1. Caso de estudio: UNIX-LINUX

1. Historia

1.1 UNIX

Unix es uno de los sistemas operativos más ampliamente usados en computadoras que varían desde las personales hasta los mainframe. Existen versiones para máquinas uniprocesador hasta multiprocesadores. Debido a su historia, que evoluciona en los Laboratorios Bell de AT&T con un simulador de un viaje espacial en el sistema solar, pasando por su expansión en universidades y la creación de las versiones más importantes que son la de la Universidad de Berkeley y el Sistema V de la misma AT&T.

Creado por un equipo del Laboratorio Bell de la AT&T a principios de los 70: Ken Thompson y Dennis Ritchie basado en el proyecto MULTICS, que había quedado abandonado.

Después de una primera versión, en ensamblador PDP-7 denominada UNICS que posteriormente pasó a llamarse UNIX, se rescribe en un lenguaje de alto nivel, llamado **B** y posteriormente, Thompson y Ritchie desarrollan el **C** y lo vuelven a rescribir. En 1976 se difunden gratuitamente los fuentes de UNIX por las universidades de EUA (Versión 6).

1978: Versión 7

1978-8x: Versión de Berkeley (BSD) - memoria virtual, utilidades, soporte de redes (TCP, sockets)

1983: System V

198x: formación de consorcios para fijación de estándares: XPG, OSF, etc. Normas POSIX. IBM y DEC empiezan a emplear UNIX

Finales de 198x: las versiones 4.3BSD y SystemV R3 son incompatibles.

199X: Linux (UNIX gratuito)

1.1 Estandarización de UNIX

Debido a las múltiples versiones en el mercado de UNIX, se comenzaron a publicar estándares para que todas la s versiones fuesen 'compatibles'. La primera de ellas la lanzó AT&T llamada SVID (System V Interface Definition) que definiría cómo deberían ser las llamadas al sistema, el formato de los archivos y muchas cosas más, pero la otra versión importante, la de Berkeley (Berkeley Software Distribution o BSD) simplemente la ignoró. Después la IEEE usó un algoritmo consistente en revisar las llamadas al sistema de ambas versiones (System V y BSD) y aquellas que eran iguales las definió como estándares surgiendo así la definición **'Portable Operating System for UNIX'** o **POSIX**, que tuvo buen éxito y que varios fabricantes adoptaron rápidamente. El estándar de POSIX se conoce como **1003.1**. Posteriormente los institutos ANSI e ISO se interesaron en estandarizar el lenguaje 'C' y conjuntamente se publicaron definiciones estándares para otras áreas del sistema operativo como la interconectividad, el intérprete de comandos y otras. En la tabla 1 se muestran las definiciones de POSIX.

Estándar	**Descripción**
1003.0	Introducción y repaso.
1003.1	Llamadas al sistema. (procedimientos de biblioteca)
1003.2	Intérprete y comandos.
1003.3	Métodos de prueba.

1003.4 Extensiones para tiempo real.

1003.5 Lenguaje Ada.

1003.6 Extensiones para la seguridad

1003.7 Administración del Sistema.

1003.8 Acceso transparente a archivos.

1003.9 Lenguaje Fortran.

1003.10 Supercómputo.

Tabla 1 Las Especificaciones de POSIX

Al momento del auge de los estándares de POSIX desgraciadamente se formó un grupo de fabricantes de computadoras (IBM, DEC y Hewlett-Packard) que lanzaron su propia versión de UNIX llamada **OSF/1** (de **Open Software Fundation**). Lo bueno fue que su versión tenía como objetivo cumplir con todas los estándares del IEEE, además de un sistema de ventanas (el **X11**), una interfaz amigable para los usuarios (**MOTIF**) y las definiciones para cómputo distribuido (**DCE**) y administración distribuida (**DME**). La idea de ofrecer una interfaz amigable en UNIX no fue original de OSF, ya en la versión 3.5 de SunOS de Sun Microsystems se ofrecía una interfaz amigable y un conjunto de librerías para crear aplicaciones con interfaz gráfica técnicamente eficiente y poderosa llamada SunWindows o SunVIEW. Esta interfaz junto con sus librerías estaban evolucionando desde la versión para máquinas aisladas hacia una versión en red, donde las aplicaciones podían estarse ejecutando en un nodo de la red y los resultados gráficos verlos en otro nodo de la red, pero Sun tardó tanto en liberarlo que le dio tiempo al MIT de lanzar el X11 y ganarle en popularidad.

