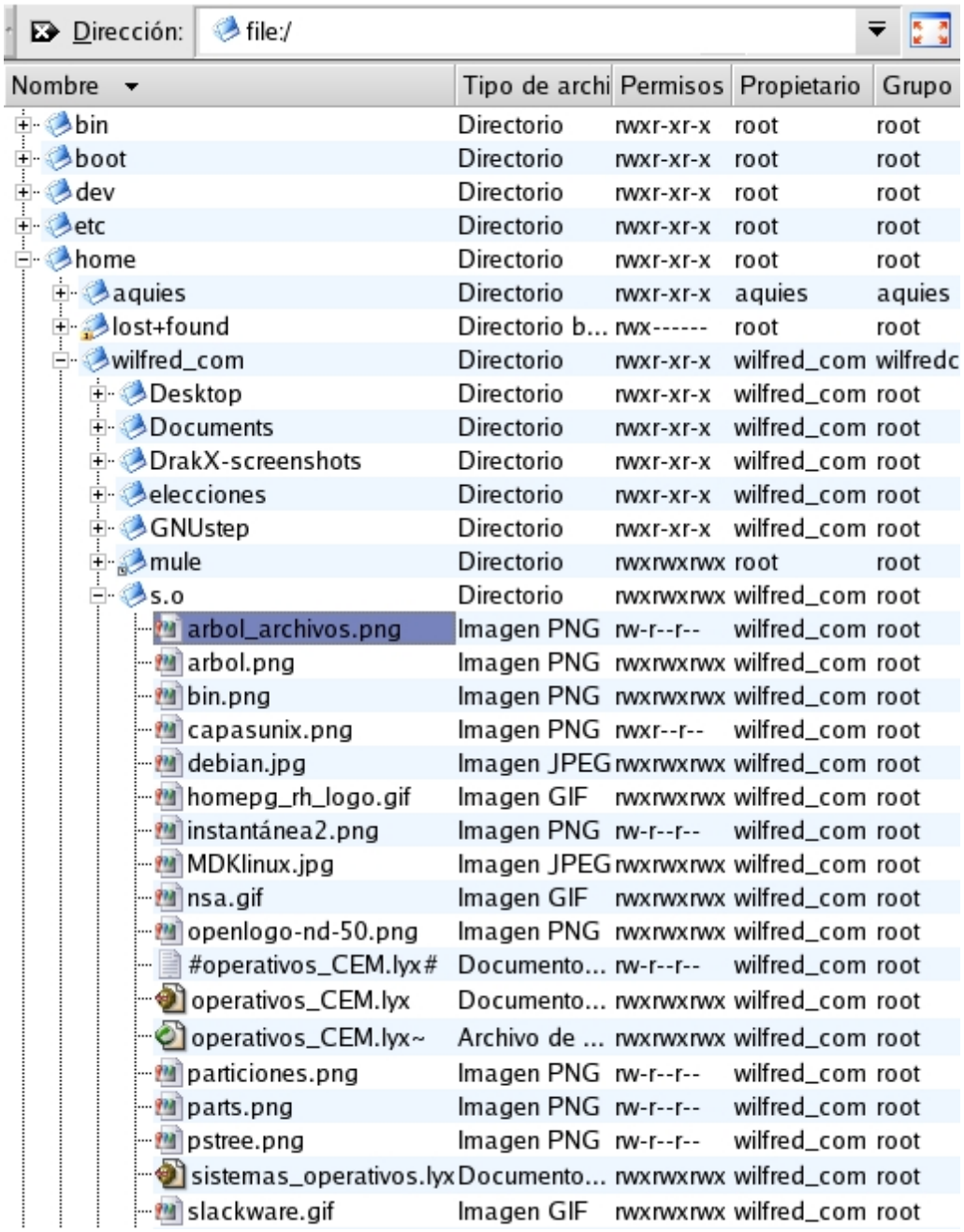
- Las de lectura y escritura.

Adicionalmente también la mayoría de sistemas permiten tener cierto control sobre los archivos pudiendo asignar características como la seguridad.

Un **archivo** realmente es una colección de *bytes* relacionados bajo un único nombre, adcionalmente como bien conocemos desde nuestra experiencia práctica, los archivos también se encuentran organizados bajo una estructura que los relaciona lógicamente, esta estructura se denomina **directorio** (algunos sistemas los denominan también como carpetas).

Debido a esto podemos ver que el sistema de archivos es también una estructura jerárquica, la cual tiene un inicio en el directorio raíz (algunos sistemas operativos como los derivados de MS DOS -este incluido- tienen un directorio raíz por cada partición encontrada en los discos). La figura 1.2 nos presenta un ejemplo de la organización jerárquica de un sistema de archivos (incluyendo información adicional como la seguridad, usuario y grupo propietarios del archivo).

Figura 1.2: Un sistema de archivos típico



1.3.3. LLAMADAS AL SISTEMA

Las llamadas al sistema son unas instrucciones especiales que utiliza el sistema para su comunicación con los programas y de estos hacia el sistema. Estan estrechamente realcionadas con las funciones de la libreria estandard en cada S.O., ya que para cada una de estas existe su similar como llamada al sistema. Algunas llamadas comunes en un sistema UNIX®se resumen en la tabla , resumén de un trabajo más completo en [TANENBAUM-93].

Llamada	**Descripción**
Administración de procesos	
`pid = fork ( )`	Crea un proceso hijo y retorna su identificador de proceso
`s =kill (pid, sig)`	Envía una señal *sig* al proceso identificado por *pid*.
Archivos y directorios	
`fd = open (file, how)`	Abre un archivo *file*, para lectura o escritura (dependiendo de *how*), retorna un descriptor de archivo *fd*
`s = close (fd)`	Cierra el archivo descrito por *fd* si este se encuentra abierto
`n = read (fd, buffer, nbytes)`	Lee *nbytes* de datos del archivo descrito por *fd* y los coloca en el almacen *buffer*
`n = write (fd, buffer, nbytes)`	Escribe *nbytes* de datos al archivo descrito por *fd*, tomados desde el almacen *buffer*
`s = mkdir (name, mode)`	Crea un directorio con nombre *name*
`s = rmdir (name)`	Elimina el direcorio *name* siempre y cuando está vacío
`s = chdir (name)`	Cambia el directorio de trabajo a *name*

Tabla 1.1: Llamadas al sistema comunes en UNIX®

1.3.4. Núcleo del Sistema (*Kernel*)

El *kernel* de un S.O. es su base fundamental, es el que se encarga de toda la comunicación entre hardware y software, así como de la administración del mismo. La mayoría de las caracteristicas y conceptos que se representan en esta primera parte del libro hacen referencia por lo general al *kernel* aún cuando se el termino mencionado sea el más general: (Sistema Operativo); sin embargo es de aclarar que esto se hace sólo como medida práctica ya que en realidad el *kernel* es un componente más del amplio conjunto de aplicaciones que debe constituir un S.O.

1.3.5. Interprete de Comandos (*shell*)

Es el componente del S.O, que le permite al usuario introducir ordenes al equipo, literelamente se puede traducir como «cáscara», esto es por que el *shell* esconde los detalles internos del S.O. o *kernel* sobre el que se está ejecutandose. Algunas personas denominan «*shell* visual» a las Interfaces Gráficas de Usuario (como *GNOME*), debido a su análogía en operaciones con el *shell* tradicional (por lo general denominado «línea de comandos»).

Capítulo 2

GENERALIDADES DE LOS SISTEMAS OPERATIVOS

2.1. TIPOS DE SISTEMAS OPERATIVOS

Siguiendo la tendencia de todo lo relacionado con la informática, los sistemas operativos también son susceptibles de clasificarlos o dividirlos dependiendo de sus carácterísticas:

- Clasificación por su estructura.
- Clasificación por servicios ofrecidos.
- Clasificación por el soporte a los servicios.

2.1.1. SISTEMAS OPERATIVOS POR LA ESTRUCTURA DEL KERNEL

De acuerdo a su estructura los sistemas operativos se pueden clasificar de la siguiente manera.

2.1.1.1. S.O. MONOLÍTICOS

Es la estructura de los primeros sistemas operativos constituídos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de estructura son:

- Construcción del programa final a base de módulos compilados separadamente que se unen a través del encadenador (*linker*).
- Buena definición de parámetros de enlace entre las distintas rutinas existentes, lo que puede provocar mucho acoplamiento.
- Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos del computador, como memoria, disco, etc.
- Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

2.1.1.2. S.O. CON CAPAS

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía subpartes y esto organizado en forma de niveles.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos.

Se constituyó una estructura jerárquica o de niveles en los sistemas operativos, el primero de los cuales fue denominado *THE* (*Technische Hogeschool*,

Figura 2.1: *Edsger Dijkstra*

Eindhoven), de *Dijkstra*[1], que se utilizó con fines didácticos. Se puede pensar también en estos sistemas como si fueran 'multicapa'. *Multics* (`http://www.multicians.org`) y *UNIX*®(y por ende sus derivados) caen en esa categoría.

Figura 2.2: Modelo de capas del sistema operativo UNIX®

En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denomina-

[1] *Edsger W. Dijkstra* (30/05/1930 - 06/08/2002) Holandés físico teórico de la universidad de Leiden, una corta biografia puede ser encontrada en el , de la página

da de anillos concéntricos o "rings". En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (*trap*), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas. La figura 2.2 da un ejemplo real de como se organizan las capas en un sistema operativo, en este caso se presenta la de *UNIX*®dedibo al énfasis que en este libro se le da a los sistemas operativos de este tipo.

2.1.1.3. S.O. CON MÁQUINA VIRTUAL

Se trata de un tipo de sistemas operativos que presentan una interface a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente.

Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de sistemas:

1. La multiprogramación y
2. La máquina extendida.

El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes. El núcleo de estos sistemas operativos se denomina monitor virtual y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario.

Un ejemplo de este tipo de sistemas es *vmware* (`http://www.vmware.com`), el cual permite gracias a la tecnología de máquina virtual ejecutar diferentes S.O. en la misma máquina.

2.1.1.4. MODELO MICROKERNEL

El tipo más reciente de sistemas operativos es el denominado Cliente/Servidor o *Microkernel*, que puede ser ejecutado en la mayoría de los computadores, ya sean grandes o pequeños. Este sistema sirve para toda clase de aplicaciones por tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro. Este paradigma ofrece gran flexibilidad en cuanto a los servicios posibles en el sistema final, ya que el núcleo provee solamente funciones muy básicas de memoria, entrada/salida, archivos y procesos, dejando a los servidores proveer la mayoría que el usuario final o programador puede usar. Estos servidores deben tener mecanismos de seguridad y protección que, a su vez, serán filtrados por el núcleo que controla el hardware. Uno de los precursores de este tipo de S.O. es *Mach* (`http://www-2.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html`), en la actualidad y como se relata más adelante en la sección 5.1.4 (pag. 90), el proyecto GNU desarrolla el *kernel Hurd* (`http://www.gnu.org/software/hurd`) y ya se encuentra disponible (aunque sólo para propósitos de trabajar en su desarrollo, ya que aún no se ha lanzado oficialmente) un sistema operativo completo con base en este: *Debian GNU/Hurd* (`http://www.debian.org/ports/hurd`).

2.1.2. SISTEMAS OPERATIVOS POR SERVICIOS

Esta clasificación es la más comúnmente usada y conocida desde el punto de vista del usuario final.

2.1.2.1. MONOUSUARIO

Los sistemas operativos monousuario son aquéllos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga el computador o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Como ejmplo de este tipo se pueden mencionar entre otros:

- *MS DOS*
- *MS Windows 9x* y *Me*
- *Mac OS* (antes de *OS X*)

2.1.2.2. MULTIUSUARIO

Los sistemas operativos multiusuario son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas al computador o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

Algunos sistemas tipo UNIX®como *GNU/Linux* ofrecen además de la posibilidad de trabajar con terminales remotas, el trabajo con terminales «virtuales», esta característica permite emular en una sóla máquina el trabajo con varias terminales remotas ya que es posible iniciar sesión como diferentes usuarios (o como el mismo varias veces) utilizando el mismo *hardware* pero de forma tal que paresca que cada uno trabaja con una terminal independiente. Ejemplos de S.O. multiusuario:

- *UNIX*® (y sus derivados)
- *MS Windows*® 2000
- *Mac OS*® *X*

2.1.2.3. MONOTAREA

Los sistemas monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo sólo una tarea a la vez. Ejemplos:

- *MS DOS*
- *MS Windows 3.X, 95* (estos sistemas tan sólo simulaban la multitarea, pero realmente pertenecen a esta clasificación)

2.1.2.4. MULTITAREA

Un sistema operativo multitarea es aquél que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, puede estar escuchando música con un reproductor de CD, escribiendo un programa, copiando archivos entre directorios y descargando música de *Internet* (además de mantener una interface gráfica). Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

- *Mac OS* ®
- *UNIX*® (y derivados)
- *MS Windows*® *98, Me, 2000* y *XP*.

2.1.2.5. UNIPROCESO

Un sistema operativo uniproceso es aquél que es capaz de manejar solamente un procesador del computador, de manera que si el computador tuviese más de uno le sería inútil. El ejemplo más típico de este tipo de sistemas es el *MS DOS*®y *MacOS*®(versiones antiguas).

2.1.2.6. MULTIPROCESO

Un sistema operativo multiproceso se refiere al número de procesadores del sistema, que es más de uno y éste es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas:

- Simétricamente
- Asimétricamente.

Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos. Cuando se trabaja de manera simétrica, los procesos o partes de ellos (hilos o *threads*) son enviados indistintamente a cualquiera de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema. Se dice que un hilo es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto. Un aspecto importante a considerar en estos sistemas es la forma de crear aplicaciones para aprovechar los varios procesadores. Existen aplicaciones que fueron hechas para correr en sistemas monoproceso que no toman ninguna ventaja a menos que el sistema operativo o el compilador detecte secciones de código paralelizable, los cuales son ejecutados al

mismo tiempo en procesadores diferentes. Por otro lado, el programador puede modificar sus algoritmos y aprovechar por sí mismo esta facilidad, pero esta última opción la mayoría de las veces es costosa en horas hombre y muy tediosa, obligando al programador a ocupar tanto o más tiempo a la paralelización que a elaborar el algoritmo inicial.

2.1.3. Sistemas Operativos por la Forma de Ofrecer sus Servicios

Esta clasificación también se refiere a una visión externa, que en este caso se refiere a la del usuario: el cómo accesa los servicios. Bajo esta clasificación se pueden detectar dos tipos principales: sistemas operativos de red y sistemas operativos distribuídos.

2.1.3.1. Sistemas Operativos de Red

Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras máquinas por medio de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades. El punto crucial de estos sistemas es que el usuario debe conocer la ubicación de los recursos que desee accesar (p.e. dirección y/o nombre de la máquina remota, ruta o camino al recurso compartido), e incluso en ocaciones puede ser necesario incluso conocer la sintaxis de una serie de instrucciones necesarias para la utilización del recurso (vale la pena aclarar que esto cada vez es menos frecuente debido al surgimiento de interfaces más «amigables» con el usuario, aunque sin lograr superar la flexibilidad que una línea de comandos puede brindar).

2.1.3.2. SISTEMAS OPERATIVOS DISTRIBUÍDOS

Los sistemas operativos distribuídos abarcan los servicios de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso) en una sola máquina virtual que el usuario accesa en forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y simplementa los usa como si todos ellos fuesen locales a su lugar de trabajo habitual. La labor del sistema operativo distribuido es la de permitir a las aplicaciones hacer uso de los diversos procesadores, memorias, discos y en general todo dispositivo conectado al sistema multiple, tal como si se encontraran físicamente en la misma máquina. Las razones para crear o adoptar sistemas distribuídos se dan por dos razones principales: por necesidad (debido a que los problemas a resolver son inherentemente distribuídos) o porque se desea tener más confiabilidad y disponibilidad de recursos. Un ejemplo de aplicación de sistemas de *cluster* o distribuidos es en la renderización de algunas partes (o toda ella) de una película cinematográfica, en especial cunado la carga de efectos especiales visuales requiere grandes capacidades de procesamiento.

Un caso de renombre ha sido la utilización de un *cluster* con máquinas corriendo *GNU/Linux* en la renderización de los efectos de la película Titanic, un excelente artículo titulado «*Linux Helps Bring Titanic to Life*» (*Linux* ayuda a *Titanic* a venir a la vida) donde uno de los ingenieros de *Digital Domain* (Empresa especializada en efectos especiales y rederización de películas cinematográficas) explican la forma en que se llevó a cabo este trabajo. Ver [STRAUSS-98]. Sin embargo no sólo en el cine esto resulta efectivo, el procesamiento en paralelo es fundamental en los grandes centros de investigación como por ejemplo el *Earth Simulator Center* (Centro de Simulación de la Tierra) laboratorio de investigación en aspectos relacionados con nuestro planeta y el ser humano, el cual trabaja gracias a un Super *Cluster* que se compone de 40 *clusters* con alrededor de 16 máquinas cada uno, integrados bajo un sistema operativo basado en UNIX®denominado "SUPER-UX"

[3]Fuente: [STRAUSS-98]

Figura 2.3: Salá de computo de *Digital Domain* con el *cluster* para el renderizado de Titanic[3].

(una versión mejorada del UNIX®de NEC); este super-computador aparece en la primera posición de la Lista del Top 500 de Sitios de Supercomputación `http://www.top500.org/list/`, acompañado en la tercera posición de un *cluster* de máquinas *GNU/Linux* en el Laboratorio Nacional Lawrence Livermore (del Departamento de Energía de los Estados Unidos).

VENTAJAS DE LOS SISTEMAS DISTRIBUÍDOS En general, los sistemas distribuídos (no solamente los sistemas operativos) exhiben algunas ventajas sobre los sistemas centralizados las cuales se describen enseguida.

- **Economía**: El cociente precio/desempeño de la suma del poder de los procesadores separados contra el poder de uno solo centralizado es mejor cuando están distribuídos.

- **Velocidad**: Relacionado con el punto anterior, la velocidad sumada es muy superior.
- **Confiabilidad**: Si una sola máquina falla, el sistema total sigue funcionando.
- **Crecimiento**: El poder total del sistema puede irse incrementando al añadir pequeños sistemas, lo cual es mucho más difícil en un sistema centralizado y caro.
- **Distribución**: Algunas aplicaciones requieren de por sí una distribución física.

Por otro lado, los sistemas distribuídos también exhiben algunas ventajas sobre sistemas aislados. Estas ventajas son:

- **Compartir datos**: Un sistema distribuído permite compartir datos más fácilmente que los sistemas aislados, que tendrian que duplicarlos en cada nodo para lograrlo.
- **Compartir dispositivos**: Un sistema distribuído permite accesar dispositivos desde cualquier nodo en forma transparente, lo cual es imposible con los sistemas aislados. El sistema distribuído logra un efecto sinergético.
- **Comunicaciones**: La comunicación persona a persona es factible en los sistemas distribuídos, en los sistemas aislados no.
- **Flexibilidad**: Al poder agregar o remover máquinas del conjunto distribuido, la flexibilidad que se logra es bastante grande aunque sin sacrificar otros factores, por tanto dependiendo de la tarea (o tareas) a realizar es posible aumentar o disminuir (rara vez) el poder de computo.

Capítulo 3

GESTIÓN DE PROCESOS

Proceso es el nombre con el que se denomina la ejecución de un programa individual, representado por una serie de instrucciones que el procesador debe ejecutar, la mayoría de sistemas operativos modernos basan toda su estructura de diseño alrededor de este concepto, debido a esta importancia, Stallings (en [STALLINGS-97]) presenta tres requisitos fundamentales que debe seguir todo sistema operativo en relación a los procesos:

- «El sistema operativo debe intercalar la ejecución de un conjunto de procesos para maximizar la utilización del procesador ofreciendo a la vez un tiempo de respuesta razonable».
- «El sistema operativo debe asignar los recursos a los procesos en conformidad con una política específica (por ejemplo, ciertas funciones o aplicaciones son de prioridad más alta), evitando, al mismo tiempo el interbloqueo»[1].
- «El sistema operativo podría tener que dar soporte a la comunicación entre procesos y la creación de procesos por parte del usuario, labores que pueden ser de ayuda en la estructuración de aplicaciones».

[1] El interbloqueo se presenta cuando dos o más procesos se bloquean mutuamente y de manera permanenete debido a algún factor externo como puede ser la espera de un recurso físico.

3.1. CICLOS DE VIDA DE UN PROCESO

Todo proceso dependiendo de la forma como se ha diseñado el sistema operativo puede tener dos o más estados, un estado como su nombre lo indica es una condición especial de la ejecución del proceso en un instante determinado de tiempo.

El modelo más sencillo obviamente es en el que sólo existen dos estados:

- En ejecución
- No ejecutandose

Esto quiere decir que el proceso puede en un momento estarse ejecutando y algún tiempo despúes no, el primer estado siempre es «No ejecuntadose» ya que este se da tan pronto se crea el proceso y este queda en espera para su proxima ejecución, en cualquier momento cuando el proceso se encuentra en estado «En ejecución» puede ocurrir un suceso interno o externo (p.e Terminación normal, Fallo de E/S, Terminación de tiempo de ejecución, No hay memoria disponible, etc[2]) que lo lleve de nuevo al estado «No ejecutandose», es de hacer notar que este modelo no permite el uso de interrupciones (ver. sección 1.1.3, pag. 19) debido a que no hay un estado de «bloqueo» o similar que permita detener por momentos la ejecución del proceso y reanudarla después desde el punto en que fue interrumpida.

Este modelo como se puede ver es bastante sencillo pero obviamente sería demasiado limitado para satisfacer las necesidades de un sistema operativo moderno por lo cual los científicos de la computación han ideado (y aplicado) modelos con mayor cantidad de estados, uno de los más completos es el que le otorga la posibilidad al proceso de tener 5 estados (o más).

[2]Para conocer más de las posibles causas de la terminación de un proceso se sugiere consultar la tabla «Razones para la Terminación de un Proceso» en [STALLINGS-97].

Debido a la complejidad que requiere una explicación detallada de un modelo tan complejo y que a su vez no es suficientemente necesaria para un curso de nivel introductorio (como lo es al que va dirigido este libro), sólo se dará una explicación muy general y tratando siempre de utilizar un lenguaje no demasiado complejo.

3.1.1. MODELO DE CINCO ESTADOS DE UN PROCESO

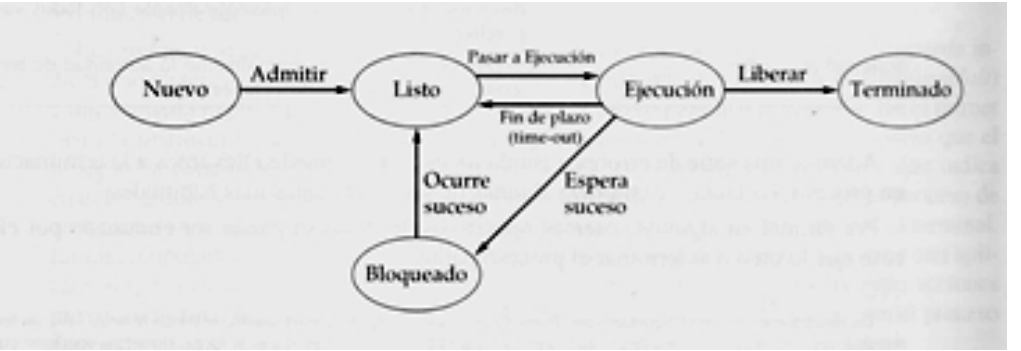
En este modelo lo que se ha hecho es dividir los estados «No ejecutandose» y «En ejecución» en otros posibles estados para brindar al sistema operativo mayor flexibilidad, estos cinco estados son:

- **Nuevo**: Este es el primer estado de todo proceso, se da tan pronto el S.O. lo crea pero aún no ha entrado a la cola de procesos listos para ser ejecutados, un proceso sólo se puede encontar en este estado una única vez durante toda su existencia.

- **Listo**: Ha este estado se llega después de pasar por «nuevo» o cuando se reanuda después de un estado «bloqueado» o «en ejecución». Cuando un proceso se encuentra en estado «listo» quiere indicarle al procesador que está disponible para su ejecución.

- **Ejecución**: Este estado se presenta cuando el procesador está trabajando con las instrucciones del proceso, es decir lo está ejecutando. Al él sólo se llega desde un estado «listo». La mayoría de sistemas operativos modernos se basan en el concepto de la multiprogramación, por lo cual un proceso no puede durar en ejecución eternamente, por este motivo cada sistema operativo asigna una cantidad de tiempo a cada proceso para ejecutarse, cuando este tiempo termina pero el proceso aún no termina su ejecución, el sistema guarda la información de estado de ejecución del proceso y lo regresa a la cola de procesos con estado «listo».

Figura 3.1: Representación gráfica del modelo de 5 estados[3]



- **Bloqueado**: Cuando ocurre un suceso que requiera algún tipo de espera por parte del proceso para poder continuar con su ejecución, este pasa al estado «bloqueado» mientras dura la espera, cuando el suceso que motivó la espera por parte del proceso ocurre, este pasa de nuevo al estado «listo».

- **Terminado**: En el momento en que el proceso ya no necesita continuar ejecutandose, los recursos que éste necesitó para tal fin son liberados y el proceso pasa al estado de «Terminado» con lo que concluye su ciclo de vida.

Es importante hacer claridad que este modelo es presentado en algunos textos con una variante en la que se suprime el primer estado («listo») y el último («terminado») pero la escencia es la misma, las transciciones entre los estados 2 a 4 son exactamente iguales.

3.1.2. ESTADOS DE UN PROCESO EN UNIX®

Como se hizo la salvedad, el modelo de 5 estados es solamente **un modelo** y no una camisa de fuerza para los desarrolladores, en este apartado se presentará,

[3]Fuente: [STALLINGS-97]

el utilizado por la mayoría de derivados de *UNIX®System V* (como ***Linux®***, *Irix®*o *Solaris®*).

3.1.2.1. Estados de un proceso en *Linux®*(*System V*)

Antes de continuar, es importante decir que el modelo que se presenta a continuación está basado exclusivamente en *Linux®*versión 2.4.x, es posible que existan algunas pequeñas diferencias en núcleos de versiones anteriores (o posteriores) y aún más si se trata de S.O. diferentes, pero la escencia es la misma en todos los que siguen el estilo del sistema V, paraa conocer más sobre el manejo de procesos y otros aspectos importante del *kernel* se debe buscar el documento «*Linux Kernel 2.4 Internals*» ([AIVAZIAN-02in]) en `http://www.tldp.org/guides.html` o su traducción al español por el Proyecto LUCAS: [AIVAZIAN-02es] (Puede estar desactualizada respecto a la última versión del original).

Linux® 2.4.x implementa tareas o procesos que pueden existir en uno de 5 estados[4] (7 si tenemos en cuenta los dos últimos que no son realmente estados):

1. **Ejecutandose:** Este estado le indica al núcleo que el proceso «se supone debe estar» en la cola de ejecución. El entrecomillado indica que no hay completa certeza de que realmente se encuentre en dicha estructura, esto se debe a la no átomicidad del trabajo de marcar el proceso como *TASK_RUNNING* y colocarlo en la cola de ejecución.

2. **Proceso con interrupciones:** En este estado el proceso se encuentra esperando (o durmiendo) pero puede ser despertado por una señal o por termino del tiempo-

[4]Para conocer que estados puede tener un proceso en *Linux* es posible revisar el archivo `include/linux/sched.h` en el directrio donde se haya desempaquetado el código fuente (típicamente `/usr/src/linux`)

3. **Proceso sin interrupciones**: Acá el proceso también se encuentra dormido pero a diferecia del anterior, el proceso no puede ser despertado por eventos externos.

4. **Proceso Zombi**: Este estado se presenta cuando la tarea ha terminado pero aún faltan por liberar algunos recursos por lo que el proceso aún puede considerarse como «vivo» dentro del sistema.

5. **Proceso Parado**: El proceso se encuentra parado bien sea por una señal de control de trabajo o por la señal de sistema *ptrace*.

6. **Proceso Exlusivo**: Realmente no es un estado de algún proceso pero puede ser uno de (2) o (3). Esto significa que cuando este proceso se encuentre dormido o en una cola de espera junto a otros, este puede ser despertado sólo en lugar de causar el problema de "movimiento general" despertando a todos los que están esperando.

7. **Terminado**: Cuando el proceso a concluido satisfactoriamente se puede decir que ha entrado en este estado (sin serlo realmente), si se revisa el archivo de cabecera mencionado en la nota al pie de página, se puede notar que no esite una constante con valor para este estado (es más no existe referencia a este estado en ningún otro lugar), esto se debe a que cuando el proceso finaliza ya no hay necesidad de mantener información alguna sobre él.

3.2. CONCURRENCIA

Cuando consultamos en un diccionario por el significado de esta palabra encorntramos entre una de las definiciones: «Acaecimiento de varios sucesos o cosas de manera simultanea».

Este mismo significado es el que adquiere éste termino cuando se relaciona con la teoría de los sistemas operativos, la concurrencia se presenta cuando dos o

más porcesos deben hacer uso de un mismo recurso a un mismo tiempo. Alguien puede pensar (con justa razón) que se está presentando una contradicción o que se está hablando sólo de sistemas con más de una CPU, ya que en una sóla CPU no se puede ejecutar más de un proceso en al mismo tiempo; esto es verdad pero hay que tener en cuenta otros factores, para poder explicar mejor esto utilizaremos el siguiente ejemplo:

Tenemos la siguiente rutina:

Figura 3.2: Rutina ejemplo (concurrencia)

```
1                                   void
2                              ejemplo (void)
3                                   {
4                      printf (”Digite un número: ”);
5                         scanf (” %d”, &pos_comp);
6                              pos_comp += 10;
7           printf (”El número digitado más 10 es: %d”, pos_comp);
8                                 return;
9                                   }
```

Supongamos que esta es una rutina del sistema y que la pueden utilizar varios diferentes procesos (en cualquier momento), la rutina solamente lee desde la entrada estandar un número, el cual almacena en la variable global *pos_comp*, esta variable es una posición compartida en la memoria y todo proceso que llame a esta rutina puede leer y escribir en ella. Ahora bien, un proceso **X** llama a la rutina pero por alguna razón la CPU detiene su ejecución después de haber finalizado la instrucción de la línea 5, por lo cual la variable ya ha tomado un nuevo valor (supongamos 7) que ha sido el introducido por el usuario, cuando el proceso **X** aún se encuentra bloqueado, el proceso **Y** es lanzado, este proceso también hace uso de la rutina «ejemplo», lee un nuevo valor (supongamos 9) y continua su ejecución normal hasta el fin de la rutina, por

tanto debe mostrar un resultado de 19, hasta este punto todo es completamente normal, sin embargo, tiempo después **X** reanuda su ejecución desde el punto siguiente al que fue interrumpido (línea 6), como debe hacer uso de *pos_comp* lo pide a la memoría y continua su ejecución normal hasta finalizar la rutina, es de esperarse que como había introducido un 7, el resultado sea 17, pero no hemos tenido en cuenta algo: *pos_comp* fue utilizada por **Y** y este modificó su valor y a ese momento vale 19, por lo cual el verdadero resultado que nos mostrará **X** es 29 ya que ese es el valor de 19 + 10.

Este sencillo ejemplo nos demuestra como la concurrencia se puede presentar (y de hecho lo hace bastante seguido) en cualquier instante y como puede afectar significativamente la operación de la máquina. Claro que el ejemplo anterior es tan sólo una de las formas en que la concurrencia afecta la interacción entre procesos, pero esta no es la única, de forma general es posible distinguir tres tipos de interacción interprocesos:

- Competencia
- Cooperación por compartición
- Cooperación por comunicación

Grado de Conocimiento	**Relación**	**Influencia de un proceso en los otros**	**Posibles Problemas de Control**

Grado de Conocimiento	**Relación**	**Influencia de un proceso en los otros**	**Posibles Problemas de Control**
Los procesos no tienen conocimiento de las demás	Competencia	❋ Los resultados de un proceso son independientes de las acciones de los otros ❋ Los tiempos de los procesos pueden verse afectados	❋ Exclusión mutua ❋ Interbloqueo (recursos renovables) ❋ Inanición
Los procesos tienen conocimiento indirecto de los otros (p.e. objetos compartidos)	Cooperación por compartición	❋ Los resultados de un proceso pueden depender de la información obtenida de los otros ❋ Los tiempos de los procesos pueden verse afectados	❋ Exclusión mutua ❋ Interbloqueo (recursos renovables) ❋ Inanición ❋ Coherencia de datos

Grado de Conocimiento	**Relación**	**Influencia de un proceso en los otros**	**Posibles Problemas de Control**
Los procesos tienen conocimiento directo de los otros (hay disponibles unas primitivas de comunicaciones)	Cooperación por comunicación	❋ Los resultados de un proceso pueden depender de la información obtenida de los otros ❋ Los tiempos de los procesos pueden verse afectados	❋ Interbloqueo (recursos consumibles) ❋ Inanición

Cuadro 3.1: Interacción entre procesos

Estos tipos de interacción se producen dependiendo del grado o nivel de conocimiento que un proceso tiene de otro, la tabla 3.1 tomada de [STALLINGS-97] nos presenta de forma resumida aunque bastante informativa los tipos de relaciones interprocesos dependiendo de los grados de conocimiento, así mismo explica también la forma en que esta relación influye en otros procesos y además algunos de los problemas que puede presentar.

Como podemos observar en este cuadro la competencia entre procesos nos plantea tres tipos diferentes de problemas:

El primero de ellos se basa en situaciones como la del ejemplo de la suma (figura 3.2), en estos casos es necesario que el sistema provea una forma de **exclusión mutua**, es decir garantizar que un proceso que utilice un recurso

compartido tenga la seguridad que otro no lo ha modificado durante su ejecución.

La exclusión mutua desafortunadamente trae consigo otro gran problema, el **interbloqueo**, este se da cuando dos o más procesos requieren acceso a dos o más recursos compartidos, el problema se puede presentar en el caso que el sistema operativo asigne a cada proceso un recurso, pero dicho proceso necesite a su vez del recurso que tiene el otro proceso (y viceversa) por lo cual ambos procesos son bloqueados a la espera de que se libere el recurso que necesita cada uno, por lo cual ambos procesos se bloquean permanentemente ya que ninguno de los dos puede terminar hasta no utilizar el recurso faltante y por este motivo el recurso ya asignado no puede ser liberado.

El tercer problema es la **inanición**, esta se presenta cuando un proceso permanece en estado de bloqueo permanente a la espera de un recurso que necesita, pero por algún motivo externo al proceso solicitante, este nunca logra que le sea asignado (por lo general por que dos o más procesos se pasan mutuamente el control de dicho recurso por lo que siempre estará ocupado).

Estos tres problemas se dan también cuando los procesos no compiten entre sí por el recurso si no que por el contrario deben cooperar entre ellos (sin tener conocimiento exacto de quién es el otro y que hace) ya que son conscientes de que no son los únicos que hacen uso de este, para el caso de la exclusión mutua se debe aclarar que sólo se debe hacer uso de ella en el caso que los datos compartidos deban ser modificados. Adicionalmente en la cooperación por compartición se debe también garantizar la **coherencia de los datos**, ya que puede darse el caso de que a pesar que cuando se modifica un dato nadie más puede modificarlo (por la exclusión mutua) hasta tanto no finalicen las instrucciones que hacen uso de este, es posible que se requiera de algún otro dato (o datos) con los cuales se trabaje en diversas partes del proceso en conjunto con el que tiene la garantía de la exlusión mutua, pero puede suceder que entre cada uso de la información del dato esta pueda variar, por lo tanto puede presentarse el caso de que aún habiendo exlusión mutua, el resultado de la ejecución puede variar su comportamiento normal. Para solucionar esto se utiliza la **atomici-**

dad de las instrucciones, es decir no pueden presentarse cambios mientras un conjunto de instrucciones (declaradas como atómicas) se esté ejecutando, este tipo de situaciones se presenta mucho en aplicaciones como bases de datos.

3.2.1. Exclusión mútua

El principio de exclusión mutua es un mecanismo empleado en el diseño de los sistemas operativos para evitar los problemas de competencia por recursos, se basa en definir una zona o región crítica la cual está marcada por las instrucciones que hacen uso del recurso o recursos por el que se presenta la competencia (recurso crítico), existen diferentes métodos de aplicación de la exclusión mutua tanto por *hardware* como por *software*, dentro de las soluciones por software se destacan el algoritmo de *Dekker* y el algoritmo de *Peterson*, sin embargo su explicación se deja para consulta en textos más avanzados según criterio del docente, aunque exhortando a hablar sobre algunos (s) de los problemas clásicos que se utilizan para ilustrar las diferentes soluciones.

3.2.2. Interbloqueo

Este problema se presenta cuando dos o más procesos se bloquean mutuamente a la espera de un recurso crítico y esta situación se mantiene de manera permanente, no existen como en la exclusión mutua grandes soluciones que permitan disolver un interbloqueo, por el contrario se deben utilizar diversas técnicas para prevenir este tipo de situaciones.

3.2.3. Inanición

La inanición se presenta cuando un proceso nunca lograr acceder a un recurso crítico, y por tanto no pudiendo continuar con su normal ejecución.

Capítulo 4

Comparativa de Sistemas Operativos

Hacer una verdadera comparativa entre un sistema operativo y otro es un proceso muy delicado, en especial si se trata de ser lo más imparcial posible y a la vez tratar de abarcar los requerimientos que la mayoría de usuarios buscan de su plataforma, teniendo en cuenta esta reflexión lo primero que es necesario preguntarse antes de empezar es ¿cuales son las bases para la comparación?, la respuesta a esta pregunta ha sido dividida en dos aspectos: administrativo y técnico. A nivel administrativo se detectó que la mayoría de usuarios sin conocimientos técnicos acerca del funcionamiento de los sistemas operativos evaluan 4 items:

1. Esquema Licenciamiento y Precio
2. Estabilidad y Desempeño
3. Facilidad de uso
4. Soporte

Por lo general estos son los principales aspectos que un administrador de sistemas de información tiene en cuenta al momento de presentar una evaluación a usuarios no técnicos (gerentes, jefes de división, etc) cuado se trata de tomar una desición en torno a cual plataforma adoptar para una empresa o proyecto.

A nivel técnico los tópicos que se han evaluado son:

1. Compatibilidad con Otras Plataformas
2. Portabilidad
3. Requerimientos Hardware

Aunque a nivel técnico se encuentran gran cantidad de factores de más que no se incluyen en esta revisión es necesario aclarar que es tan sólo un caso particular y que procura ser muy genérico, cada compañía (e incluso persona) puede tener una perspectiva diferente en cuanto a la evaluación.

Ahora que se han explicado los componentes de la comparativa, es indispensable enumerar a su vez los «sujetos de estudio», es decir los sistemas operativos sobre los que se pretende hacer este estudio, cabe destacar que al igual que los anteriores aspectos, estos también tienen la característica de la generalidad, por lo cual sólo se ha escogido una pequeña muestra de la gran cantidad disponible, estos son:

1. FreeBSD™
2. GNU/Linux
3. Mac OS® X
4. NetBSD
5. OpenBSD
6. Windows® 98

7. Windows® 2000

8. Windows® XP

A continuación se dará una breve descripción de cada uno de estos sistemas, con el objetivo de dar primero una visión general, después de esto se pasará a explicar cada uno de los tópicos de la comparación y dentro de estas se pasará de la generalidad a la especificidad.

4.1. Sistemas Operativos

4.1.1. FreeBSD

http://www.freebsd.org

FreeBSD es un sistema operativo derivado del 386 BSD, es un sistema operativo libre (y gratuito) creado por cientos de desarrolladores, es altamente usado como servidor de Internet debido a sus altas prestaciones en comunicaciones,

4.1.2. GNU/Linux

Aunque se hablará más en profundidad acerca de este sistema operativo en la segunda parte del libro, cabe destacar que es un sistema libre desarrollado por miles de personas a través de Internet, y que es el sistema operativo de mayor crecicmiento en la actualidad, y el segundo de mayor uso en el mundo (tomando como uno sólo a toda la gama de la familia Windows.

4.1.3. Mac OS X

http://www.apple.com/macosx/

Es un sistema operativo desarrollado por Apple Computer Inc., es una reescritura practicamente completa de su sistema operativo Macintosh®, este nuevo sistema Mac está basado en el sistema Darwin http://developer.apple.com/darwin/, un proyecto «OpenSource» (aunque esta es la definición que da Apple, prefiero utlizar el termino «Libre»), el cual utiliza caracteristicas de otros sistemas UNIX como Mach, FreeBSD y otros, la idea detrás de Mac OS X y por supuesto de Darwin es crear un completo sistema operativo con la flexibilidad y robustez de un UNIX y la facilidad de uso que siempre a caracterisado a los MAC.

4.1.4. NetBSD

http://www.netbsd.org

NetBSD es tal vez el sistema operativo más portado en el mundo, es otro decendiente de 4.4 BSD y 386 BSD, por lo cual también se distribuye bajo los términos de la licencia BSD[1] , lo que implica que puede ser libremente distribuido en forma binaria o de código fuente. El principal objetivo de NetBSD es la portabilidad, obviamente sin descuidar seguridad y estabilidad como la mayoría de derivados UNIX.

[1]Un ejemplo de esta licencia se puede encontrar en: http://www.opensource.org/licenses/bsd-license.php

4.1.5. OpenBSD

http://www.openbsd.org

Este sistema operativo es derivado de NetBSD, sus principales metas son la seguridad, la estandarización y la portabilidad, está catalogado como el sistema operativo más seguro del mundo (aunque en téminos de seguridad no se puede hablar de una verdad absoluta).

4.1.6. Windows 98

http://www.microsoft.com/windows98/

Este es uno de los sistemas operativos de escritorio con mayor cantidad de usuarios en el mundo, ya que se encuentra a mitad de camino (en términos de tiempo) entre la aparición de Windows 95 y de Windows 2000, entre sus principales características se encuentra el soporte real a multitarea (en Win95 era simulado).

4.1.7. Windows 2000

http://www.microsoft.com/windows2000/

Win2000 es la nueva generación del sistema operativo para redes de esta empresa (Windows NT), constituyendose en una mejora significativa desde varias perspectivas, entre las que cabe destacar la estabilidad.

4.1.8. Windows XP

http://www.microsoft.com/windowsxp

Este sistema operativo hace uso del nuevo «motor de sistema» que Microsoft desarrollló para Windows 2000, por lo tanto integra altas prestaciones gráficas junto a carcateristicas de trabajo corporativo (en redes) heredadas de Win2000.

4.2. Comparación a nivel administrativo

4.2.1. Esquema Licenciamiento

Cuando se habla de esquema de licenciamiento se hace referencia al acuerdo que se «firma» entre las dos partes: productor y usuario del software, y que por lo general el segundo descuida (la mayoría de las veces se limita a aceptar el acuerdo sin siquiera leerlo brevemente) pero que constituye un salvamento para el primero. Se encuentra muy ligado al precio (a pesar de ser dos conceptos diferentes) debido a que por lo general lo que el usuario compra son «licencias», los esquemas de licenciamiento son muy diversos y por lo general varian sustancialmente entre cada compañía productora. Antes de entrar en materia con la evaluación es importante aclarar que los precios que se incluyen en esta muestra pueden variar según el mercado e incluso dependiendo de la versión o fecha de lanzamiento del sistema. A continuación se dará un vistazo a los esquemas de licenciamiento de cada uno de nuestros sistemas objetos de estudio.

4.2.1.1. FreeBSD

Como ya se dijo inicialmente, este sistema operativo es libre, su *kernel* está licenciado bajo la licencia BSD al igual que gran cantidad de sus aplicaciones, en especial aquellas de integración con el sistema o configuración, sin embargo como la mayoría de UNIX utiliza algunas de las herramientas generadas por el proyecto GNU, por lo cual algunas porciones de FreeBSD estan licenciadas por la GPL. En cuanto al precio, FreeBSD se puede conseguir de forma gratuita

descargandolo desde Internet, sin embargo también es posible (ninguna de las dos licencias lo impide) encontrarlo por algún costo, por lo regular bastante bajo comparado a otros sistemas operativos.

4.2.1.2. GNU/Linux

El sistema operativo GNU/Linux se encuentra liberado y protegido a su vez por la licencia pública general o GPL `http://www.gnu.org/copyleft/gpl.html`[2], la cual se destaca por ofrecer por ofrecer las siguientes libertades básicas:

1. Libertad para ejecutar el programa, con cualquier propósito.
2. Libertad para modificar el programa y adaptarlo a sus necesidades. (Para que esta libertad sea efectiva en la práctica, es necesario disponer del código fuente, porque modificar un programa sin disponer del código fuente es extraordinariamente dificil.)
3. Libertad para redistribuir copias, **tanto gratis como por un cánon**.
4. Libertad para distribuir versiones modificadas del programa, de tal manera que la comunidad pueda beneficiarse con sus mejoras.

En cuanto al precio, GNU/Linux se encuentra disponible en Internet sin costo alguno, sin embargo incluso los propios fabricantes de las diferentes distribuciones (para comprender mejor este concepto se puede leer la sección 5.2 en la segunda parte de este libro) o cualquier persona o empresa puede cobrar por el sistema, los precios pueden ser tan dispares como por ejemplo: USD$ 18 por el conjunto de 7 CDs de Debian 3.0 r1, USD$ 69 por el *powerpack de Mandrake* 9.2 (7 cds, 1 manual), USD$ 200 el *ProSuite* de Mandrake 9.2 (9 CDs, 1 DVD, 2 manuales, incluyendo también la versión para este S.O. de la base de

[2]Se puede encontrar una traducción al español de la GPL en la dirección: `http://gugs.sindominio.net/gnu-gpl/gples.html`

datos comercial IBM DB2 8.1, además de soporte telefónico), o incluso USD$ 750 por el *SUSE LINUX Enterprise Server* (cabe aclarar que también incluye software propietario y soporte).