AT&T formó, junto con Sun Microsystems y otras compañias UNIX International y su versión de UNIX, provocando así que ahora se manejen esas dos corrientes principales en UNIX.

1.2 LINUX

El sistema operativo Linux se genera inspirándose en dos sistemas operativos, el sistema abierto UNIX creado en 1969 por Ken Thompson y Dennis Ritchie en los laboratorios de Bell. De este sistema se toman sus características, especificaciones y funcionamiento. Mas el sistema educativo Minix creado en 1987 por Andrew S. Tanenbaum del cual se toma la estructura y código del núcleo. Con todo esto en 1991 Linus Torvalds crea Linus's Unix = Linux Kernel, esto es crea solo el núcleo del sistema sin la capa de servidores, manejadores, aplicaciones graficas, etc. que serán creadas posteriormente por otros autores. El código del núcleo lo podemos encontrar en la dirección (www.kernel.org). El núcleo actual tiene aproximadamente 1,5 millones de líneas de código, y representa menos del 50 por ciento de todo el código del sistema.

En la comunidad de programadores se crea el proyecto GNU (Gnu's Not Unix), proyecto para generar software libre, donde se generan editores Emacs, compiladores gcc, interprete de comandos bsh, sistema operativo Hurd, aplicaciones, etc., bajo la licencia publica general GPL (General Public License), usar, copiar, distribuir y modificar (con las mismas condiciones). Se conserva la firma del autor. Se puede cobrar.

Linux se crea con esta filosofía de libre distribución y el sistema operativo completo que se construye con este núcleo también. A todo el sistema se le da el nombre de GNU/Linux (distribución completa del sistema operativo con Linux), que contiene el núcleo (1,5 millones de líneas de código) mas las otras capas del sistema operativo y utilidades. Si bien muchas veces se denomina a todo el sistema simplemente LINUX.

2. Visión General

2.1 UNIX

El núcleo de UNIX (kernel) se clasifica como de tipo monolítico, pero en él se pueden encontrar dos partes principales: el núcleo dependiente de la máquina y el núcleo independiente. El **núcleo dependiente** se encarga de las interrupciones, los manejadores de dispositivos de bajo nivel (lower half) y parte del manejo de la memoria. El **núcleo independiente** es igual en todas las plataformas e incluye el manejo de llamadas del sistema, la planificación de procesos, el entubamiento, el manejo de señales, la paginación e intercambio, el manejo de discos y del sistema de archivos.

Las ideas principales de UNIX fueron derivadas del proyecto MULTICS (Multiplexed Information and Computing Service) del MIT y de General Electric. Estas ideas son:

Todo se maneja como cadena de bytes: Los dispositivos periféricos, los archivos y los comandos pueden verse como secuencias de bytes o como entes que las producen. Por ejemplo, para usar una terminal en UNIX se hace a través de un archivo (generalmente en el directorio /dev y con nombre ttyX).

Manejo de tres descriptores estándares: Todo comando posee tres descriptores por omisión llamados **'stdin'**, **'stdout'** y **'stderr'**, los cuales son los lugares de donde se leen los datos de trabajo, donde se envían los resultados y en donde se envían los errores, respectivamente. El 'stdin' es el teclado, el 'stdout' y el 'stderr' son la pantalla por omisión (default).

Capacidades de 'entubar' y 'redireccionar': El 'stdin', 'stdout' y el 'stderr' pueden usarse para cambiar el lugar de donde se leen los datos, donde se envían los resultados y donde se envían los errores, respectivamente. A nivel comandos, el símbolo de **'mayor que' (>)** sirve para enviar los resultados de un comando a un archivo. Por ejemplo, en UNIX el comando 'ls' lista los archivos del directorio actual (es lo mismo que 'dir' en DOS). Si en vez de ver los nombres de archivos en la pantalla se quieren guardar en el archivo 'listado', el redireccionamiento es útil y el comando para hacer la tarea anterior es 'ls > listado'. Si lo que se desea es enviar a imprimir esos nombres, el 'entubamiento' es útil y el comando sería 'ls | lpr', donde el símbolo **"|" (pipe)** es el entubamiento y 'lpr' es el comando para imprimir en UNIX BSD.