4.2.1.3. Mac OS X

El licenciamiento en el Mac OS X merece especial atención ya que este sistema operativo incluye un esquema tanto libre como propietario, ckmo ya se comentó, este sistema tiene su base fundamental (*core*) en el sistema libre Darwin, el cual es liberado bajo la licencia *Apple Public Source License* (APSL `http://www.opensource.apple.com/apsl/`) la cual en su versión 2.0 ha sido aprobada por la *Free Software Foundation* `http://www.fsf.org/fsf/fsf.es.html` como una licencia de software libre. Sin embargo como se mencionó ya, también tiene compponentes propietarios, como por ejemplo la interfaz gráfica Aqua (introducidad originalmente con la aparición de los iMac), por lo cual su esquema de licencia también es propietario, el Mac OS X *Phanter* (la última versión hasta este momento) tiene dos alternativas de compra, 1 usuario (es decir sólo se puede instalar en un computador) por USD$ 129, y el *family pack* para 5 usuarios por USD$ 199.

4.2.1.4. NetBSD

Este sistema operativo por ser otra variante de BSD y derivado directo de 4.3 BSD se encentra liberado bajo la licencia BSD, esto implica obviamente que se garantiza su libre redistribución para cualquier fin, bien sea de forma gratuita o mediante el cobro de un cánon.

4.2.1.5. OpenBSD

Esta es la tercera variante directa (por que Darwin es indirecta) de los BSD, al igual que los anteriores es software libre y se rige por la licencia BSD, la cual

garantiza su libre redistribución, aunque sujeto a los siguientes lineamientos genreales:

- La redistribución debe mantener cualquier nota de Copyright del original, preservando así la propiedad intelectual.
- El software regido por esta licencia se entrega sin ningún tipo de garantia.

4.2.1.6. Windows 98

El esquema de licenciamiento de la empresa responsable de este sistema operativo ofrece varias alternativas desde la licencia «OEM» para los computadores con este sistema preinstalado, hasta licencias de tipo «*campus agrement*» es decir licencias en «masa» cuando se requiere licenciar un número significativo de máquinas. Sin embargo ya no es posible de forma oficial conseguir este sistema operativo ya que su productor a decidido (unilateralmente) dejar de venderlo, tan sólo es posible conseguirlo a través de terceros que aún tengan remanentes.

4.2.1.7. Windows 2000

Por pertenecer a la misma familia de sistemas operativos el esquema de licenciamiento es similar al de Win98, aunque para el caso de esta versión es aún un poco más complicado, ya que no sólo se debe tener en cuenta el número de máquinas a licenciar, sino también el número de procesadores en cada máquina, en especial si pasa de 2, ya que la licencia (y el mismo programa) deben ser «versiones especiales».

Windows 2000 es ofrecido en 4 diferentes versiones:

- *Professional*

- *Server*
- *Advanced Server*
- *Datacenter Server* (esta versión es únicamente disponible para distribución con equipos nuevos y sólo para distribuidores autorizados, por lo cual no se tendrá en cuenta durante la evaluación).

Cada una de estas posee características que la hacen más adecuada a determinadas necesidades, la siguiente tabla resume el propósito geeral de la versión y su precio.

Professional	Es la versión para equipos de escritorio, provee únicamente funcionalidades básicas.	USD$ 319
Versión	Descripción	Precio Unitario
Server	Provee funcionalidades de servidor de archivos e impresión, así como capacidades de servidor Web.	USD$ 999
Advanced Server	Además de las capacidades de Win 2000 Server provee servicios de alta disponibilidad.	USD$ 3999

Cuadro 4.1: Precios Windows 2000

4.2.1.8. Windows XP

Al igual que su antecesor, Windows XP viene en dos versiones y obviamente con dos precios diferentes:

1. *Home Edition*: Es una versión orientada al sector familiar, su precio es de USD$ 199.
2. *Professional*: Esta incluye mayores capacidades de seguridad y está orientada al sector empresarial, su precio es de USD$ 299.

Aunque sobra la aclaración, es importante hacer gran énfasis en que los sistemas operativos propietarios (al igual que cualquier otro tipo de aplicación con esta misma caracteristica) prohiben de forma categórica la copia, redistribución e instalación de los mismos sin que medie un contrato de licencia, por lo tanto cuando alguna persona nos convide a utilizar algún programa propietario sin el previo pago de la licencia y sin que le entregen algún tipo de certificación de la legalidad de la compra se debe pensar muy bien y ser totalmente conscientes que se está cometiendo un acto ilegal en la mayoría de paises del mundo, e incluso estamos ante un problema ético. Por esta razón y aunque suene un poco reiterativo, el autor de este libro exorta a todos los estudiantes a abandonar estas practicas que dejan mucho que desear de un profesional, y recomienda simpre actuar dentro del marco de la legalidad, obviamente si no desea verse atado a este tipo de contratos que en muchas ocaciones impiden incluso buenas prácticas como la ayuda a los amigos, se recomienda la utilización de software libre.

4.2.2. Estabilidad y Desempeño

Este tópico es bastante crítico y controvertido ya que a pesar que existen varias pruebas para medir el rendimiento de un sistema, estas por lo general se desempeñan en condiciones «generadas», es decir por lo general son conducidas

en laboratorios y con máquinas especialemnte puestas a punto para la comparación, por lo tanto no son muy fiables cuando se trata de evidenciar la realidad del usuo diario. Cuando se habla de estabilidad se hace referncia a las capacidades de la máquina de soportar la carga sin tener alteraciones que impliquen bloqueos o reinicios, lo cual se deriva en tiempo perdido e incluso posible perdidad de la información; el desempeño se refiere a la relación trabajo/tiempo, es decir a la mejor administración de los recursos del sistema logrando una ejecución rápida de las aplicaciones pero sin descuidar la estabilidad.

4.2.2.1. FreeBSD

Sin duda alguna FreeBSD es uno de los sistemas operativos más estables del mercado, obviamente esto sucede en gran parte por ser una variante de UNIX, aunque va un poco más allá. por ejemplo en las estadisticas de servidores web que realiza la consultora independiente Netcraft `http://www.netcraft.com`, los 50 sitios web que más tiempo han durado sin tener que reiniciarse estan repartidos casi equitativamente entre máquinas con sistema operativo FreeBSD y OpenBSD. En cuanto a rendimiento también podemos obsevar que una buena cantidad de sitios en Internet se encuentran soportados por este sistema operativo.

4.2.2.2. GNU/Linux

Aunque el rendimiento puede variar entre una distribución y otra y depende en gran medida del hardware sobre el que se está ejecutando, GNU/Linux en un sistema operativo bastante confiable tanto en rendimiento como en estabilidad. Debido a su gran flexibilidad es posible «personalizarlo» de multiples maneras, de tal forma que cumpla con los requerimientos necesarios, es decir si necesitamos una estación de trabajo con una buena interfaz gráfica para el hogar o la oficina, podríamos no arrancar los servidores u otras aplicaciones y dejar simplemente lo que necesitamos, si por el contrario lo que se desea es

un servidor para el Web o algún otro tipo de servicio como FTP o Bases de Datos, no es necesario que se tenga una interfáz gráfica, ya que va a consumir recursos de memoria y procesador que pueden en algún momento necesitar los servidores.

En cuanto a estabilidad como buen derivado de UNIX, GNU/Linux es altamente estable, a manera de comentario y/o evaluación personal, en los cerca de 4 años de trabajo con este sistema operativo no he obtenido más allá de 10 o 20 bloqueos de la máquina, la mayoría más a cuasa de algún tipo de «experimento» que por inestabilidad del sistema.

4.2.2.3. Mac OS X

La estabilidad en los sistemas Macintosh ha sido altamente calificada a través de su historia, ahora teniendo como base otro gran sistema operativo en terminos de estabilidad, Apple ha logrado una muy buena combinación y este nuevo producto se considera ahora suficiente competencia incluso a sistemas como GNU/Linux o FreeBSD en terminos de confiabilidad. Así mismo el trabajo a nivel de rendimiento en esta plataforma parece (según algunos usuarios) superar a las versiones anteriores de este mismo sistema operativo.

4.2.2.4. NetBSD

El desarrollo de NetBSD se caracterisa por su «limpieza» en la codificación para permitir la protabilidad, sin embargo esto a su vez trae otro beneficio añadido como lo es la estabilidad y el desempeño, la principal meta de NetBSD no es brindar cada vez cantidad de nuevas caracteristicas que puedan resultar incompatibles entre algunas plataformas, sino que por el contrario procura escribir bien y mantener un código sin mayores problemas para que su mantenimiento sea facilmente realizable.

4.2.2.5. OpenBSD

Aunque el principal propósito de OpenBSD es la seguridad, esto no implica que su rendimiento y estabilidad no sean los más adecuados, esto en gran parte se debe a que por más seguro que pueda llegar a ser un sistema operativo si no es lo suficientemente estable como para que esa seguridad extra sea utilizable, este no llegaría a ser ni medianamente utilizado; claro está que esto no pasaría de ser más allá de una conjetura si no se pudiese citar el ejemplo utilizado al describir la estabilidad de FreeBSD, es decir que entre los sitios expuestos a Internet que se caracterizan por su altisimo tiempo sin necesidades de mantenimiento, se encuentra en posición destacada OpenBSD.

4.2.2.6. Windows 98

A diferencia de los anteriores sistemas evaluados, la estabilidad es precisamente una de las grandes carencias de este sistema, casí cualquier usuario por muy poco tiempo que halla pasado frrente a una máquina operada por este sistema, habrá encontrado algún bloqueo del sistema, reinicio, o la tristemente famosa «pantala azul» indicandole que ha ocurrido algún tipo de fallo y que muy posiblemente sólo se pueda solucionar reiniciando el sistema completo. Claro que es de reconocer el adelanto que significó respecto a su antecesor (Windows 95) tanto en estabilidad como en desempeño, este último en especial al incorporar por primera vez la multitarea de forma real.

4.2.2.7. Windows 2000

Este es uno de los mejores sistemas Windows que han sido desarrollados, integraba el superior rendimiento y estabilidad de NT (frente a Win 95 y 98), con la intefaz unificada y más amigable que hizo de Win95 una novedad. Su estabilidad especialmente en las versiones Server y Advanced Server puede llegar a compararse con la de algunos UNIX, hecho que le ha permitido incluso ganar una porción del mercado de los servidores en Internet.

4.2.2.8. Windows XP

En sus primeras versiones realmente tenía varias deficiencias en cuanto a estabilidad, sin embargo en la actulidad (después del *Service Pack* 4) ha logrado mejorar ampliamente en este aspecto aunque sin lograr superar a Win 2000.

4.2.3. Facilidad de uso

Si el punto anterior de la evaluación podia ser en cierta medida subjetivo, este de la facilidad de uso si que lo es aún más, ya que lo que para algunos usuarios puede ser de gran facilidad (como una línea de comandos) para otros puede resultar absolutamente incomprensible. Por esta razón en este apartado tan sólo se comentaran algunas alternativas de las interfaces tanto de línea de comando como de interfáz gráfica. Para esta evaluación se han agrupado los 3 sistemas BSD y GNU/Linux en un sólo grupo ya que las características de las interfaces son bastante similares, de la misma manera se han agrupado las 3 versiones de Windows por la misma razón. Caso a parte es el sistema Mac OS X, ya que a pesar de ser también una variante UNIX, sus cambios a la interfaz resultan altamente significativos.

4.2.3.1. FreeBSD, NetBSD, OpenBSD, GNU/Linux

Como todo UNIX, en estos sistemas la interfáz primaria es la línea o interprete de comandos, la cual les provee gran flexibilidad, pero esta puede llegar a ser intimidante para algunos usuarios, en especial si están acostumbrados a trabajar exclusivamente con interfaces gráficas como ocurre con otros sistemas operativos; sin embargo esto no es ningún impedimento para que los *BSD y GNU/Linux puedan tenre interfeces gráficas bastante amigables y que le permitan al usuario realizar algunas de las operaciones a las que tiene acceso por el interprete de comandos. Todo UNIX tiene como motor gráfico el sistema *X-Window*, y específicamente estos tres sistemas utilizan una implementación

libre de este servidor, a la que se denomina XFree 86, por esta razón la mayoría de gestores de ventanas e incluso los dos metaproyectos de entornos de escritorio (GNOME y KDE) pueden ser utilizados en estas plataformas. Para entender un poco mejor estos conceptos es importante hacer claridad acerca del manejo de la GUI en UNIX, el gráfico 4.1 nos muestra un esquema simplificado de las capas que conforman un entorno gráfico típico para las plataformas a las que nos hemos estado refiriendo en esta sección.

Figura 4.1: Entorno Gráfico en UNIX

La primera capa (obviamente por encima del kernel y otros componentes) es el servidor X (para nuestro caso el XFree 86), este lo único que provee al sistema es una forma de soportar aplicaciones que hagan uso de las interfaces gráficas, sin embargo con el XFree lo único que podemos hacer es colocar en la pantalla el aplicativo, pero sin bordes de ninguna clase, y mucho menos la posibilidad de maximizar, minimizar o mover la ventana, para esto se utiliza una nueva capa sobre X, la que se le denomina WindowManager o Manejador de Ventanas, estos ya nos permiten hacer todas las operaciones que regularmente hacemos en cualquier sistema gráfico, e incluso algunos un poco más, ejemplos de estas aplicaciones tenemos muchos, entre otros: Window Maker, IceWM, AfterStep, Enlightenment, FluxBox, KDM, GDM, etc. por encima de los manejadores es posible colocar los entornos de escritorio GNOME (*GNU Network Object Model Environment*) o KDE (*K Desktop Environment*), estos

poseen varias características que los hacen más convenientes que los sólos manejadores de ventanas, entre las que se destaca la integración. A continuación se van a mostrar algunos pantallasos (*screenshots*) de estos sistemas operativos corriendo algunas versiones bien sea de GNOME o KDE o de Manejadores de Ventanas.

Figura 4.2: FreeBSD con GNOME

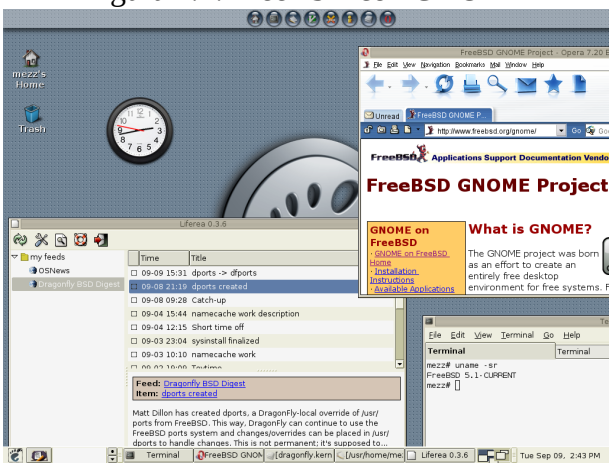


Figura 4.3: NetBSD con FluxBox

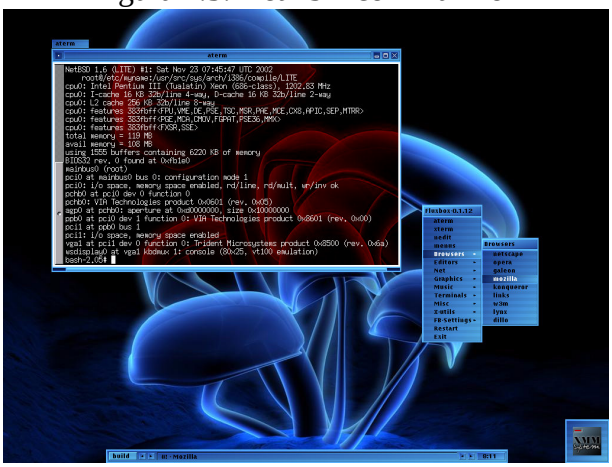


Figura 4.4: OpenBSD con KDE

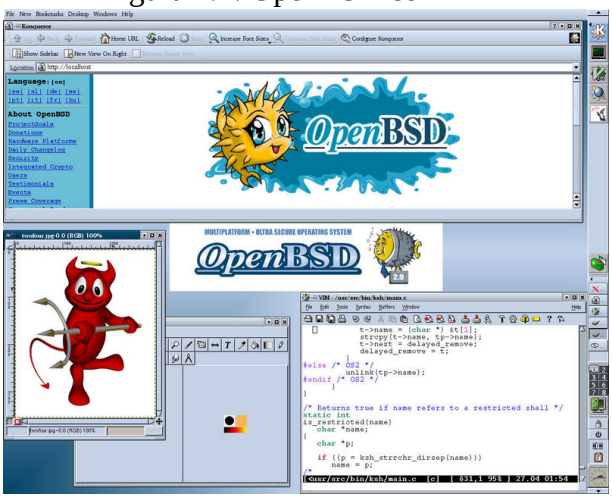


Figura 4.5: GNU/Linux con AfterStep

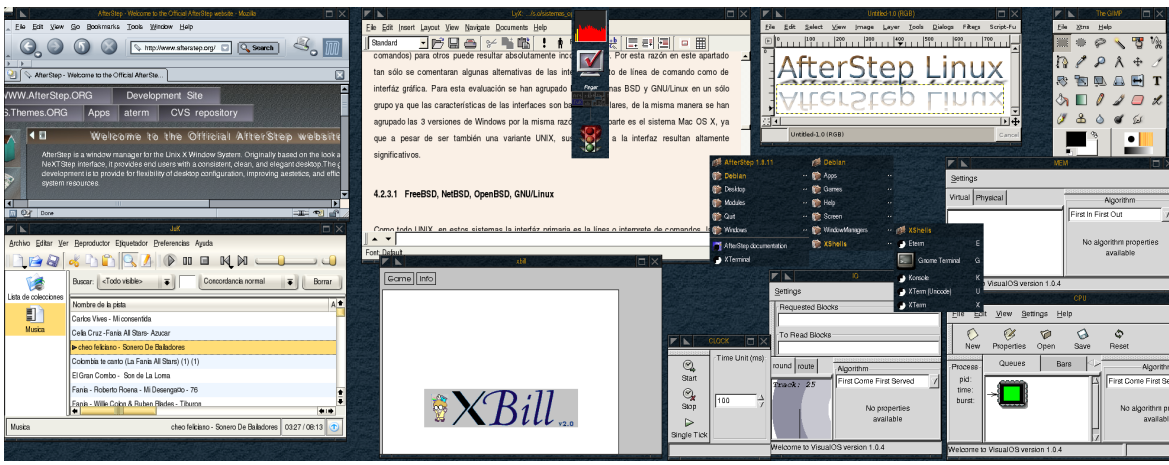


Figura 4.6: GNU/Linux con Enlightenment

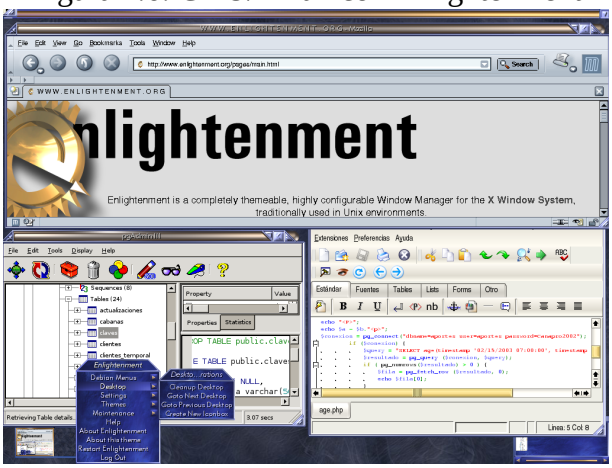


Figura 4.7: GNU/Linux con FluxBox

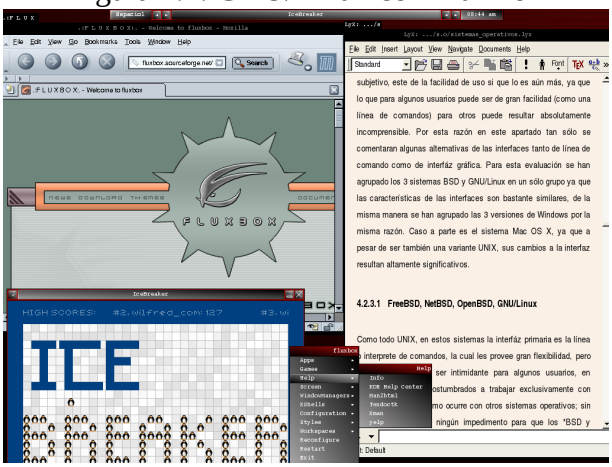


Figura 4.8: GNU/Linux con GNOME

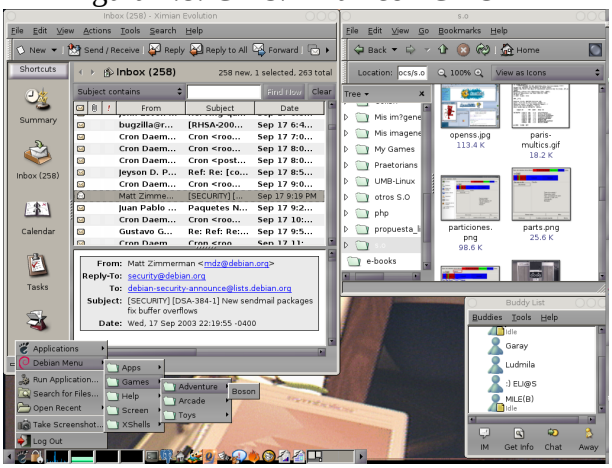


Figura 4.9: GNU/Linux con IceWM

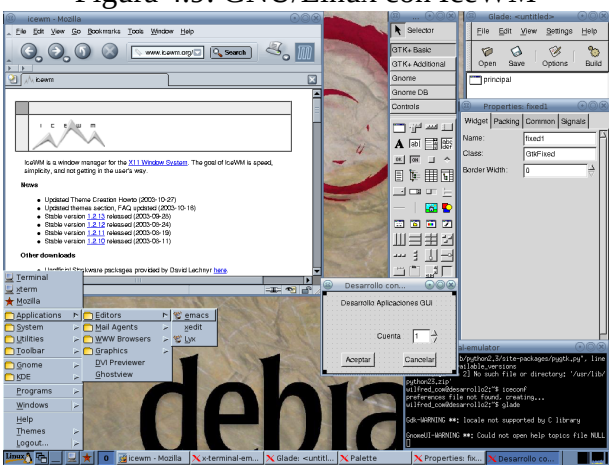
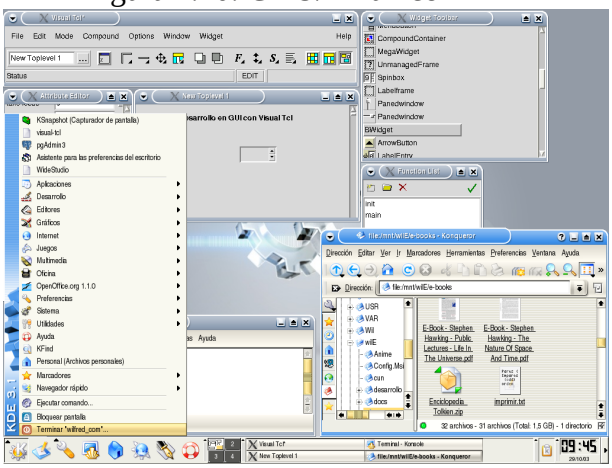


Figura 4.10: GNU/Linux con KDE



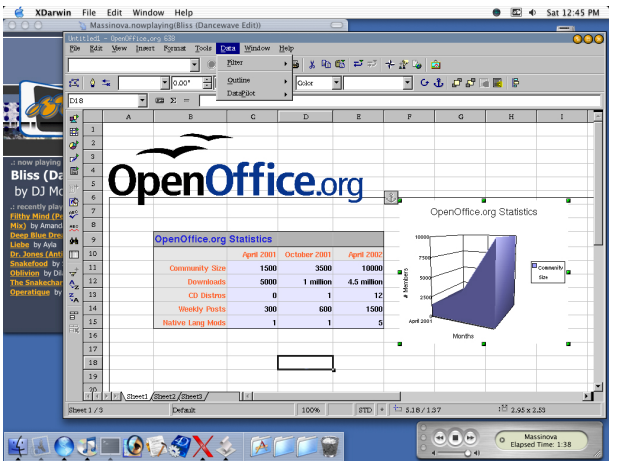
Como se puede apreciar por esta serie de panatallasos, en estos sistemas UNIX cada quien puede personalizar la interfáz para acomodarla a sus gustos o necesidades, incluso como se puede ver en los pantallasos de GNU/Linux en una misma máquina es posible tener varios entornos y trabajar en cada uno de ellos en cualquier momento.

De todo esto es posible concluir como se dijo al comienzo que la facilidad de uso es muy relativa, pero que por lo menos en estos sistemas es posible contar con varias alternativas que se ajusten a las necesidades particulares de cada usuario, por lo cual esta «facilidad» se puede grarantizar.

4.2.3.2. Mac OS X

Desde sus inicios los sistemas Macintosh se han destacado por su facilidad de uso y por su amigable interfáz gráfica, y esta última versión obviamente no es una excepción, su sistema de ventanas Aqua se integra de forma nativa con el sistema de ventanas XFree86, lo cual logra una gran apariencia gráfica que se puede apreciar en el siguiente pantallaso.

Figura 4.11: Mac OS X



4.2.3.3. Windows 98, 2000, XP

A diferencia de los UNIX la interfaz primaria para Windows es la GUI y no la línea de comandos, mientras que en UNIX la mayoría de cosas que se hacen a través del interprete de comandos es posible (aunque algunas veces de forma más complicada) hacerlas mediante la GUI, en estos sistemas por el contrario es bastante complicado (y muchas veces imposible) desempeñar la mayoría de acciones en la GUI a través de una interfaz de linea de comandos.

Para la mayoría de usuarios la interfaz más sencilla para trabajar y visualmente más atractiva es esta (aunque en mayor medida esta apreciación se debe a la falta de comocimiento de otras alternativas), la organización de la intefáz en los sistemas Windows sumado a su amplia difución hacen que para el usuario sea un poco más rápido encontrar lo que necesita ya que por tratarse de un producto bajo el control de una sóla compañía siempre se va a encontrar la misma organización de los menús en todos los sistemas. A contunuación se exponen algunos pantallasos de estos sistemas para ayudar a comprender mejor la idea de las diferencias en las interfaces entre cada uno de los sistemas evaluados:

Figura 4.12: Windows XP

Figura 4.13: Windows 2000

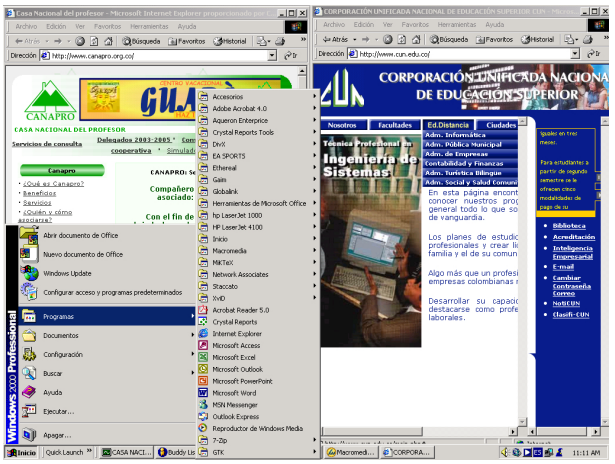
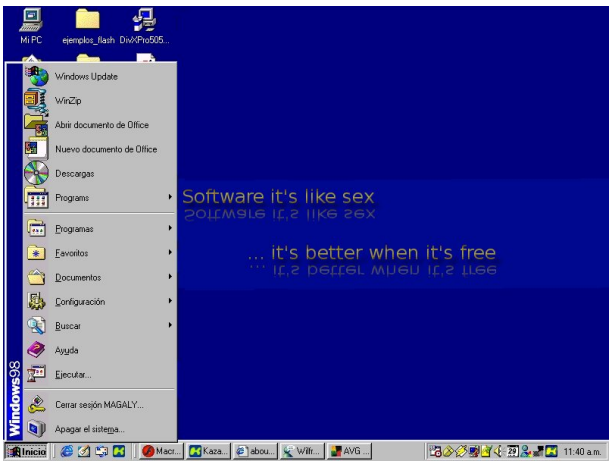


Figura 4.14: Windows 98



Observación: *Como en la versión impresa del libro no se puede apreciar con claridad estos pantallos, se recomienda verlos en su versión HTML (Web) en el CD que acompaña al texto.*

4.2.4. Soporte

Por soporte se entiende la ayuda que otra compañía o persona pueda prestar en caso de fallos o incluso ayuda para otro tipo de problemas, existen diversos tipos de soporte, siendo los más comunes el telefónico y aquel en que el técnico se traslada hasta donde se encuentra la máquina afectada. Sin embargo no siempre es necesario que el soporte se preste de esta forma, un muchas ocaciones también se puede conseguir soporte a travñes de Internet, bien sea buscando la documentación que solvente el problema o incluso consultando en foros o grupos de interes, en los cuales otras personas pueden ayudar con posibles soluciones concretas. Para el caso de esta evaluación, nuevamente se han agrupado algunos sistemas debido a que también comparten las mismas caracteristicas en cuanto a soporte técnico.

4.2.4.1. FreeBSD, NetBSD, OpenBSD

Debido a que estos sistemas operativos no estan desarrollados por una compañía que dicte sus lineamientos, el soporte acá es algo opcional en el sentido de que no hay «personal certificado» por la empresa desarrolladora para llevar a cabo esta tarea, esto aparentemente significaría que no es posible encontrar soporte técnico calificado para estos sistemas, sin embargo esto dista mucho de la realidad, ya que en todo el mundo existen cientos de empresas que se dedican a este trabajo, en los sitios Web de cada uno de estos sistemas los desarrolladores recomiendan algunas de ellas, aunque siempre es posible encontrar otras con igula o mayor calidad en su trabajo; la ventaja de ser sistemas libres permite que cualquiera pueda estudiar su funcionamiento hasta en los aspectos más profundos (cosa que no suele ocurrir con los soportes de otros sistemas donde el código es cerrado) y por tanto ofrecer en cualquier momento la ayuda más adecuada, incluso en algunas ocaciones esta mismas empresas que brindan soporte pueden tener entre sus integrantes ha personas que han participado directamente en el desarrollo y por lo tanto estan lo suficientemente calificadas como cualquier otra e incluso mucho más. Pero esta no es la única forma de encontar ayuda para la solución de problemas con estos sistemas, en Internet se encuentra gran documentación sobre como hacer las cosas, muy buenos tutoriales y en general muchas ideas que le pueden servir grandemente; así mismo también siempre es posible contactar con alguno de los muchos grupos de interés y pedir ayuda cuando se encuentren dificultades, por lo regular en poco tiempo otros usuarios que ya conocen la solución para ese problema responderan con la solución, e incluso con varias alternativas de solución; esto con lleva que a su vez quién en algún momento dado haya recibido soporte pueda en algún momento colaborarle a alguién más que lo necesite, conformando así vínculos de comunidad colaborativa.

4.2.4.2. GNU/Linux

En GNU/Linux todo lo que se mencionó en el apartado anterior es también cierto para este sistema, pero en adición a esto es de resaltar que la mayoría de distribuciones que son desarrolladas por una empresa, poseen la opción de adqurirlas con soporte técnico por lo general vía telefónica y/o por e-mail, lo cual para algunas instituciones es de carácter fundamental a la hora de decidirse por una solución de software.

4.2.4.3. Mac OS X

Apple sólo ofrece soporte a través de Internet a sus usuarios registrados, aunque también mantiene gran cantidad de documentación para que pueda ser accedida libremente, así mismo a través de está página también es posible encontrar información aceca de grupos de usuarios en todo el mundo. Además por ser fundamentalmente un derivado UNIX hay muchas cosas que se hacen de forma similar, por lo cual gran parte de la abundante documentación para sistemas operativos de este tipo también es de utilidad para el Mac OS X.

4.2.4.4. Windows 98, 2000, XP

En esta familia de plataformas aunque existen muchisimas empresas que de forma independiente brindan soporte, también existen algunas que son contratadas o certificadas directamente por el fabricante, además como esta empresa tiene representación en varios paises del mundo (incluido Colombia), también es posible establecer contacto por vía telefónica.

4.3. Comparación a nivel técnico

En la sección anterior se pudieron apreciar algunos detalles que diferencian o relacionan a los sistemas operativos de nuestra evaluación pero sin adentrarnos

en detalles que también son altamente importantes para una persona encargada de de la administración de los recursos informáticos en una empresa pero que al usuario promedio no le son de gran relevancia, en esta sección se prentenden abordar algunos de estos detalles para poder formarnos una opinión más completa de cada sistema y poder de esta forma identificar cual (o cuales) son los que mejor se adaptan a nuestras necesidades.

4.3.1. Compatibilidad con Otras Plataformas

Cuando se habla de compatibilidad con otras plataformas se deben tener en cuenta diferentes conceptos:

- **Soporte a Sistemas de Archivos diferentes al nativo**: Este concepto se asocia a la capacidad que tiene una plataforma de leer y/o escribir en sistemas de archivos de otras plataformas diferentes.
- **Ejecución de Aplicaciones compiladas para otras plataformas**: Algunos sistemas operativos son capaces de ejecutar apliaciones que fueron originalemente diseñadas para trabajar en una plataforma diferente, la mayoría de las veces esto se lleva a cabo por intermedio de otra aplicación que actua como una capa intermedia entre el sistema operativo y la aplicación no nativa, esta capa permite **emular** las llamadas al sistema y algunos otros procesos del sistema operativo para que el programa crea que se está comunicando con la plataforma para la cual fue compilado.
- **Compartir recursos en red con otras plataformas**: No se puede desconocer que el mundo actual es un mundo «conectado», es decir las redes y la comunicación entre diferentes máquinas es algo cada vez más indispensable, por lo tanto el sistema operativo debe ser capáz de comunicarse con sistemas diferentes en otras máquinas, y obviamente brindarles la posibilidad de utilizar sus recursos (por lo general archivos e impresoras, aunque realmente podría ser cualquier dispositivo).

4.3.2. Portabilidad

Cuando en sistemas operativos se habla de portabilidad esto hace referencia a la posibilidad de utilizarlos en diferentes tipos de procesadores incluso si tienen arquitecturas o diseños diferentes.

4.3.3. Requerimientos Hardware

Esto es el hardware minimo sobre el que podría ejecutarse el sistwma operativo.

Parte II

El Sistema Operativo GNU/Linux

Capítulo 5

INTRODUCCIÓN A GNU/LINUX

5.1. HISTORIA DE UNIX® Y GNU/LINUX

5.1.1. DE MULTICS A UNIX®

A principios de los años 60's la Oficina de Técnicas de Procesamiento de la Información (*IPTO - Information Processing Techniques Office*) de la Agencia de Proyectos de Investigación Avanzados (*ARPA - Advanced Research Projects Agency*)[1], *Bell Telephone Laboratories* de AT&T y *General Electrics* (en esa época G.E. era uno de los más grandes fabricantes de computadores), se unieron en un gran proyecto para desarrollar un sistema operativo de tiempo compartido, el cual derivó en la presentación de las especificaciones del nuevo sistema operativo denominado *MULTICS* (*Multiplexed Information and Computing Service*), las características presentadas constituyeron un gran avance a su tiempo en el desarrollo de sistemas operativos, sin embargo y debido precisamente al gran avance y complejidad que significaban y a pesar de los grandes esfuerzos de sus desarrolladores por varios años *MULTICS* no lograba ser totalmente funcional y resultaba además demasiado costoso para utilizarlo, por

[1] Agencia responsable de la creación de la red *ArpaNet*

este motivo en 1969 *Bell Labs* se retiró del proyecto aunque sin abandonar del todo la idea de un sistema operativo de tiempo compartido.

Ese año gracias a la financiación por parte de *Bell Labs*, *Kennneth Thompson*, *Dennis Ritchie* y *Rudd Canaday* desarrollan *UNICS* (*Uniplexed Information and Computing Service*), una versión simplificada de *MULTICS*, aunque conservando buena parte de las revolucionarias ideas que hacían a este adelantado a su tiempo (en parte debido a que algunos de los miembros del equipo también habían participado en el grupo desarrollador de MULTICS*),* el desarrollo de *UNICS* a diferencia de su antecesor rápidamente dio frutos, y a partir de 1970 (año en el que además se suma al grupo de desarrollo *Brian Kernighan)* el nuevo sistema empieza a denominarse *UNIX*TM *Time Sharing System* (Sistema de tiempo compartido) y el 3 de noviembre de 1971 el *UNIX Time Sharing System First Edition* Versión 1 ve la luz ejecutándose en una máquina DEC PDP-7.

En 1973 *Thompson* reescribe **UNIX** (que estaba escrito en ensamblador) en el lenguaje de alto nivel **C** creado por *Ritchie*, brindando la posibilidad de que el sistema se ejecute no sólo en la nueva DEC PDP-11 en la que fue recompilado, sino también en varias otras plataformas de manera relativamente sencilla, lo que sumado a la decisión de *Bell Labs* de mantener público el código fuente de su sistema y dejarlo disponible para las universidades tan sólo por el costo del medio magnético, manuales y transporte logró que para 1975 muchas universidades lo utilizaran para la investigación y la enseñanza práctica (a cambio de reportar a Bell cualquier avance investigativo relacionado con este sistema); la mejor prueba de que fue una buena idea fue la distribución de la Universidad de California en *Berkeley* (conocida como *BSD*[2]) desarrollada ese año en esta universidad bajo el liderazgo de *Bill Joy* y *Chuck Haley* y gracias al entusiasmo

[2] *BSD = Berkeley Software Dristibution*

Figura 5.1: *Dennis Ritchie* y *Ken Thompson* en una PDP-11, trabajando en *UNIX*

de sus estudiantes y profesores motivados en gran parte por la presencia de *Ken Thompson* (quién había decidido «tomarse un año sabático» en esta universidad enseñando UNIX). Uno de los mayores logros de BSD fue el que en 1980 el DoD (Department of Defense - `http://www.dod.gov`-, Departamento de la Defensa de los EE.UU) utilizara UNIX BSD 4.0 en su red *ARPANet*[3], gracias a la incorporación de los nuevos protocolos TCP/IP en esta distribución.

5.1.2. LAS VARIANTES BSD Y SISTEMA V

A partir de 1981, AT&T decidió poner fin a la distribución de fuentes con UNIX y distribuirlo bajo una licencia comercial, lo cual generó gran malestar dentro de la comunidad académica, la cual respondió dejando de utilizar UNIX para la enseñanza y continuó únicamente enseñando la teoría de los sistemas operativos, sin embargo, esta no fue la única reacción, algunas personas de

[3] *ARPANet* fue la precursora de la actual *Internet*, para conocer un poco más de esta historia puede verse «Historia de *Internet*» en [Pachón-2001]

instituciones acostumbradas a la difundida práctica de compartir el software decidieron hacer algo para remediarlo, a pesar de que esta practica de licenciar el software no era algo novedoso y para esa época ya empezaba a ser una practica difundida.

En 1983 AT&T introduce el nuevo *UNIX Sytem V* versión 1, la primera gran reforma de UNIX desde que se distribuye de forma comercial y además la primera en ser compatible con las siguientes versiones (hasta esa época un gran problema con UNIX era que entre cada versión se sucedían muchos cambios y la mayoría de las veces un programa escrito para una versión en particular no podía ser ejecutado en otra sin tener que ser compilado (y algunas veces hasta reescrito) de nuevo). Ese mismo año la universidad de California da a conocer su UNIX BSD versión 4.2, la cual no es completamente compatible a nivel binario con la versión *System V*, y a pesar que desde 1974 ya se habían empezado a desprender algunas ramas de la versión original (empezaron a surgir nuevos SOs basados en el UNIX T.S.S Ver. 4 y luego con cada siguiente versión aún más sistemas nuevos nacieron), pronto se vio que estas dos serían las grandes alternativas y cada vez que se desarrolló un nuevo UNIX hubo que decidir entre seguir el estilo *BSD* o el *System* V. Es importante destacar que cada versión tuvo grandes “hijos”, por ejemplo el sistema *SunOS* (en la actualidad se conoce como *Solaris*™ y fue desarrollado por *Sun Microsystems* (empresa fundada por *Bill Joy*)) seguía la variante BSD (así como los sistema operativos libres *NetBSD*, *OpenBSD*, *FreeBSD*), mientras que sistemas como XENIX (desarrollado por Microsoft, orientado al mercado de los PCs), *HP-UX* (de Hewlett-Packard) e incluso *Linux* han seguido la otra variante .Es importante hacer la aclaración que actualmente la mayoría (por no comprometerme diciendo que todas) de variantes de UNIX no son 100 % *System V* o *BSD*, con

el tiempo y como se ve en la siguiente sección, estas han tendido a unificarse aunque siguen teniendo en mayor o menor grado influencia de estas versiones; un ejemplo (aunque extremo) de un sistema que usa ambas variantes es *Solaris*, en la actualidad la mayoría del sistema es *System V*, pero en especial por razones de compatibilidad hacia atrás (por lo que antes era *SunOS*) incluye buena parte de las características de *BSD*.

5.1.3. EL ESTÁNDAR POSIX

Como se ha podido ver para la década de los ochenta la popularidad de lo sistemas UNIX era bastante grande, sin embargo también lo eran las diferencias e incompatibilidades entre las diferentes versiones, por lo cual los programas debían escribirse y/o compilarse para cada versión en específico, por lo cual desarrollar programas de propósito general para UNIX era bastante complicado debido a estas incompatibilidades, por este motivo a mediados de esta década el grupo */usr/group* (organización conformada por usuarios de los diferentes sistemas) cuyo primordial objetivo era la compatibilidad de las aplicaciones con los diferentes sistemas, publica en 1986 un estándar denominado *POSIX*[4] (*Portable Operating System Interface*, Interfaz de Sistema Operativo Portable) con el que se pretende poner fin a la falta de portabilidad entre los UNIX, aunque a pesar de que el estándar tuvo buena acogida no consiguió del todo su propósito; sin embargo *POSIX* no fue el único intento de estandarización, en 1984 se fundó la compañía *X/Open* Ltda., organización sin animo de lucro conformada por varias de las compañías vendedoras de UNIX más influyentes en el mundo (entre otras AT&T, HP, IBM y Sun), la cual desarrolló también un estándar UNIX conocido como *X/Open*. En la actualidad estos dos esfuerzos han sido unificados por la IEEE (*Institute of Electrical and Electronics Engineers* - `http://www.ieee.org`-, Instituto de Ingenierías Eléctrica y Electrónica) bajo la vigilancia del *PASC* (Portable Application Standards Committee - `http://www.pasc.org`-, Comité de Estándares de Aplicaciones Portables) y

[4] «El nombre fue sugerido por *Richard Stallman* y se recomienda pronunciarlo pahz-icks» [PASC]

Figura 5.2: *Richard Stallman* en el *Linux World* (China, Agos/2000)

representan no un sólo estándar sino una familia de estos procurando la portabilidad de las aplicaciones en los sistemas abiertos.

5.1.4. EL PROYECTO GNU

Como ya se comentó anteriormente, no todo el mundo estuvo muy de acuerdo con la ”comercialización” de UNIX, y el creciente auge de la practica de no entregar el código fuente con las aplicaciones (o el SO) , y algunos decidieron hacer algo para solucionarlo.

A comienzos de la década de los 80’s ***Richard M. Stallman*** trabajaba como programador de sistemas operativos en el Laboratorio de Inteligencia Artificial del MIT (*Massachussetts Institute of Tecnology* - `http://www.mit.edu`- , Instituto de Tecnología de *Massachussetts*), pero pronto se vio enfrentado como él mismo dice a una “elección moral severa”, ya que el MIT había empezado a adquirir nuevas máquinas que ya traían incorporado su propio sistema operativo (propietario), esto significó que el grupo de *hackers*[5] del laborato-

[5]Es importante aclarar que en este libro la palabra *hacker* hace referencia a su definición correcta, es decir se utiliza para referirse a la persona o personas que disfrutan usar los equipos de computo y sacar el máximo provecho de estos, y no a la definición errada que presentan los medios de comunicación y que en general conocemos, me refiero a la persona o personas que acceden de forma ilegal a los recursos informáticos de terceros. Ver [RAYMOND-03]

rio ya no podía desarrollar sus propios sistemas o modificar los existentes dependiendo de las necesidades, las palabras de Stallman resumen muy bien la situación y su decisión:

> «Al desaparecer mi comunidad, se hizo imposible continuar como antes. En lugar de ello, me enfrenté a una elección moral severa.
>
> »La elección fácil era unirme al mundo del software propietario, firmar los acuerdos de no revelar, y prometer que no iría en ayuda de mi amigo *hacker*. Es muy probable que desarrollara software que se entregaría bajo acuerdos de no revelar y de esa manera incrementara también las presiones sobre otra gente para que traicionen a sus compañeros.
>
> »Podría haber hecho dinero de esta manera, y tal vez me hubiese divertido escribiendo código. Pero sabía que al final de mi carrera, al mirar atrás a los años construyendo paredes para dividir a la gente, sentiría que usé mi vida para empeorar el mundo.
>
> »Ya había estado del lado en que se reciben los acuerdos de no revelar, por experiencia propia, cuando alguien se negó a entregarme, a mí y al Laboratorio de IA del MIT, el código fuente del programa de control de nuestra impresora. (La ausencia de ciertas características en este programa hacía que el uso de la impresora fuera frustrante en extremo.) Así que no podía decirme a mí mismo que los acuerdos de no revelar son inocentes. Me enojó mucho cuando él se negó a compartir con nosotros; no podía ahora cambiarme de lugar y hacerle lo mismo a todos los demás.
>
> »Otra elección, fácil pero dolorosa, era abandonar el campo de la computación. De esta manera no se usarían mis habilidades para mal, pero aún así se desperdiciarían. Yo no sería culpable por dividir y restringir a los usuarios de computadoras, pero ello sucedería igual.
>
> »Así que busqué la manera en la cual un programador podría

> hacer algo para bien. Me pregunté: ¿habrá algún programa o programas que yo pueda escribir, de tal manera de otra vez hacer posible una comunidad?
>
> »La respuesta era clara: lo primero que se necesitaba era un sistema operativo. Este es el software crucial para empezar a usar un computadora. Con un sistema operativo usted puede hacer muchas cosas; sin uno, ni siquiera puede funcionar la computadora. Con un sistema operativo libre, podríamos tener de nuevo una comunidad de hackers cooperando–e invitar a cualquiera a unírsenos. Y cualquiera sería capaz de utilizar una computadora sin que de movida conspire a favor de la privación de sus amigas o amigos.» [Stallman-1998]

Debido a esta decisión en enero de 1984 Stallman comenzó el **Proyecto GNU**[6], el cual buscaba crear un sistema operativo **libre**, por ser UNIX el sistema más utilizado en esa época, en especial por la gente por la gente que conocía de computación, la decisión fue que GNU debería ser similar a UNIX y totalmente compatible con este, y obviamente debería tener todos los componentes que cualquier sistema operativo serio poseía (Editor de textos, Compilador, Encadenador, etc). En poco tiempo debido a la popularidad alcanzada por aplicaciones como EMACS y GCC desarrolladas por *Stallman* al inicio del proyecto, muchos programadores alrededor del mundo se sumaron a la causa y de esta forma en pocos años GNU logró tener prácticamente todas las herramientas necesarias para convertirse en un sistema operativo completo, a excepción del componente más importante: el *kernel*, desde hacía tiempo GNU venía trabajando en un micro-kernel derivado de *Mach*, denominado *Hurd*, desafortunadamente por las mismas especiales características de complejidad de un núcleo de este tipo, *Hurd* no estuvo listo en mucho tiempo, por lo cual las herramientas GNU sólo se podían utilizar en otros sistemas operativos, pero esta situación no duró mucho tiempo.

[6]GNU es un acrónimo recursivo para decir **GNU no es UNIX**, *GNU* es el término inglés que traduce Ñu (el antílope africano, un gran símbolo de libertad)

Figura 5.3: *Linus Torvalds*

5.1.5. LINUX

En 1991 el joven finlandés *Linus Benedict Torvalds* comenzó a desarrollar un sistema operativo inspirado en ***Minix*** (sistema clónico de UNIX desarrollado por el profesor *Andrew S. Tanenbaum* en 1987 como complemento práctico de su libro *Operating Systems: Design and Implementation*, Minix era distribuido bajo una licencia libre, y por este motivo muy utilizado para la enseñanza práctica de los sistemas operativos) aunque como él mismo dice simplemente como un hobby:

> «Hola a todos los que allá afuera están usando minix – Estoy haciendo un sistema operativo (gratuito[7], sólo como un hobby, no tan grande y profesional como gnu) para clones AT 386(486). Comencé en abril y está empezando a ser utilizable. Me gustaría retroalimentación de cosas que a la gente le guste / disguste en minix, como mi SO se asemeja en cierta forma (igual capa física del

» [7]Se ha empleado la palabra gratis en la traducción a pesar de que existen opiniones acerca de que se debería usar Libre, ya que a pesar de que el propio *Linus* no ha hecho gran claridad sobre lo que quiso decir, ha dejado entrever que no estaba pensando precisamente en libertad.

sistema de archivos (debido a razones prácticas) entre otras cosas)

»[...] Me gustaría conocer que características busca la mayoría de la gente. Cualquier sugerencia es bienvenida, pero no prometo que la implementaré :-).

»Linus (torvalds@kruuna.helsinki.fi) ...» [Torvalds-1991] [TORVALDS-91]

En poco tiempo y gracias a la ayuda de muchas personas entre otras el propio Tanenbaum (con quién sostuvo acaloradas discusiones sobre varios puntos del desarrollo , siendo la más comentada la conveniencia de un kernel de tipo modular (opinión de Tanenbaum) o monolítico (opinión de Torvalds)[8] Linus pudo desarrollar un núcleo al que denominó ***Linux***, cuando Linux empezó a ser funcional alrededor de 1992, la decisión fue obvia, GNU poseía todas las herramientas de un sistema operativo menos el *kernel* y Linus había desarrollado uno que sin ser el óptimo cumplía con las necesidades del proyecto, el resultado: ***GNU/Linux***.

5.1.6. GNU/LINUX

Al comienzo la utilización de GNU/Linux resultaba bastante complicada aún teniendo buenos conocimientos de informática debido a que era necesario descargar y compilar cada paquete de forma independiente, debido a esta complejidad algunas personas como *Patrick Volkerding* (creador de *Slackware*, una de las primeras distribuciones) decidieron inventar una forma de hacer esto más sencillo creando el concepto de distribución, una distribución agrupaba en medios magnético diversos paquetes y brindaba un sistema de instalación relativamente más sencillo, con el tiempo algunas otras personas como *Bob Young* y *Marc Ewing* (fundadores de *RedHat*) adoptaron esta idea y pronto comenzó una guerra de distribuciones, en la cual todos ganaban pero el mayor beneficiario fue el usuario final.

[8] La discusión completa se puede ver en `http://groups.google.com/groups?threadm=12595`

5.2. DISTRIBUCIONES GNU/LINUX

Una distribución GNU/Linu*x* es un conjunto conformado por el *kernel Linux*, aplicaciones del proyecto GNU y un sistema de empaquetado e instalación, esto permite de una manera sencilla instalar y "poner a punto" una máquina con este sistema operativo.

Existen muchas veces grandes diferencias entre las distribuciones entre otras la forma de organizar los archivos (aunque poco a poco se está logrando una estandarización) y/o el sistema de paquetes para la instalación de software (o la ausencia de este en algunos casos), en el mercado existen muchas distribuciones cada una con un énfasis o características especiales que la hacen más recomendable para un usuario u otro, sin embargo existen algunas de propósito general y de reconocida trayectoria o popularidad en el mundo, para un listado "actualizado" de algunas distribuciones populares disponibles se puede visitar la dirección `http://www.linux.org/dist/index.html`, en este sitio es posible buscar por varios parámetros para encontrar una que se ajuste a nuestra necesidad. En las secciones de la 5.2.1 hasta la **??** se puede encontrar una breve descripción de algunas de las más representativas[9].

5.2.1. DEBIAN

`http://www.debian.org`

Es la única distribución GNU/Linux totalmente libre, su desarrollo a diferencia de las demás no está soportado por una empresa comercial, Debian es desarrollado por una comunidad organizada a través de Internet (en la que cualquiera

[9]Es importante resaltar que la información acá presentada es una apreciación personal fruto de la experiencia con cada distribución o consultas en su sitio web, parte de esta información (en especial lo que se refiere a forma de distribución) esta sujeta a cambios, en parte debido a una de las principales características de este sistema operativo: la permanente actualización e innovación, por lo cual cuando se dice "actual" se hace referencia a lo que era en el momento de escribir el libro.

que lo desee y demuestre la suficiente capacidad puede participar) y que tiene como objetivo común seguir el **contrato social** ([DEBIAN-03]), el cual entre otras garantiza que todos los paquetes de la distribución sean 100 % software libre (excepto los que pertenecen al grupo *non-free* (no libre), un grupo de paquetes que sin ser enteramente software libre sus condiciones de distribución permiten su uso sin mayores restricciones y son mantenidos en la distribución pero bajo la salvedad de que se está trabajando con software no libre).

El proyecto Debian nació el 16 de agosto de 1993 como una iniciativa de *Ian Murdock* de desarrollar una distribución que se basara en el espíritu de libertad de *Linux* y *GNU*, lo cual logró gracias a la desinteresada ayuda de otros *hackers* y más adelante con el apoyo de la *Free Software Foundation* (Fundación para el Software Libre (entidad encargada de la protección y expansión del software libre, y además responsable del Proyecto GNU -`http://www.fsf.org`-)), la primera versión disponible de Debian fue la 0.01 (liberada en agosto de 1993). Debian se consigue en 3 versiones diferentes según las necesidades de los usuarios:

1. Estable (*Stable*): Sólo hacen parte de esta los paquetes (y las versiones de estos) que han sido extensamente probados en las otras dos versiones, tratando de garantizar el máximo de estabilidad y seguridad (es la más recomendable para servidores)

2. De prueba (*Test*): En esta versión se prueban todos los paquetes que posiblemente formaran parte de la siguiente versión estable.

3. Inestable (*Unstable*): Lo de “inestable” en esta versión en realidad es sólo una precaución para decir que no ha sido suficientemente probada, todo nuevo paquete que se considere puede formar parte de Debian entra primero en esta versión para ser probado exhaustivamente por la comunidad, después de cierto tiempo continua su paso ascendente hacia la estable.

En la actualidad (al momento de escribir este libro) la versión oficial (estable) de Debian es la 3.0r1 (conocida como *woody*) y está en prueba para su paso a estable la denominada *sarge* (posiblemente se numerará como 3.1).

5.2.2. KNOPPIX

http;//www.knopper.net/knoppix

Para hablar sobre esta distribución es indispensable primero comentar brevemente acerca del concepto de «*Live* CD», un «*Live* CD» es una distribución GNU/Linux que no necesita ser instalada, ya que arranca desde el propio CD y crea un disco virtual en la memoria RAM en donde almacena de forma temporal todas las aplicaciones que necesita para su ejecución, por lo cual es posible mantener un sistema operativo completo (incluso con entorno gráfico) sin alterar absolutamente nada en la máquina, en el momento en que se requiere el uso de un progrma determinado, el *kernel* carga desde el CD hasta el disco virtual el paquete binario y sus librerias compartidas, y a partir de estos ejecuta el programa; otra caracteristica muy interesante acerca de esta tecnología es la auto-detección de hardware, por lo regular el propio sistema se encarga de detectar el hardware sobre el que está funcionando y trata de configurarlo de forma automática. Los *Live* CDs son ampliamente utilizados para demostraciones y para discos de rescate.

Knoppix es la más común de las distribuciones «*Live*», fue desarrollado por Klauss Knopper teniendo como base la distribución Debian GNU/Linux, en un sólo CD Knoppix incluye la mayoría de herramientas que cualquier persona necesitaría en un sistema operativo: *Suite* de Oficina (*OpenOffice*, *KOffice*), Aplicaciones Multimedia (*XMMS*, *MPlayer*, etc), Tratamiento de Imágenes (*GIMP* y ptros), Aplicaciones para Internet (*Mozilla*, *Konqueror*, *GFTP*, etc), Herramientas de Desarrollo (*GCC*) y la mayoría de herramientas de línea de comando para administración del sistema (y de la red) que traen por lo regular las demás distribuciones. Adcionalmente KNOPPIX utiliza *KDE* como interfaz gráfica.

5.2.3. MANDRAKE

`http://www.mandrakelinux.com`

Mandrake GNU/Linux es desarrollado por la casa francesa *Mandrake Soft* (`http://www.mandrakesoft.com`), es basada en la distribución RedHat y es una de las más amigables con el usuario. A la fecha la última versión es la 9.2 (conocida con el nombre de *FiveStar*).

5.2.4. MORPHIX

`http://www.morphix.org`

Morphix es otra de las «grandes» (en términos de popularidad) distribuciones del tipo «*Live* CD», al igual que *Knoppix* esta tiene como base *Debian* GNU/Linux, la principal caracteristica de Morphis es que en realidad es una «meta-distribución», es decir una distribución a partir de la cual es relativamente sencillo crear otra, como ejemplo de esto en el sitio web de esta distribución es posible encontrar 4 diferentes versiones:

- *Morphix Combined Gamer*: Versión orientada al entretenimiento, esta versión incluye gran cantidad de juegos tanto multi-jugador como individuales, además de los controladores (*drivers*) para varias de las tarjetas de aceleración gráfica más difundidas.

- *Morphix Combined Gnome (ó M. C. HeavyGUI):* Esta versión es para máquinas con altas capacidades de procesamiento (+128 Mb en RAM, *Pentium* II -o superior-), incluye como el nombre lo indica el gestor de ventanas *GNOME*, así como gran variedad de aplicaciones para el trabajo diario, y el tratamiento de imágenes, la especial característica de la mayoría de estas aplicaciones es su gran uso de los recursos del sistema,

en especial para poder desplegar una interfaz de usuario visiblemente muy atractiva.

- *Morphix Combined KDE*: La versión *KDE* de esta distribución además de incluir este gestor de ventanas contiene gran cantidad de aplicaciones que estan diseñadas para un mejor desempeño sobre KDE (algunas incluso sólo corren bajo este entorno gráfico).

- *Morphix Combined LigthGUI*: La versión de interfáz liviana ofrece como entorno gráfico *XFCE*, y algunos populares paquetes de productividad pero con la carcateristica principal de su bajo consumo de recrusos de *hardware*.

5.2.5. REDHAT

http://www.redhat.com

Es tal vez la más utilizada a nivel mundial, su desarrollo está a cargo de *Red Hat Inc*. empresa norteamericana que es la más grande (a nivel de ventas y capital) en el mundo *Open Source*. *RedHat* fue desarrollada originalmente por *Marc Ewing* en 1994 como una alternativa a *Slackware* (las más popular en ese entonces), luego a finales de ese mismo año se asocia con *Bob Young* y se empieza a consolidar lo que más adelante sería *RedHat Inc*.

La principal característica a resaltar de *RedHat Linux* es su sistema de paquetes (*RedHat Package Manager* -Manejador de Paquetes de RedHat, más conocido como RPM) el cual se constituyó en una verdadera revolución facilitando la instalación de software de manera sorprendente (en esa época para instalar cualquier paquete tocaba hacerlo desde sus fuentes, compilando cada paquete de manera independiente).

En la actualidad la última versión en ser liberada es la 9.0.

5.2.6. SLACKWARE

slackware linux **http://www.slackware.com**

Es una de las distribuciones más antiguas en el mercado (su primera versión salió en abril de 1993), fue desarrollada originalmente por *Patrick Volkerding* y en la actualidad es mantenida por la empresa *Slackware Inc*. en la que participan varias personas más. La filosofía de *Slack* es ser la distribución "más similar a UNIX", sin dejar de lado obviamente los estándares Linux. La última versión oficial es la 9.1.

5.2.7. SUSE

http://www.suse.com

Es una distribución Alemana basada en RedHat, es una de las más amigables con el usuario destacándose especialmente su sistema de instalación / configuración ***YaSt*** , el cual le permite al usuario desde un sólo lugar configurar tanto Hardware como Software, además de la administración general del sistema. Su ultima versión es la 9.0.

5.3. INSTALACIÓN

GNU/Linux se puede instalar de dos formas:

- Sobre un sistema DOS o Windows, utilizando una partición existente (que sólo puede ser FAT o FAT 32) o
- Utilizando particiones idendependientes

La primera opción no es la más recomendada ya que el sistema perdería gran parte de sus características como seguridad y eficiencia, por este motivo no se va a comentar acá.La segunda es la mejor opción y por lo tanto es a esta a la que nos vamos a referir en este apartado. Una partición es una división lógica que se hace sobre un disco duro con el fin de poder tener diferentes sistemas de archivos; cada partición actua practicamente como si fuese un disco totalmente independiente de las demas particiones.

5.3.1. CONSIDERACIONES ANTES DE LA INSTALACIÓN

5.3.1.1. SISTEMAS DE ARCHIVOS

Un sistema de archivo es la forma en que se organizan tanto lógica como físicamente los archivos en un sistema operativo, para el caso más popular como lo es Windows, sus sistemas de archivos pueden ser FAT 16 (desde DOS hasta Win95 SP 2), FAT 32(Win 95 SP2, Win 98, Win Me) o NTFS (Windows NT, Win 2000 y 2003); Linux soporta gran cantidad de estos, entre los que se pueden destacar *extended* 2 (o 3)FS, FAT 16 (o32), NTFS (sólo lectura, la escritura todavía se encuentra en estado beta), MinixFS, HFS (sistema de archivos de los Macintosh), entre otros.

5.3.1.2. DISTRIBUCIÓN DEL ESPACIO (PARTICIONAMIENTO)

Para su instalación GNU/Linux requiere **por lo menos** dos particiones (teóricamente no hay limite en el numero de particiones sobre las que se puede distribuir el sistema de archivos), una debe ser de tipo *swap* (utilizada como memoria virtual, se puede asimilar casi como el archivo Win386.swp en Windows, salvando las grandes diferencias entre ambos sistemas) y la otra (s) de cualquier tipo de archivos soportado por el kernel (por defecto en los núcleos anteriores al 2.4 se utilizaba el ext2, a partir de este se utiliza mayormente ext3).

Este requerimiento de por lo menos una partición es necesario ya que todo el sistema de archivos debe "montarse" en alguna partición libre y en UNIX todo archivo esta bajo la jerarquía del directorio / (*root* o raíz), por lo cual si sólo existe una partición en esta será donde se monte el mencionado directorio, al poder hacer esto si se tienen más particiones disponibles es posible montarla como un directorio en cualquier lugar por "debajo" del raíz.

Es recomendable el particionar el disco en especial si se trata de una máquina servidor ya que esto brinda una poca más de seguridad; típicamente en un servidor se crean particiones para los directorios de nivel superior `/`, `/boot`, `/home`, `/root`, `/tmp`, `/usr/var` (más adelante en la sección 6.2 se describe el contenido de estos y para que son utilizados) la siguiente lista presenta unas recomendaciones del espacio en disco sugerido para cada una de estas:

- `/` ⇒ Requiere por lo menos 500 Mb, esto en el caso de que las particiones `/home`, `/usr` y `/var` sean independientes, de lo contrario es necesario sumar a esta cifralo sugerido para estas particiones.

- */boot* ⇒ Esta es una partición de pequeño tamaño (no más de 30 o 50 Mb), algunas personas recomiendan que se coloque en los primeros cilidros del disco debido en especial al problema con las rutinas de arranque de algunos BIOS antiguos, las cuales no pueden leer más alla del cilindro 1024 (algo así como los 8 Gb), pero es importante resaltar que esto no es de obligatorio cumplimiento debido a que la mayoria de BIOS modernos ya solucionan este problema y además que se recomienda por lo regular instalar el gestor de arranque en el mbr (master boot record -registro maestro de arranque-).

- */home* ⇒ Esta partición es de tamaño muy variable ya que este depende del número de usuarios que va a tener el sistema y a la cantidad de espacio que se le va a asignar a cada uno, es recomendable hacer una buena previsión y siempre mantener espacio libre por si hay nuevos usuarios, claro que este espacio debe ser razonable y si por algún caso llega a fallar no hay problema ya que se puede utilizar otra partición en otro disco

o incluso crear un raid por sofware (arreglo de partciones que actuan lógicamente como si fueran una sóla).

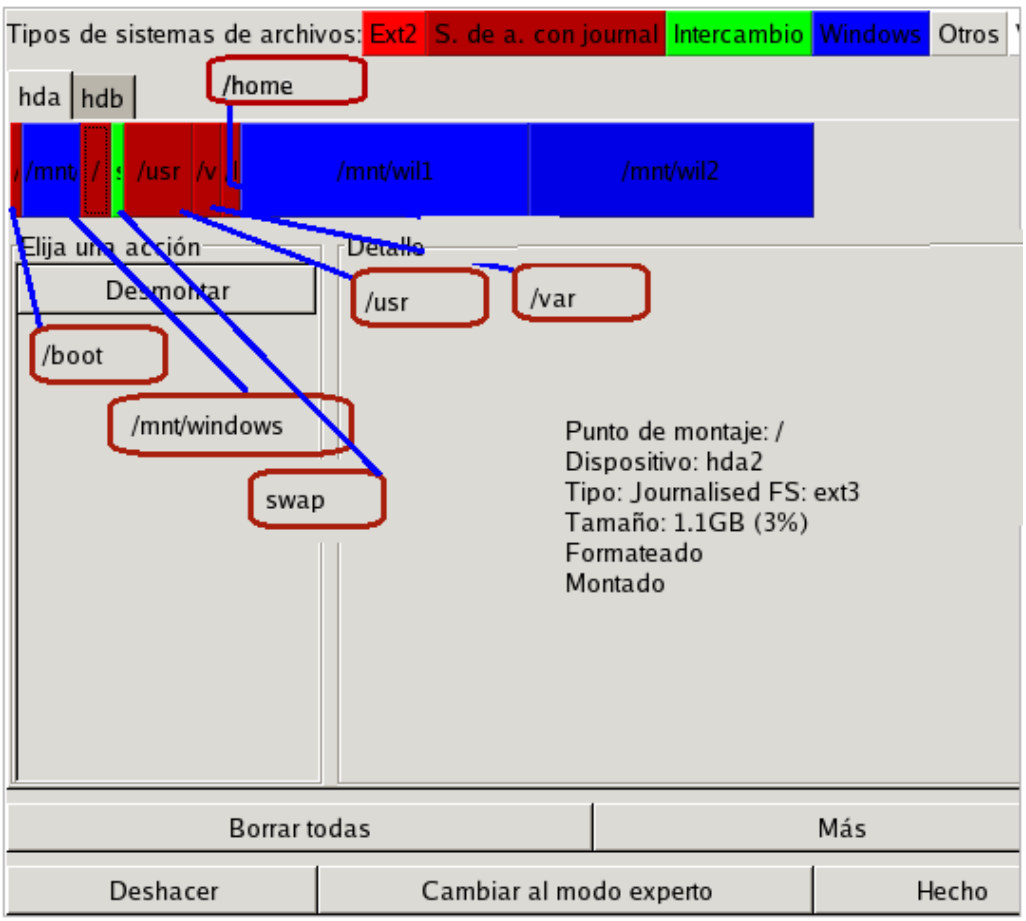
- */root* ⇒ Como acá se guardan los archivos privados del administrador (root) esta partición también puede ser de tamaño variable dependiendo de nuestra necesidad o pensamiento. Es recomendable cerca de 500 Mb o más.

- */tmp* ⇒ Esta partición no requiere mas de 100 o 200 Mb.

- */usr* ⇒ Esta es una de las más grandes requiere alrededor de 4 Gb, dependiendo de la cantidad de paquetes de software a instalar.

- */var* ⇒ Para el caso de los servidores esta partición también debe tener entre 3 y 5 Gb o dependiendo de la necesidad, ya que acá se guardan muchos archivos que pueden llegar a crecer bastante (p.e. las bases de datos o los *spool* de correo o *proxy*).

Figura 5.4: Particionamineto de un disco (en una instalación de Mandrake GNU/Linux 9.1)



Capítulo 6

COMENZANDO CON GNU/LINUX

6.1. EL SISTEMA DE ARCHIVOS DE UNIX

Una de las principales características de UNIX es que para este todo es un archivo, esto quiere decir que no trata de forma diferente a los archivos y a los dispositivos, permitiendo un acceso uniforme a los recursos del sistema ya que todo archivo debe estar montado sobre la raíz /.Cabe notar antes de continuar hablando sobre el sistema de archivos que UNIX hace diferencia entre MAYUSCULAS y minusculas, por lo cual para UNIX son diferentes Archivo, archivo, ARCHIVO y ArchivO; por este motivo en todo el libro se ha segido este mismo comportamiento.

6.1.1. TIPOS DE ARCHIVOS

Como se dijo antes en UNIX todo son archivos, y dependiendo de sus características pueden ser:

- Archivos corrientes

- Directorios
- Enlaces
- Archivos especiales

6.1.1.1. ARCHIVOS CORRIENTES

Son los que se utilizan de forma común, en estos se guarda la información con la que trabajamos de forma habitual (textos, gráficos, etc), generalmente se distiguen mediante un nombre y una extención (aunque esta no es obligatoria).

6.1.1.2. DIRECTORIOS

Es un tipo de archivo que se utiliza para agrupar otros archivos (de cualquier tipo), los directorios de presencia obligatoria en una máquina UNIX se describen en la sección 6.2.

6.1.1.3. ENLACES

Los enlances surgen como una mejora a los archivos tradicionales, con el fin de permitir hacer referencia a un mismo archivo con diferente nombre o desde otra ubicación.

ENLACES DUROS (O FUERTES) Un enlace duro es una forma de dar un nuevo nombre a un archivo ya existente, pero sin tener que hacer una nueva copia (y por tanto no ocupar mayor espacio en el disco), pero este proceso no se puede llevar a cabo si el archivo es un directorio o si este se encuentra ubicado en una partición diferente (o incluso en una máquina diferente) a donde se va a crear el enlace.

ENLACES SIMBÓLICOS Los enlaces simbólicos suplen las limitaciones de los anteriores, es posible con estos enlazar cualquier tipo de archivo sin importar la ubicación de su destino u origen (si se utiliza un sistema de archivos distribuido -como NFS- es posible incluso hacer enlaces entre diferentes máquinas). En realidad un enlace símbolico es un nuevo archivo que como única información contiene la ruta del que está enlazando, por lo general las aplicaciones no tienen necesidad de distinguir entre un enlace simbólico (o duro) ya que estas acceden al sistema de archivos mediante las rutinas estandar que provee el S.O.

6.1.2. LA RUTA O PATH

Para poder referirnos a un archivo dentro del arbol de directorios es necesario indicar el camino o ruta (path) hacia este, la ruta es una secuencia de caracteres donde cada directorio se separa por una barra (/) hasta llegar a donde se encuentra el archivo, etas rutas pueden ser de dos tipos:

- Absolutas o
- Relativas

6.1.2.1. RUTAS ABSOLUTAS

Las rutas absolutas se distinguen por que comienzan con /, se denominan absolutas por que la referencia se hace desde el directorio raíz hacia abajo.

Ejemplo: ***/etc/postgresql/pg_hba.conf***

6.1.2.2. RUTAS RELATIVAS

En estas la referencia depende del directorio de trabajo actual, se caracterizan por que **no** comienzan con /. Por ejemplo si tenemos la siguiente estructura:

- */etc*
  - */etc/postgresql*
    - */etc/postgresql/pg_hba.conf*
  - */etc/ssh*
    - */etc/ssh/ssh_config*

Y estando posicionados sobre el directorio */etc/ssh* queremos hacer referencia al archivo */etc/postgresql/pg_hba.conf*, se puede escribir: *../postgresql/pg_hba.conf*, otro ejemplo con la misma estructura puede ser: estando sobre */etc/ssh* se desea hacer referencia al archivo */etc/ssh/ssh_config*, por lo cual de forma relativa la ruta sería *./ssh_config*.

Cabe aclarar las convenciones utilizadas en estas rutas:

- **.** → Directorio actual.
- **..** → Directorio superior o padre.
- **~** → Directorio *home* del usuario.

6.1.3. MANIPULACIÓN DE ARCHIVOS Y DIRECTORIOS

6.1.3.1. LISTADO DEL CONTENIDO DE UN DIRECTORIO

La orden en *UNIX* para listar el contenido de un directorio es ***ls*** (list), esta orden a pesar de ser muy sencilla tiene una variedad de opciones que le permiten personalizar la salida de esta, su sintaxis varia entre algunos UNIX sin embargo se emplean dos estándares *POSIX* o *GNU*, dependiendo si el sistema operativo utiliza el conjunto de utilidades proporcionados por el proyecto *GNU* o no:

```
ls [opciones] [archivo...]
```

6.1.3.2. MOSTRAR DIRECTORIO ACTUAL

Para este propósito se utiliza la instrucción *pwd* (*print working directory*), su sintaxis es:

```
pwd [--help,--version]
```

La opción *–help* proporciona una ayuda en línea de la instrucción, *–version* muestra la versión de *pwd* que se está utilizando.

6.1.3.3. CAMBIAR DE DIRECTORIO

Para cambiar de un directorio de trabajo a otro es necesario utilizar la orden *cd* (*change directory*), la sintaxis de *cd* es:

```
cd [directorio]
```

Cabe notar que la ausencia de directorio al llamar la orden indica que se debe cambiar al directorio casa (*home*), así mismo los comodines presentados al final de la sección 6.1.2.2 en la página 108, tienen el mismo comportamiento (esta aclaración vale para todas las demás instrucciones que manipulen directorios, por esta razón no se seguirá incluyendo).

6.1.3.4. CREACIÓN DE DIRECTORIOS

Para realizar esta operación simplemente se utiliza la instrucción *mkdir* (*make directory*) cuya sintaxis es:

```
mkdir [opciones] directorio ...
```

Las opciones de esta instrucción se presentan en la tabla 6.1.

Opción Corta	Opción Larga	Descripción
-m	*–mode=MODE*	Establece el modo de creación del directorio, por defecto este es rwxrwxrwx
-p	*–parents*	En caso de no existir los padres de la ruta en un directorio, los crea
-v	*–verbose*	Por cada directorio creado muestra un mensaje
	–help	Muestra el texto de ayuda
	–version	Informa la versión del programa

Cuadro 6.1: Opciones de mkdir

6.1.3.5. BORRADO DE DIRECTORIOS

Para borrar directorios debemos utilizar la orden *rmdir* (*remove directory*), la sintaxis de esta es:

rmdir **[opciones] directorio ...**

Opción Corta	Opción Larga	Descripción
	–ignore-fail-on-non-empty	Elimina el mensaje de error que se produce cuando un directorio no está vacio, por lo cual lo deja igual y salta al siguiente (en caso que se traten de borrar varios.

-p	*–parents*	Elimina el directorio al final de la ruta y luego intenta borrar en orden ascente (der. a izq.) los demás directorios que componen la ruta. Ejemplo: *rm -p xxx/yyy/zzz* intentará borrar zzz, luego yyy y por último xxx.
-v	*–verbose*	Despliega un mensaje cada vez que se intenta borrar un directorio.
	–help	–Muestra un mensaje con la ayuda de estas opciones.
	–version	Visualiza la versión del programa.

Cuadro 6.2: Opciones de rmdir

La tabla6.2 resume las opciones de esta instrucción, vale la pena tener en cuenta que *rm* sólo borrará el directorio si este se encuentra vacio.

6.1.3.6. COPIAR ARCHIVOS Y DIRECTORIOS

Para copiar archivos y directorios se necesita utilizar la orden *cp*:

```
cp [opción] ... origen destino
```

Origen hace referencia a uno o más archivos que se desean copiar, origen puede ser el nombre de otro archivo (si no existe es creado) cuando origen es un sólo archivo, en caso contrario destino debe ser un directorio (que exista). Algunas de las opciones más comunes de esta instrucción se encuentran resumidas en la tabla 6.3.

Opción Corta	Opción Larga	Descripción
-d	*–no-dereference*	Omite los enlaces simbólicos al copiar
-f	*–force*	En caso que exista el archivo en destino, lo sobreescribe
-i	*–interactive*	Pide confirmación antes de sobreescribir cada archivo
-l	*–link*	En lugar de copiarlos, crea un enlace entre los archivos
-p	*–preserve*	Preserva (si es posible) el modo de los archivos cuando los copia
-R, -r	*–recursive*	Copia recursivamente, es decir copia también los subdirectorios y su contenido
-u	*–update*	Sólo sobreescribe cuando la fecha de modificación del origen es más reciente que la del destino
-v	*–verbose*	Muestra un mensaje al comenzar a copiar cada archivo.

Cuadro 6.3: Opciones de cp

6.1.3.7. MOVER O RENOMBRAR ARCHIVOS

Aunque existe una instrucción únicamente para renombrar archivos es posible (y algunas veces recomendable) la utilización de *mv* (*move*) tanto para mover archivos o directorios, como para renombrarlos, esto se debe que cuando movemos un archivo realmente lo que estamos haciendo es cambiar su ruta o camino absoluto, para aclar mejor esta idea podemos utilizar un ejemplo:

Supongamos que vamos a mover el directorio `/etc/postgresql` a `/usr/local/et` para esto la instrucción completa sería:

```
mv /etc/postgresql /usr/local/etc/
```

Ahora imaginemos que no sólo queremos mover el directorio */etc/postgresql*, también deseamos que tenga el nombre *postgres* aunque en el mismo directorio */usr/local/etc/*, para este fin utilizaremos:

```
mv /etc/postgresql /usr/local/etc/postgres
```

Como podemos ver hemos utilizado la misma instrucción tanto para mover el directorio como para cambiarle de nombre, aunque en el ejemplo al cambiar el nombre también se haya movido a otro directorio, es perfectamente posible cambiarle el nombre sin necesidad de tocar los directorios, la sintaxis de *mv* puede ayudar a despejar dudas:

`mv [opción]... origen... directorio`

Opción puede ser cada una de las que aparecen en la tabla 6.4.

Opción Corta	**Opción Larga**	**Descripción**
-f	*–force*	Forza el reemplazo de los archivos o directorios en el destino cuando este existe
-i	*–interactive*	Antes de efectuar un reemplazo le pide confirmación al usuario.

Opción Corta	Opción Larga	Descripción
-u	*–update*	Sólo sobreescribe cuando la fecha de modificación del origen es más reciente que la del destino
-v	*–verbose*	Muestra un mensaje indicando origen y destino al comenzar a copiar cada archivo.
	–help	Presenta la ayuda breve de la instrucción.

Cuadro 6.4: Opciones de mv

6.1.3.8. BORRADO DE ARCHIVOS

Para borrar archivos es necesaria la utilización de la orden *rm* (*remove*), debido a las características *UNIX* de permitir enlaces duros, cuando se utiliza esta orden sólo es recuperado el espacio en disco al momento de borrar el último enlace duro que haga referencia al archivo. La sintaxis de *rm* se muestra a continuación, y en la tabla 6.5 se resumen las opciones más populares.

Opción Corta	Opción Larga	Descripción
-f	*–force*	Al igual que en las otras instrucciones, esta opción forza el reemplazo en el destino.
-i	*–interactive*	Pregunta antes de sobreescribir.
-r, -R	*–recursive*	Borra el contenido de los directorios (y sus subdirectorios).

Opción Corta	Opción Larga	Descripción
-v	*–verbose*	Indica cada vez que un archivo va a ser borrado.
	–help	Muestra la ayuda corta, con todas las opciones.

Cuadro 6.5: Opciones de rm

6.1.3.9. CREACIÓN DE ENLACES

Los enlaces son una gran utilidad que nos proveen los sistemas *UNIX*, como se mencionó en la sección 6.1.1.3 en la página 106, estos pueden ser de dos tipos: **duros** o **simbólicos**.

ENLACES DUROS Como ya se habló un enlace duro es un tipo especial de archivo que pudiendo estar en diferentes directorios hace referencia a otro archivo (realmente al nodo-i, es decir a su ubicación física en el disco), con lo cual podemos tener el mismo archivo en diferentes lugares pero sin tener que mal gastar el espacio en disco, para la creación de un enlace duro se utiliza la sencilla instrucción *ln* (*link*), cuya sintaxis es:

ln origen ... [enlace] o **ln origen ... directorio**

Como se puede ver crear un enlace es bastante sencillo, basta con escribir cada uno de los archivos a los que se les desea crear el enlace y un directorio, cuando se desea crear un enlace sólo a un archivo se puede utilizar como último parámetro el nombre y/o ruta que va a tener el nuevo enlace, cuando se omite este parámetro se crea un enlace con el mismo nombre del archivo origen en el direcotrio de trabajo actual.

ENLACES SIMBÓLICOS Los enlaces simbólicos son una alternativa bastante útil respecto a los enlaces duros, en especial cuando es necesario hacer enlaces de directorios ya que no es posible hacer enlaces duros de directorios, un enlace simbólico se crea utilizando la siguiente sintaxis:

```
ln -s origen ... [enlace]
```

Cuando se desea crear enlaces a varios directorios o archivos es necesario que el último parámetro sea un directorio o que este se encuentre en blanco, con esto se crearan los enlaces con el mismo nombre que el original en el directorio seleccionado (o en el directorio actual, según el caso), si se desea cambiar el nombre de un enlace cuando es creado no es posible crear simultaneamente varios.

6.1.4. SEGURIDAD EN ARCHIVOS Y DIRECTORIOS

En los sistemas *UNIX* todo archivo tiene un usuario que es el propietario del mismo y también un grupo al que pertenece el archivo (por defecto el mismo grupo del propietario), por este motivo la seguridad en estos sistemas se basa en las posibilidades de acceso tanto del propietario como del grupo así como del resto de usuarios que no encajan en ninguna de las dos categorias. Los permisos de acceso (o modos) se basan en las tres posibilidades de uso de los archivos y/o directorios, es decir un archivo puede ser **leido**, **escrito** o **ejecutado** (para el caso de los directorios la ejecución es la posibilidad de entrar en este), de estas dos afirmaciones se desprende que cada archivo le puede o no permitir a los tres diferentes tipos de usuarios de forma independiente permisos de lectura, escritura y/o ejecución.

6.1.4.1. CAMBIO DE PROPIETARIO Y GRUPO

Para cambiar el propietario de un archivo es necesaria la utilización de la orden *chown* (*change owner*), cuya sintaxis es la siguiente:

`chown [opción] ... propietario[.grupo] archivo ...`

De esta forma podemos cambiar el propietario y opcionalmente el grupo de un archivo o de varios archivos, opción es alguno de los posibles modificadores de *chown*, siendo el más popular **-R** ya que permite cambiar de forma recursiva el propietario y/o grupo de un directorio y de su contenido de archivos y directorios recursivamente, para obtener detalles de la orden se recomienda consultar las paginas de manual o utilizar el modificador ***–help*** para una ayuda corta.

Esxite también la orden *chgr* (*change group*), la cual permite únicamente cambiar el grupo propietario del archivo, esta orden se utiliza de igual manera que *chown*.

6.1.4.2. MANIPULACIÓN DE PERMISOS

Ya se ha comentado que todo archivo en *UNIX* tiene tres tipos de permisos para tres tipos de usuarios, ahora bien estos permisos obviamente son susceptibles de ser cambiados de acuerdo a la necesidad, para esta acción se utiliza la herramienta *chmod* (*change mode*), la cual tiene la siguiente sintaxis:

`chmod [OPCIÓN]... MODO[,MODO]... archivo...` o también:

`chmod [OPCIÓN]... MODO-OCTAL archivo...`

el MODO de un archivo es su nivel de permiso, pudiendo ser es una o más de las letras **u**, **g**, **o**, **a**, **(**user, group, other, all, usuario o propietario, grupo, otros, todos) junto a uno de los símbolos **+**, **-**, **=** (poner permiso, quitar permiso, dejar igual) y una o más de las letras **r**, **w**, **x** (lectura, escritura, ejecución).

El modo octal se basa en la caracteristica de que 3 bits puedan conformar un número octal (base 8, 0-7), por lo cual cada combinación de 3 bits puede representar una cadena con los símbolos + o - y una de las letras r, w, x, tal como lo resume la siguiente tabla (6.6).

Binario	Octal	Cadena
000	0	-r-w-x
001	1	-r-w+x
010	2	-r+w-x
011	3	-r+w+x
100	4	+r-w-x
101	5	+r-w+x
110	6	+r+w-x
111	7	+r+w+x

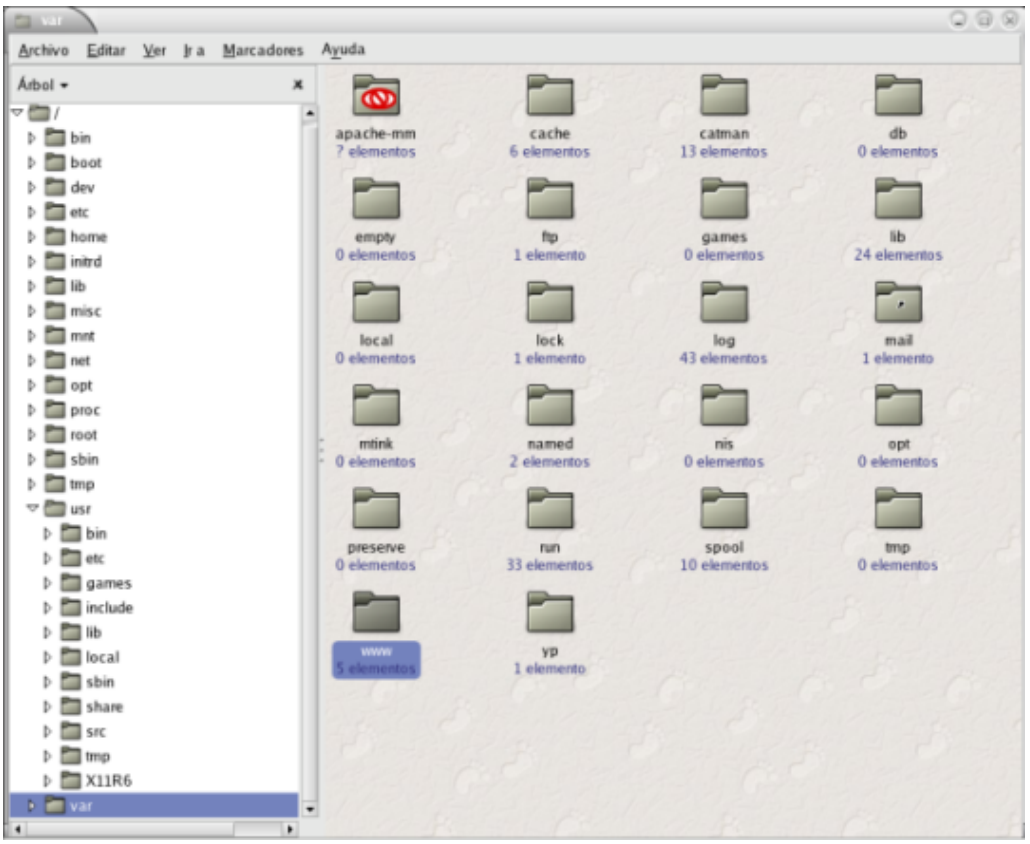
Cuadro 6.6: Valores Binarios/Octales de Permisos

De acuerdo a esto es posible establecer permisos agrupando en una sóla cadena tres valores octales que en su orden representen los permisos para propietario, grupo y otros. Por ejemplo para conceder todos los permisos al propietario, lectura y ejecución al grupo y sólo ejecución a los demás, se debe utilizar la cadena numérica **751** como modo octal en una instrucción *chmod*.

6.2. ESTRUCTURA DEL SISTEMA DE ARCHIVOS

En esta sección se pretende dar una introducción a la forma en que se organizan los archivos y directorios en GNU/Linux, se va a procurar en todo seguir el *Filesystem Hierarchy Standard* (Estandard de Jerarquía del Sistema de Archivos) versión 2.2 [FHS-2.2] (Una versión en español aunque antigua (1.2) se puede encontrar en la web del poyecto LUCAS en la sección sección estandares - `http://es.tldp.org/htmls/estandares.html`-), aunque algunas veces cuando el estandar no lo defina claramente y el directorio sea importante, se seguira la estructura de Debian GNU/Linux.

Figura 6.1: Arbol del sistema de archivos



El FHS hace énfasis en la distinción de dos categorías de archivos independientes: “compartible” vs “no compartible” y “estática” vs “variable”. Esto quiere significar que un directorio puede ser compartido o no compartido y a la vez de contenido variable o estático, un ejemplo típico que propone el estandar es:

	Compartible	**No compartible**
Estático	`/usr` `/opt`	`/etc` `/boot`
Variable	`/var/mail` `/var/spool/news`	`/var/run` `/var/lock`

Cuadro 6.7: Categorias de archivos

6.2.1. EL SISTEMA DE ARCHIVOS RAÍZ (/)

Los contenidos de este sistema archivos son adecuados para arrancar, recuperar y/o reparar el sistema, por ello los archivos y directorios que se encuentren bajo está jerarquía deben ser los suficientes para que el usuario (aunque sobra decir que este sólo puede ser **root** o alguno que tenga los suficientes privilegios) pueda llevar a cabo estas tareas.

La tabla 6.8, resume la descripción de los directorios que según el FHS pueden aparecer en este directorio.

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
/bin	Contiene comandos que pueden ser usados tanto por el administrador del sistema como por los usuarios, en este directorio no pueden haber subdirectorios	*cat* *dd* *chgrp* *df* *chmod* *hostname* *chown* *ln* *cp* *ls*
/boot	Contiene todo lo necesario para el proceso de arranque excepto archivos de configuración y el instalador del mapa, el kernel utiliza la información almacenada en este directorio para poder arrancar el sistema, en los sistemas GNU/Linux por lo general se debe encontrar el archivo *vmlinuz* o *vmlinuz-versión* el cual corresponde al kernel y es el que se cargará en memoria principal al inicio de la máquina. Algunas distribuciones guardan estos archivos bajo / en lugar de este directorio, o utilizan enlaces símbolicos en / de los archivos reales en /boot.	*vmlinuz-[ver]* *initrd.img-[ver]*
/dev	En este directorio UNIX guarda los archivos de dispositivo, cada uno de estos representa un dispositivo en el sistema o la posibilidad de acceder a este en caso de que esté instalado en la máquina.	*hda* *fd0* *mouse* *psaux* *sdc* *tty*

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
`/etc`	Acá se almacena información de carácter específico para la máquina en que se está ejecutando el S.O. Es importante resaltar que no debe haber binarios en este directorio.La mayoría de paquetes de software instalan sobre este directorio (o un subdirectorio dentro de este) sus archivos de configuración.	Ver sección: 6.2.1.1
`/home`	Este directorio es opcional pero se constituye en un estandar de-facto ya que la mayoría de distribuciones crean acá los directorios de trabajo de los usuarios.	
`/lib`	Contiene las librerias compartidas necesarias para arrancar el sistema y ejecutar comandos almacendos en directorios como /bin y /sbin.Dentro de este existe el subdirectorio modules donde se encuentran los módulos cargables por el kernel.	

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
/mnt	Sobre este directorio se montan los sistemas de archivos que son temporales, por ejemplo el floppy (diskkette) o cdrom, algunas distribuciones incluyen /mnt/disk como punto de montaje de sistemas de red y /mnt/windows (o /mnt_win* para cuando hay varias particiones) para montaje de particiones windows (estos nombres no hacen parte del estandar y varian dependiendo de la distribución).	
/opt	Paquetes de software grandes de terceros	
/proc	En este directorio se guarda información importante para el tratamiento de procesos e información del sistema. Por cada proceso que se inicia en la máquina se crea un directorio que contiene los archivos con la información de este.	*/proc/cpuinfo* ≻ (Información del procesador o procesadores) */proc/filesystems* ≻ (Información de los sistemas de archivos soportados por el kernel) */proc/meminfo* ≻ (Estado de la memoria) */proc/mounts* ≻ (Sistemas de archivos montados), */proc/pci* ≻ (información de los dispositivos PCI encontrados).

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
`/root`	Este es el directorio *home* del usuario root.	
`/sbin`	Se encuentran acá archivos binarios para la administración del sistema (y otros comandos sólo ejecutables por root), en especial para el arranque, restauración, recuperación y/o reparación del sistema. El único archivo que el FHS define como obligatorio sobre este es *shutdown* (el comando usado para apagar el sistema), sin embargo algunos otros opcionales (según el estandar pero de uso regular en la mayoría de dsitribuciones) son: *ifconfig* (cofigura las interfaces de red), *init* (proceso inicial , además permite cambiar el nivel de arranque del sistema), *route* (configura la tabla de enrutamiento IP).	
`/tmp`	Se utiliza para almacenar archivos de uso temporal por parte de los programas, algunos administradores acostumbran en cada arranque del sistema borrar todos los archivos de este directorio.	
`/usr`	La mayoría del software del sistema se instala acá	
`/var`	Guarda información de tipo variable.	

Directorio	Descripción	Algunos archivos (o directorios) importantes

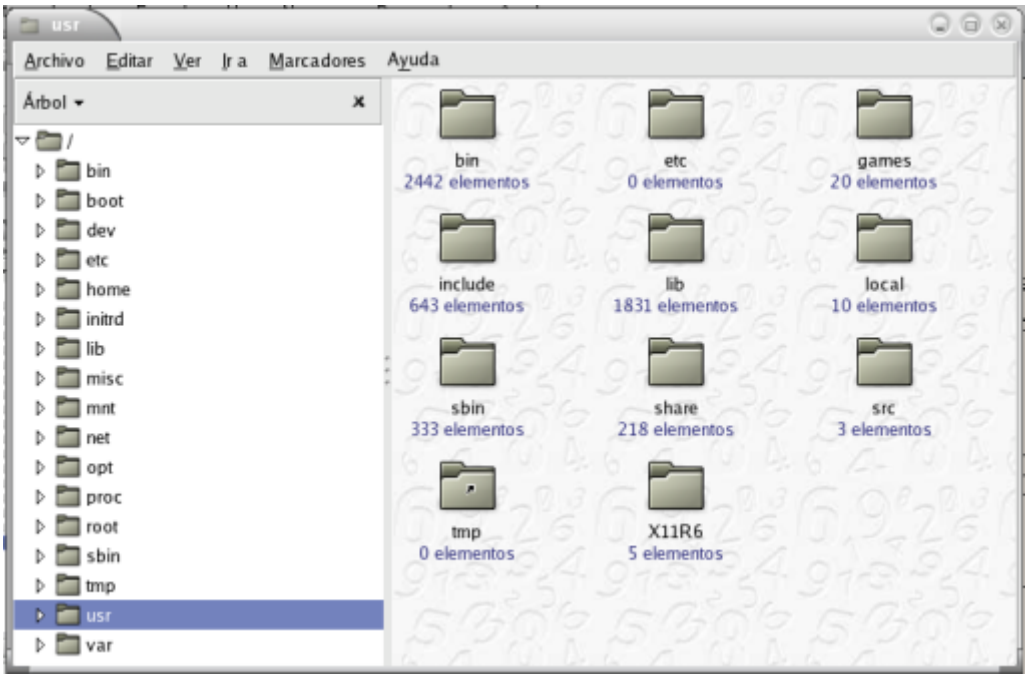
Cuadro 6.8: Jerarquia /

6.2.1.1. /etc

Algunos archivos importantes que se pueden encontrar (no son obligatorios) en este directorio son:

- `exports` ≻ Binarios Listas de control de acceso del sistema de archivod NFS
- `fstab` ≻ Información estática sobre el sistema de archivos
- `hosts` ≻ Información estática sobre nombres de *hosts*
- `inittab` ≻ Archivo de configuración de *init* (proceso inicial que arranca todos los demás procesos)
- `passwd` ≻ Archivo de contraseñas del sistema
- `resolv.conf` ≻ Configuración del sistema de resolución de nombres
- `services` ≻ Nombres de puerto para servicios de red
- `shadow` ≻ Archivo de contraseñas del sistema encriptadas (no hace parte del FHS pero se encuentra en la mayoría de distribuciones)
- `syslog.conf` ≻ Archivo de configuración de *syslog* (historial del sistema)

Figura 6.2: Jerarquía */usr*



6.2.2. LA JERARQUÍA /usr

Esta es la segunda mayor sección del sistema de archivos, el FHS recomienda montar esta partición para que se pueda compratir, pero de sólo lectura, con lo cual se grantiza que se pueda compartir el software entre varias máquinas pero que no se pueda modificar la información acá guardada.

A continuación se describen algunos de los directorios más importantes de esta jerarquía.

Directorio	Descripción	Algunos archivos (o directorios) importantes
/usr/bin	En este directorio se encuentran la mayoría de comandos en el sistema ejecutables por los usuarios.	

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
`/usr/include`	Acá se localizan todos los archivos de cabeceras de uso general para la programación en lenguaje C. Los encabezados específicos de los programas se guardan como subdirectorios dentro de este (p.e */usr/include/postgresql* o */usr/include/php4*)	`ctype.h` `stdlib.h` `malloc.h` `string.h` `math.h` `time.h` `stdio.h`
`/usr/lib`	En esta localización se guardan los archivos objeto, librerias y binarios internos que no están pensados para ejecutarse directamente por usuarios o *scripts shell* (p.e. el subdirectorio *perl5* para los modulos y librerias de Perl5)	

Directorio	**Descripción**	**Algunos archivos (o directorios) importantes**
`/usr/local`	Este directorio se utiliza para instalar software de manera local, por lo general se intalan acá aplicaciones que luego serán de uso general en la red en el directorio */usr*	`bin` Binarios locales `games` Binarios de juegos locales `include` Archivos locales de encabezados C `lib` Librerias locales `man` Manuales en línea locales (también llamadas páginas de manual) `sbin` Binarios locales del sistema `share` Jerarquía local independiente de la arquitectura `src` Código fuente local
`/usr/sbin`	Contiene binarios de uso exclusivo por el administrador del sistema	

Directorio	Descripción	Algunos archivos (o directorios) importantes
/usr/share	Es una jerarquía para compartir archivos independientes de la arquitectura y de sólo lectura para todos.	doc Documentación miscelanea (opcional) info Directorio primario del sistema *Info* de *GNU* (op) locale Información de "locales"[1] man Páginas de manual. misc Manuales en línea locales
/usr/src	En este directorio se recomienda colocar cualquier código fuente. (opcional)	/usr/src/linux Directorio del código fuente del kernel *Linux*.
/usr/X11R6		

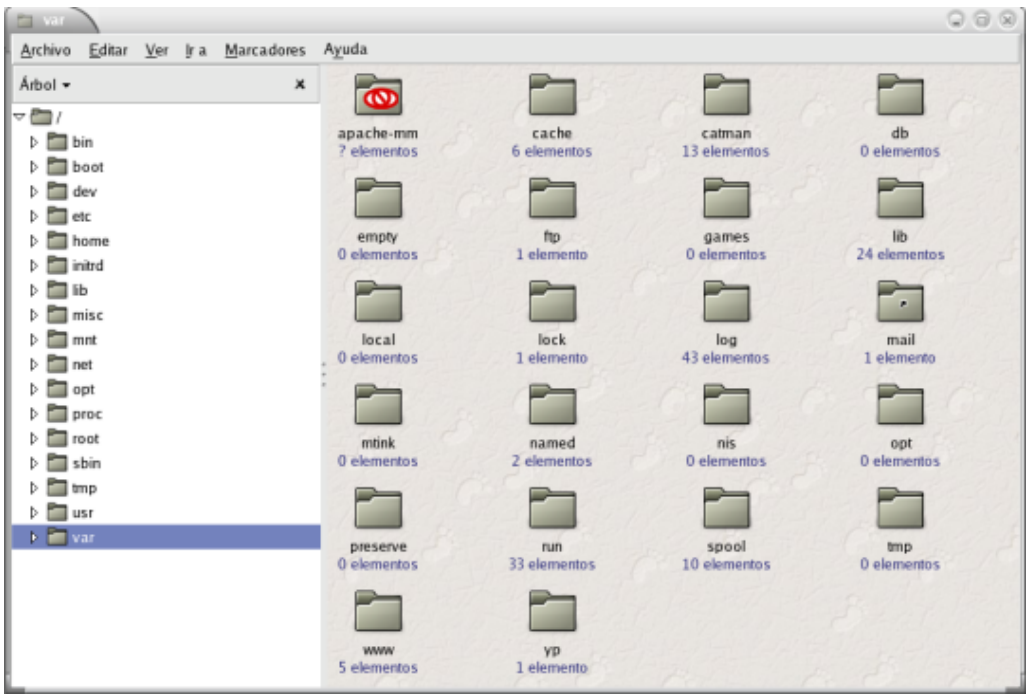
Cuadro 6.9: Jerarquia /usr

6.2.3. LA JERARQUÍA /var

Bajo este directorio se encuentra la información que por lo general es susceptible de cambios (variable). Entre otros inclye los *spools* de correo e impresión, así como los *logs* o registros de eventos del sistema.

[1]Las locales son información acerca de la configuración regional para las aplicaciones, por ejemplo formato de fecha, moneda, etc.

Figura 6.3: Jeraquía */var*



Algunos de los directorios más relevantes en esta jerarquía se resumen en la tabla 6.10:

Directorio	**Descripción**
backups	En este directorio por lo general son guardadas las copias de seguridad. (El FHS le da estatus de reservado, es decir no debe ser utilizado de forma arbitaria debido a que puede generar inconsistencias con otras aplicaciones que simpre han utilizado este directorio)
cache	Aca se mantienen los datos en cahé necesitados por algunas aplicaciones. (obligatorio)
games	Almacena información variable necesitada o utilizada por los juegos. (opcional)
lib	Algunas aplicaciones como bases de datos y otras utilizan este directorio para almacenar los datos con los que estas trabajan pero que no hacen parte de la aplicación (información de estado variable). (obligatorio)
local	Como */var* puede ser exportado para su uso distribuido, en este directorio se mantienen los datos variables que son locales para la máquina. (obligatorio)
lock	Acá se encuentran los archivos de bloqueo de las diferentes aplicaciones. (obligatorio)
log	Se guardan los registros de sucesos del sistema. (obligatorio)
mail	Por lo general en las distribuciones *Linux* este archivo se mantiene como un enlace simbólico a */var/spool/mail* por compatibilidad con el FHS y otros *UNIX*. (obligatorio)

opt	Información de tipo variable para las aplicaciones instaladas en `/opt`. (obligatorio)
run	En este directorio se guarda información relevante acerca del estado de los procesos en ejecución. Todos los indentificadores de procesos (PID) se almacena acá siguiendo la convención `/var/run/<nombre-programa>.pid`. (obligatorio)
spool	La información almacenada acá es guarda para posteriormente algún tipo de procesamiento, por ejemplo su envío al correo o a una impresora. (obligatorio)
tmp	Los datos de carácter temporal que necesiten preservarse entre los reinicios del sistema se almacenan en este directorio. (obligatorio)
www	Algunas distribuciones (como *Debian*) guardan los archivos de las páginas de *Internet*.

Cuadro 6.10: Jerarquia /var

Apéndice A

BIOGRAFIAS

ACLARACIÓN

En este Apendice se incluyen las biografias de algunas de las personas que más han influido en el mundo de los Sistemas Operativos, en especial aquellos vinculados estrechamente con UNIX y/o GNU/Linux.

Estas biografias en su forma original han sido tomadas de la «Wikipedia `http://www.wikipedia.org`»[1] la enclicopedia **libre** más grande del mundo, la traducción al español de estas ha sido hecha por el autor de este libro para poder incluirlas como material complementario, excepto las de Richard Stallman y Linus Torvalds, las cuales ya se encontraban traducidas a esta lengua por parte de la comunidad, aunque es de resaltar que en algunas ocaciones estas traducciones fueron modificadas bien sea en su contenido o en su redacción, además, es importante también aclarar que como los documentos originales están protegidos por la **Licencia de Documentación Libre de GNU** (GFDL), es necesario que este producto derivado (las traducciones) a su vez queden cobijadas

[1] La *Wikipedia*, es un proyecto internacional administrado por voluntarios, con el propósito de crear una enciclopedia libre y gratuita, libre en el sentido GNU, accesible a todo el mundo, colaborativa y en todos los idiomas, de acuerdo con la política de punto de vista neutral. [Esta defición fue tomada de la página inicial de la enciclopedia en castellano]

por esta licencia, por supuesto, esto no implica que el resto del libro sea cobijado por la GFDL, quedando este sujeto a las tradicionales normas de *copyright*. El B en la página 145 incluye una traducción al español de la GFDL, como lo requiere la misma.

A.1. DIJKSTRA, EDSGER

Edsger Wybe Dijkstra (Mayo 30, 1930 - Agosto 6, 2002) fue un científico de la computación nacido en Rotterdam, Holanda y de gran reconocimeinto por sus aportes en está área del conocimiento.

Dijkstra se graduó en física teórica en la Universidad de Leiden y obtuvo un Ph.D. en ciencias de la computación[2]. Trabajó como programador en el «Centro de Matemáticas» de Amsterdam, entre 1952 1962; fue profesor de matemáticas en la Universidad de Tecnología Eindhoven de 1962 a 1984; trabajó como investigador adjunto en la Corporación Burroughs durante 11 años (1973-1984); en la Universidad de Austin Texas ocupó la centenaria plaza «Schlumberger» en ciencias de la computación de 1984 a 1999, y finalmente se retiró como profesor Emerito en 1999.

[2]El término «*computer science*» se puede traducir como ciencias de la computación o como informática, por lo cual estas expresiones se utilizan sin distinción.

Entre sus grandes contribuciones a la informática se encuentra el «algorítmo de camino minimo», también conocido como «algoritmo de Dijkstra», el cual presenta una solución al problema de encontrar el camino más corto en términos de costo entre dos vertices. Es también muy conocido por su baja opinión acerca de la instrucción GOTO en la programación, en 1968 escribió el articulo «Go To Statement Considered Harmful» (La instrucción Go To Considerada Dañosa), el cual es mirado como paso importante hacia la «deprecated» de esta instrucción (la cual fue efectivamente reemplazada por estructuras de control como *do ... while*, *repeat ... until* y otras).

Dijkstra ha enriquecido el lenguaje de la computación con muchos conceptos y frases, tales como programación estructurada, sicronización. la cena de los filósofos, y los famosos «semáforos» (para el control de procesos), entre otras muchas. El Diccionario del Ingles de Oxford cita el uso por parte de este cientifico de palabras como «vector» y «pila» en el contexto de la computación.

En inteligencia artificial demostró una comprensión sutil con su pensamiento «**la pregunta de si un computador puede pensar no es más interesante que la cuestión de si un submarino puede nadar**». Un conocimiento sutil similar es expresado por su declaración: la «**informática no es no más sobre los computadores que la astronomía sobre los telescopios**».

El profesor Edsger Dijstra fue un proligo escritor, una colección de la mayoría de sus manuscritos (algo más de 1300) puede encontarse en el sitio web `http://www.cs.utexas.edu/users/EWD`, donde la Universidad de Texas en Austin mantiene de forma permanente este gran archivo.

Entre los premios que recibió Dijkstra en su carrera se encuentran:

* En 1972, Premio ACM[3] Turing (este premio es considerado el equivalente al premio Nobel en el área de la informática).

[3]ACM = Association for Computing Machinery o Asociación para la Maquinaria de la Computación.

- En 1974, Premio Harry Goode de la AFIPS (Federación Americana de las Sociedades para el Procesamiento de la Información).

- En 1982, Premio a los Pioneros en Computación de la IEEE.

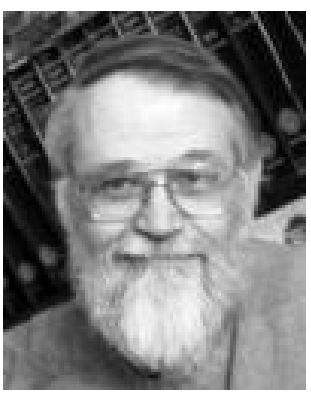
- En 1989, Premio del ACM SIGCSE (Grupo de Interes Especial en la Educación en Informática), por sus Contribuciones Excepcionales en la Educación en Informática.

- En 2001 la Universidad de Economia de Atenas le concedió un doctorado honorario.

- En 2002, la Fundación japonesa C&C, dio a Dijkstra un reconocimiento por «sus contribuciones pioneras al establecimiento de las bases científicas del software a través de la investigación creativa en la teoría básica del software, la teoría del algoritmo, programación estructurada y semáforos».

Edsger Dijkstra murió el 6 de Agosto de 2002 después de una prolongada lucha contra un cancer.

A.2. KERNIGHAN, BRIAN



Brian Kernighan es un cientifico de la computacion, quien trabajó para los Laboratorios Bell. Contribuyó al diseño de los lenguajes de programacion AWK y AMPL.

Su nombre es ampliamente conocido gracias a la co-autoria con Dennis Ritchie de su primer libro «El Lenguaje de Programación C» (*The C programming language*), esto le ha valido para que algunas personas lo vinculen a la creación del Lenguaje C, pero él siempre ha sido enfatico en recalcar que no ha tenido participacion en el diseño de C: «Este es un trabajo enteramente de Dennis Ritchie».

Kernighan es autor de varios programas UNIX, incluyendo *ditroff*. Nació en Toronto, Canada, y recibio su Licenciatura en Físicas de la Ingenieria de la Universidad de Toronto. Recibio un Ph.D. en Ingeniería Electrica en la Universidad de Princeton, donde obtuvo una catedra en el departamento de ciencias de la computacion.

Entre sus escritos destacados se encuentran: «Por que Pascal no es mi lenguaje de programacion favorito».

'Kernighan' se pronuncia (en Ingles) Ker'-ni-han; la 'g' no se pronuncia.

A.3. RITCHIE, DENNIS

Dennis MacAlistair Ritchie (Septiembre 9, 1941) es uno de los pioneros de la computación moderna. Nació en Bronxville, Nueva York. En 1963 se graduó como físico de la Universidad de Hardvard, y en 1968 otuvo en esta misma universidad un Ph.D. en matemáticas.

En 1967 entró a formar parte del Centro de Investigación en Informática de los laboratorios Bell, y en 1968 se unió al equipo de investigación de estos laboratorios que venia trabajando en el desarrollo de *Multics*.

Trabajando junto a *Ken Thompson* y otros desarrolladores, crearon la primera versión del sistema operativo UNIX. Es responsable también de la creación del lenguaje de programación **C**, basado en el lenguaje **B** creado por Thompson. Su libro más conocido y que se ha constituido a través de los tiempos como referencia obligada para la programación en C, es: «El Lenguaje de Programación C», escrito junto a Brian Kernighan.

Entre los premios recibidos por este científico se encuentran:

- 1983 - Premio ACM Turing (junto a Kenneth Thompson).
- 1998 - Medalla Nacional de Tecnología (USA) en compañía de Dennis Ritchie por la creación del sistema UNIX.
- 1989 - Junto a Kenneth Thompson el premio NEC C&C por sus significativas contribuciones a la tecnonología de los computadores.

A.4. STALLMAN, RICHARD

Richard Matthew Stallman (a quien se hace referencia comúnmente por sus iniciales RMS) es una figura central del movimiento de Software Libre. Sus mayores logros como programador incluyen el editor de texto *Emacs*, el compilador *GCC*, y el depurador *GDB*, bajo la rúbrica del Proyecto GNU. Pero su influencia es mayor por el establecimiento de un marco de referencia moral, político y legal para el movimiento de Software Libre, como una alternativa al desarrollo y distribución de software propietario.

Stallman nació en 1953 en Manhattan.

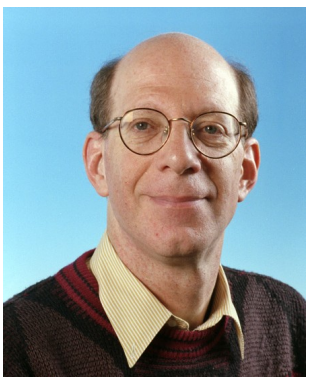
En 1971, siendo estudiante de primer año en la Universidad de Harvard, Stallman se convirtió en un *hacker* del Laboratorio de inteligencia artificial del MIT. En los 1980s, la cultura hacker que constituía la vida de Stallman empezó a disolverse bajo la presión de la comercialización en la industria del software. En particular, otros hackers del Laboratorio de IA fundaron la compañía *Symbolics*, la cual intentaba activamente reemplazar el Software Libre del Laboratorio con su propio software propietario. Por dos años, desde 1983 a 1985, Stallman por sí solo duplicó los esfuerzos de los programadores de Symbolics para prevenir que adquirieran un monopolio sobre los computadores del Laboratorio. Por ese entonces, sin embargo, él era el último de su generación de hackers en el Laboratorio.

Se le pidió que firmara un acuerdo de no revelado (*non-disclosure agreement*) y llevara a cabo otras acciones que él consideró traiciones a sus principios. En 1986, Stallman publicó el Manifiesto GNU, en el cual declaraba sus intenciones y motivaciones para crear una alternativa libre al sistema operativo UNIX, el cual nombró GNU (GNU no es UNIX). Poco tiempo después se incorporó a la organización no lucrativa *Free Software Foundation* para coordinar el esfuerzo. Inventó el concepto de *copyleft* el cual fue utilizado en la Licencia Pública General GNU (conocida generalmente como la "GPL") en 1989. La mayoría del sistema GNU, excepto por el kernel, se completó aproximadamente al mismo tiempo. En 1991, Linus Torvalds liberó el kernel Linux bajo los términos de la GPL, creando un sistema GNU completo y operacional, el sistema operativo GNU/Linux (generalmente referido simplemente como Linux).

Las motivaciones políticas y morales de Richard Stallman le han convertido en una figura controversial. Muchos programadores de influencia que se encuentran de acuerdo con el concepto de compartir el código, difieren con las posturas morales, filosofía personal o el lenguaje que utiliza Stallman para describir sus posiciones. Un resultado de estas disputas condujo al establecimiento de una alternativa al movimiento de Software Libre, el movimiento de código abierto.

Stallman ha recibido numerosos premios y reconocimientos por su trabajo, entre ellos una membrecía en la *MacArthur Foundation* en 1990, el *Grace Hopper Award* de la ACM en 1991 por su trabajo en el editor Emacs original, un doctorado honorario del *Royal Institute of Technology* de Suecia en 1996, el *Pioneer award* de la *Electronic Frontier Foundation* en 1998, el *Yuki Rubinski memorial award* en 1999, y el *Takeda award* en 2001.

A.5. TANENBAUM, ANDREW



Andrew S. "Andy" Tanenbaum (1944-) es la cabeza visible del Departamento de Sistemas de Computo de la Universidad Vrije en Amsterdan (Holanda), donde es profesor de Organización (Estructura) de Computadores y Sistemas Operativos. Recibió su título de pregrado en el MIT y su doctorado en la UC Berkeley. Es bien conocido por sus libros de computación, especialmente:

- Redes de Computadores, ISBN 9702601622
- Sistemas Operativos: Diseño e Implementación, ISBN 9688801534
- Sistemas Operativos Modernos, ISBN 9688803235

Tanenbaum es el autor de Minix, un clónico gratuito de UNIX, el cual sirvió de inspiración para el Sistema Operativo Linux. Se vio involucrado junto a Linus Torvalds (el creador de Linux) en una famosa discusión en Usenet en 1992, acerca del uso de un núcleo (kernel) monolítico para el diseño de Linux en lugar de uno basado en microkernel, lo que Tanenbaum suponía que era la manera del futuro.

A.6. THOMPSON, KENNETH

Kenneth Thompson, nació en 1943 en New Orleans, Louisiana.

Este ingeniero electrico de la Universidad de California en Berkeley entró a trabjar en 1966 a los laboratorios Bell y fue parte del proyecto Multics, en 1970 creó el lenguaje de programación B, a partir del cual Dennis Ritchie creó el Lenguaje **C**. En 1973 reescribe UNIX completamente en el recien creado **C**, siendo el primer sistema operativo escrito en un lenguaje de alto nivel.

Durante un año «sabático» (1975-76) fue profesor vistante en la Universidad de California en Berkeley, algunos de los premios que ha recibido por su trabajo son:

- 1980 – Su juego de ajedréz "Belle" desarrollado junto con Joe H. Condon, ganó el Campeonato de Ajedréz por Computador de Estados Unidos y el Mundo.
- 1983 – Junto a Dennis Ritchie recibe el premio ACM Turing.
- 1998 – Recibe la Medalla Nacional de Tecnología (USA) en compañía de Dennis Ritchie por la creación del sistema UNIX.

A.7. TORVALDS, LINUS

Linus Benedict Torvalds es el creador del núcleo (*kernel)* del sistema operativo GNU/Linux, a menudo llamado simplemente Linux.

Nacido en Helsinki, Finlandia, en 1969.

Comenzó sus andaduras informáticas a la edad de 11 años. Su abuelo, un matemático y estadístico de la Universidad se compró uno de los primeros Commodore en 1980 y le pidió ayuda para usarlo.

En 1988 Linus es admitido en la Universidad de Helsinki. Ese mismo año Andy Tannenbaum saca a la luz el S.O. Minix. En 1990 empieza Torvalds a aprender C en sus estudios.

A finales de los 80 tomó contacto con los computadores IBM/PC compatibles y en 1991 adquirió un 80386. A la edad de 21, con 5 años de experiencia

programando (uno en C), ya conocía lo bastante del S.O. MS-DOS como para tomarle algunas ideas prestadas y empezar un proyecto personal: basándose en *Design of the UNIX O.S.* y modificando gradualmente el núcleo del Minix crearía una adaptación del potente S.O. que ejecutara el software de GNU, pero sobre PC.

Este proyecto personal desembocó en octubre de 1991 en el anuncio de la primera versión de Linux capaz de ejecutar el BASH (Bourne Again Shell) y el GCC (GNU C Compiler), pero poco más. En enero de 1992 se adoptó la GPL (Licencia Pública General) para Linux. Esta no prohibe vender su código fuente, pero que debe ser susceptible de ser modificado por el usuario final. De hecho, miles de programadores en el mundo colaboran en el desarrollo de este entorno.

En 1997 Linus Torvalds recibe los premios '1997 Nokia *Foundation Award 'Lifetime Achievement Award at Uniforum Pictures'*. Ese mismo año finaliza los estudios superiores (1988 - 1997) tras 10 años como estudiante e investigador en la Universidad de Helsinki, coordinando el desarrollo del núcleo del S.O. desde 1992. Ahora Torvalds trabaja en Silicon Valley (EE.UU.). Solo el 2 % de Linux fue creado por él en los 90, pero en su persona sigue descansando la paternidad de este revolucionario S.O.

Apéndice B

Traducción a Español de la GNU Free Documentation License 1.1 (GFDL)

Los autores de esta traducción son:

- Igor Támara `ikks@bigfoot.com`
- Pablo Reyes `reyes_pablo@hotmail.com`

Esta es una traducción no oficial de la GNU Free Document License (GFDL), versión 1.1 a Español. No ha sido publicada por la Free Software Foundation, y no establece legalmente los términos de distribución para trabajos que usen la GFDL – sólo el texto de la versión original en Inglés de la GFDL lo hace. Sin embargo, esperamos que esta traducción ayudaré a hablantes de español a entender mejor la GFDL.

This is an unofficial translation of the GNU Free Document License (GFDL) into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for works that uses the GFDL –only the original English text of the GFDL does that. However, we hope that this translation will help Spanish speakers understand the GFDL better.

La versión original de la GFDL esta disponible en: `http://www.gnu.org/copyleft/fdl.html`

Licencia de Documentación Libre GNU

Versión 1.1, Marzo de 2000

0. PREÁMBULO

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea "libre" en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien publica una manera de obtener reconocimiento por su trabajo, al tiempo que no se considera responsable de las modificaciones realizadas por terceros, lo cual significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Complementa la Licencia Pública General GNU, que es una licencia de copyleft[1] diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: Un programa libre debe venir con los manuales que ofrezcan la mismas libertades que da el software. Pero esta licencia no se limita a manuales de software; puede ser usada para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

1. APLICABILIDAD Y DEFINICIONES

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indique que puede ser distribuido bajo los términos de la Licencia. El "Documento", en adelante, se refiere a culquiera de dichos manuales o trabajos. Cualquier miembro del público es un como "Usted".

[1] **N. del T.** copyleft es un nuevo término acuñado por GNU. Nace de un juego de palabras en Inglés, en vez de "copyright" usan "copyleft", indicando que no se restringe la copia, sino por el contrario se permite mientras que el derecho a seguir copiando no se restrinja. `http://www.gn\u.org/copyleft/copyleft.es.html`

Una "Versión Modificada" del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una "Sección Secundaria" es un apéndice titulado o una sección preliminar al prólogo del Documento que tiene que ver exclusivamente con la relación de quien publica, o con los autores del Documento, o con el tema general del Documento (o asuntos relacionados) y cuyo contenido no entre directamente en tal tema general. (Por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar matemáticas). La relación puede ser un asunto de conexión histórica, o de posición legal, comercial, filosófica, ética o política con el tema o la materia del texto.

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos son denominados como Secciones Invariantes en la nota que indica que el documento es liberado bajo esta licencia.

Los "Textos de Cubierta" son ciertos pasajes cortos de texto que se listan, como Textos de Título o Textos al respaldo de la página de tí, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia "Transparente" del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público general, cuyos contenidos pueden ser vistos y editados directamente con editores de texto genéricos o (para imágenes compuestas por pixeles) con programas genéricos de dibujo o (para dibujos) con algún editor gráfico ampliamente disponible, y que sea adecuado para exportar a formateadores de texto o para traducción automática a una variedad de formatos adecuados para ingresar a formateadores de texto. Una copia hecha en un formato de un archivo que no sea Transparente, cuyo formato ha sido diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es Transparente. Una copia que no es "Transparente" es llamada "Opaca".

Como ejemplos de formatos adecuados para copias Transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible publicamente, y HTML simple que siga los estándares, PostScript o PDF diseñado para modificaciones humanas. ejemplos de formatos de imágenes transparentes incluyen PNG, XCF y JPG. Los formatos Opacos incluyen formatos propietarios que pueden ser leídos y editados unicamente en procesadores de palabras propietarios, SGML o XML para los cuáles los DTD y/o herramientas de procesamiento no están disponibles generalmente, y el HTML generado por máquinas, PostScript o PDF producto de algún procesador de palabras solo para propósitos de salida.

La "Pagina de Título" en un libro impreso significa, la pagina de titulo misma, más las páginas siguientes que sean necesarias para mantener, legiblemente, el material que esta Licencia requiere en la página de titulo. Para trabajos en formatos que no tienen página de título como tal, "Página de Título" significa el texto cercano a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del trabajo.

2. COPIA LITERAL

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de derecho de autor, y la nota de licencia que indica que esta Licencia se aplica al Documento se reproduzca en todas las copias, y que usted no adicione ninguna otra condición a las expuestas en en esta Licencia. Usted no puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias publicamente.

3. COPIADO EN CANTIDAD

Si publica copias impresas del Documento que sobrepasen las 100, y la nota de Licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible, todos esos Textos de Cubierta: Textos de título en la portada, y Textos al respaldo de la página de tílo. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como quien publica tales copias. La portada debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede adicionar otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, pueden considerarse como copias literales.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la cubierta real y continuar el resto en páginas adyacentes.

Si publica o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente, que pueda ser leída por una máquina, con cada copia Opaca o entregar en o con cada copia Opaca una dirección en una red de computadores accesible publicamente que contenga una copia completa Transparente del Documento, sin material adicional, a la cual el público en general de la red pueda acceder a bajar anónimamente sin cargo usando protocolos públicos y estandarizados. Si usted hace uso de la última opción, deberá tomar medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio por lo menos un año después de su última distribución de copias Opacas (directamente o a través de sus agentes o distribuidores) de esa edición al público.

Se solicita, aunque no es requisito, que contacte a los autores del Documento antes de redistribuir cualquier gran número de copias, para darle la oportunidad de que le provean una versión actualizada del Documento.

4. MODIFICACIONES

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las seccions 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto licenciando la distribución y modificación de la Versión Modificada a quienquiera posea una copia de esta. Además, debe hacer lo siguiente en la Versión Modificada:

- **A.** Usar en la Página de Título (y en las cubiertas, si hay alguna) un título distinto al del Documento, y de versiones anteriores (que deberían, si hay alguna, estar listados en la sección de Historia del Documento). Puede usar el mismo título de versiones anteriores al original siempre y cuando quien publicó originalmente otorge permiso.
- **B.** Listar en la Página de Título, como autores, una o más personas o entidades responsables por la autoría o las modificaciones en la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (Todos sus autores principales, si hay menos de cinco).
- **C.** Incluir en la Página de Título el nombre de quién publica la Versión Modificada, como quien publica.
- **D.** Preservar todas las notas de derechos de autor del Documento.

- **E.** Adicionar una nota de derecho de autor apropiada a sus modificaciones, adyacente a las otras notas de derecho de autor.

- **F.** Incluir, immediatamente después de la nota de derecho de autor, una nota de licencia dando el permiso público para usar la Versión Modificada bajo los términos de esta Licencia, de la forma mostrada en el anexo al final de este documento.

- **G.** Preservar en esa nota de licencia el listado completo de Secciones Invariantes y de los Textos de Cubiertas que sean requeridos como se especifique en la nota de Licencia del Documento

- **H.** Incluir una copia sin modificación de esta Licencia.

- **I.** Preservar la sección llamada "Historia", y su título, y adicionar a esta una sección estableciendo al menos el título, el año,los nuevos autores, y quién publique la Versión Modificada como reza en la Página de Título. Si no hay una sección titulada "Historia" en el Documento, crear una estableciendo el título, el año, los autores y quieng publicó el Documento como reza en la Página de Título, añadiendo además una sección describiendo la Versión Modificada como se estableció en la oración anterior.

- **J.** Preservar la localización en red, si hay , dada en la Documentación para acceso público a una copia Transparente del Documento, tanto como las otras direcciones de red dadas en el Documento para versiones anteriores en las cuáles estuviese basado. Estas pueden ubicarse en la sección "Historia". Se puede omitir la ubicación en red para un trabajo que sea publicado por lo menos cuatro años antes que el Documento mismo, o si quien publica originalmente la versión da permiso explícitamente.

- **K.** En cualquier sección titulada "Agradecimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de los agradeimientos y/o dedicatorias de cada contribuyente que estén incluídas.

- **L.** Preservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección.

- **M.** Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

- **N.** No retitular ninguna sección existente como "Aprobaciones" o conflictuar con el título de alguna Sección Invariante.

Si la Versión Modificada incluye secciones o apendices nuevos o preliminares al prólogo que califiquen como Secciones Secundarias y contienen material no copiado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, adicione sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede adicionar una sección titulada "Aprobaciones", siempre que contenga unicamente aprobaciones de su Versión Modificada por varias fuentes–por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como la definición oficial de un estándar.

Puede adicionar un pasaje de hasta cinco palabras como un Texto de Portada, y un pasaje de hasta 25 palabras al respaldo de la página de título al final de la lista de Textos de Cubierta en la Versión Modificada. Solamente un pasaje de Texto de Portada y uno al respaldo de la página de título puede ser adicionado por (o a manera de arreglos hechos por) una entidad. Si el Documento ya incluye un texto de cubierta para la misma portada o al respaldo de la portada,

previamente adicionado por usted o por arreglo hecho por la misma entidad, a nombre de la cual usted está actuando, usted no puede adicionar otro; pero puede reemplazar el anterior, con permiso explícito de quien publicó previamente y agregó el texto anterior

El(los) autor(es) y quien(es) publica(n) el Documento no dan con esta Licencia permiso para usar sus nombres para publicidad o para asegurar o implicar aprobación de cualquier Versión Modificada.

5. COMBINANDO DOCUMENTOS

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, y listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y múltiples Secciones Invariantes idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título de cada una de estas secciones único adicionándole al final del mismo, entre paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección titulada "Historia" de los varios documentos originales, formando una sección titulada "Historia"; de la misma forma combine cualquier seción titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Debe borrar todas las secciones tituladas "Aprobaciones."

6. COLECCIONES DE DOCUMENTOS

Puede hacer una colección consistente del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para cada copia literal de cada uno de los documentos en cualquiera de los demás aspectos.

Puede extraer un solo documento de una de tales colecciones, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraido, y siga esta Licencia en todos los otros aspectos concernientes a la copia literal de tal documento.

7. AGREGACIÓN CON TRABAJOS INDEPENDENTES

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, cuenta como una Versión Modificada del Documento,

siempre que no se aleguen derechos de autor por la recopilación. Tal recopilación es llamada un "agregado", y esta Licencia no aplica a los otros trabajos auto-contenidos así recopilados con el Documento, o a cuenta de haber sido recopilados, si no son ellos mismos trabajos derivados del Documento.

Si el requerimiento de la sección 3 sobre el Texto de Cubierta es aplicable a estas copias del Documento, entonces si el Documento es menor que un cuarto del agregado entero, los Textos de Cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento entre el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

8. TRADUCCIÓN

La Traducción es considerada como una clase de modificación, Así que puede distribuir traducciones del Documento bajo los términos de la sección 4. Reemplazar las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero usted puede incluir traducciones de algunas o todas las Secciones Invariantes adicionalmente a las versiones originales de tales Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre que incluya también la versión en Inglés de esta Licencia. En caso de un desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la versión original en Inglés prevalecerá.

9. TERMINACIÓN

No se puede copiar, modificar, sublicenciar, o distribuir el Documento excepto por lo permitido expresamente bajo esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o distribución del Documento es nulo, y serán automáticamente terminados sus derechos bajo esa licencia. Sin embargo, los terceros que hayan recibido copias, o derechos, de su parte bajo esta Licencia no tendrán por terminadas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

10.REVISIONES FUTURAS DE ESTA LICENCIA

La Free Software Foundation puede publicar versiones nuevas y revisadas de la Licencia de Documentación Libre GNU de tiempo en tiempo. Tales nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar nuevos problemas o intereses. Vea `http://www.gnu.org/copyleft/`.

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que se aplica una versión numerada en particular de esta licencia o "cualquier versión posterior", usted tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que haya sido publicada (no como un borrador) por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada(no como un borrador) por la Free Software Foundation.

Cómo usar esta Licencia para sus documentos

Para usar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y ponga el siguiente derecho de autor y nota de licencia justo después de la página de título:

> Derecho de Autor (c) AÑO SU NOMBRE. Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre GNU, Versión 1.1 o cualquier otra versión posterior publicada por la Free Software Foundation; con las Secciones Invariantes siendo LISTE SUS TÍTULOS, con los Textos de Portada siendo LISTELO, y con los Textos al respaldo de la página de título siendo LISTELOS. Una copia de la licencia es incluida en la sección titulada "Licencia de Documentación Libre GNU".

Si no tiene Secciones Invariantes, escriba "Sin Secciones Invariantes" en vez de decir cuáles son invariantes. Si no tiene Texto de Portada, escriba "Sin Texto de Portada" en vez de "con los Textos de Portada siendo LISTELO"; e igualmente para los Textos al respaldo de la página de título.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar estos ejemplos en paralelo bajo su elección de licencia de software libre, tal como la Licencia Pública General GNU (GNU General Public License), para permitir su uso en software libre.

Traducción: Igor Támara P. `ikks@bigfoot.com` Pablo Reyes `reyes_pablo@hotmail.com`

Revisión 1 (25.May.2000): Vladimir Támara Patiño `vtamara@gnu.org`

Bibliografía

[AIVAZIAN-02in] AIVAZIAN, Tigran. *Linux 2.4 Kernel Internals* **[en línea]**. s.l.: The Linux Documentation Project, 7 ago. 2002 <http://www.moses.uklinux.net/patches/lki.html>[Consulta: julio 2003]

[AIVAZIAN-02es] AIVAZIAN, Tigran. *Dentro del núcleo Linux 2.4* **[en línea]**. Traducción de Rubén Melcón. s.l.: Projecto LUCAS (TLDP-ES), 27 oct. 2002 <http://es.tldp.org/Manuales-LuCAS/DENTRO-NUCLEO-LINUX/dentro-nucleo-linux-html>[Consulta: julio 2003]

[CATALINA-99] CATALINA GALLEGO, Miguel y CATALINA GALLEGO, Alfredo. *UNIX/Linux: Iniciación y referencia*. 1 ed. Madrid: McGraw-Hill Interamericana de España, 1999. (Serie Iniciación y Referencia). ISBN 84-481-2101-5

[DEBIAN-03] DESARROLLADORES DEL SISTEMA DEBIAN GNU/LINUX. *Contrato Social Debian* **[en línea]**. s.l.:Debian, 1997, 29 jun. 2003. <http://www.debian.org/social_contract>[Consulta: jul 2003]

[ESTIVILL-97] ESTIVILL, Assumpció y URBANO, Cristobal. *Cómo citar recursos electrónicos* **[en línea]**. Ver. 1.0. Barcelona: Escola Universitària Jordi Rubió i Balaguer de

Biblioteconomia i Documentació (Univesitat de Barcelona), 30 mayo 1997. <http://www.ub.es/biblio/citae-e.htm>[Consulta: jul. 2003]

[FHS-2.2] FILESYSTEM HIERARCHY STANDARD GROUP. *Filesystem Hierarchy Standard* **[en línea]**. Rusty Russell, Daniel Quinlan. Ver. 2.2. s.l.: Freestandards.org, 6 nov. 2001, 24 mayo 2001. <http://www.pathname.com/fhs/>[Consulta: mayo 2003]

[PACHÓN-01] PACHÓN L., Wilfredo I. *Guía didáctica de apoyo para módulo de informática*, Corporación Unificada Nacional de Educación Superior. 1 ed. Bogotá: Filigrana, 2001.

[RAYMOND-03] RAYMOND, Eric S. *The Jargon File* **[en línea]**. Ver. 4.4.2. s.l.: 22 mayo 2003. «Hacker». <http://catb.org/~esr/jargon/html/H/hacker.html>[Consulta: jun 2003]

[SHAH-01] SHAH, Steve. *Manual de administración Linux*. 1 ed. Madrid: McGraw-Hill Interamericana de España, 2001. ISBN 01-07-0212229-3

[STALLINGS-97] STALLINGS, William. *Sistemas Operativos*. 2 ed. Madrid: Prentice Hall, 1997. ISBN 84-89660-22-0

[STALLMAN-98] STALLMAN, Richard M. *El Proyecto GNU* **[en línea]**. Traducción de César Ballardini. s.l.: Free Software Foundation, 1998. <http://www.gnu.org/gnu/thegnuproject.es.html>[Consulta: Abr. 2003]

[STRAUSS-98] STRAUSS, Daryll. *Linux Helps Bring Titanic to life* **[en línea]**. s.l.:Linux Journal, 01 feb 1998. <key-50http://www.linuxjournal.com/article.php?sid=2494>[Consulta: Abr. 2003]

[TANENBAUM-93] TANENBAUM, Andrew S., *Sistemas Operativos Modernos*. 1 ed. Naucalpan de Juárez, Mexico: Prentice Hall Hispanoamericana, 1993. ISBN 968-880-323-5

[TANENBAUM-97] TANENBAUM, Andrew S. & WOODHULL, Albert S., *Operating Systems: Design and Implementation*. 2 ed. SD: Prentice key-50Hall, 15 ene. 1997. ISBN 013-638-677-6

[PASC] THE PORTABLE APPLICATION STANDARDS COMMITTE, *Official Web Site* **[en línea]**. IEEE. s.l.: The Open Group, 1995-2003. «What is POSIX». <http://www.pasc.org/#POSIX>[Consulta: Jun. 2003]

[TORVALDS-91] TORVALDS, Linus B. *What would you like to see most in minix* **[en línea]** . En: [**Minix**]. 25 ago. 1991, message id.: <news:1991Aug25.205708.9541@klaava.Helsinki.FL>. Usenet newsgroup <news:comp.os.minix>. Mensaje archivado en: <http://groups.google.com/groups?selm=1991Aug25.205708.9541%40klaava.Helsinki.FI>[Consulta: Abr. 2003]