Crear sistemas grandes a partir de módulos: Cada instrucción en UNIX está diseñada para poderse usar con 'pipes' o 'redireccionamiento', de manera que se pueden crear sistemas complejos a través del uso de comandos simples y elegantes. Un ejemplo sencillo de esto es el siguiente. Suponga que se tienen cuatro comandos separados A, B, C y D cuyas funcionalidades son:

A: lee matrices comprobando tipos de datos y formato.

B: recibe matrices, las invierte y arroja el resultado en forma matricial.

C: recibe una matriz y le pone encabezados 'bonitos'

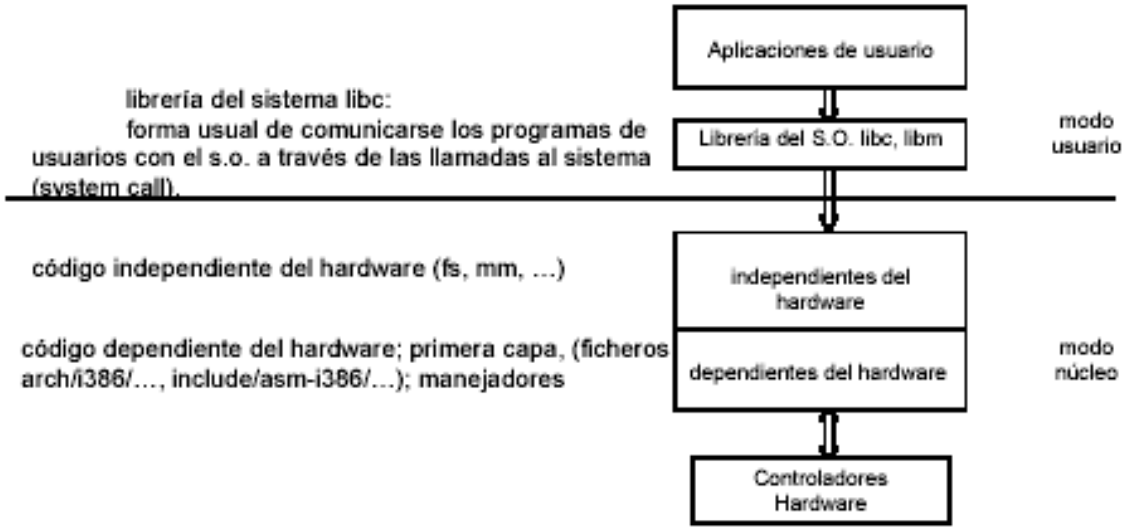
D: manda a la impresora una matriz cuidando el salto de página, etc.

Como se ve, cada módulo hace una actividad específica, si lo que se quiere es un pequeño sistema que lea un sistema de ecuaciones y como resultado se tenga un listado 'bonito', simplemente se usa el entubamiento para leer con el módulo A la matriz, que su resultado lo reciba el B para obtener la solución, luego esa solución la reciba el módulo C para que le ponga los encabezados 'bonitos' y finalmente eso lo tome el módulo D y lo imprima, el comando completo sería ' A | B | C | D '. ¿Fácil no?

2.2 LINUX

CARACTERISTICAS DEL NÚCLEO

- Su código es de libre uso.
- Escrito en lenguaje C, compilado con gcc, primera capa en ensamblador.
- Ejecutable en varias plataformas hardware.
- Se ejecuta en máquinas con arquitectura de 32 bits y 64 bits.
- Estaciones de trabajo y servidores.
- Código y funcionamiento escrito bajo la familia de estándares POSIX (Portable Operating System Interface).
- Soporta CPU's con uno o varios microprocesadores (SMP) symmetric multiprocessing.
- Multitarea.
- Multiusuario.
- Gestión y protección de memoria.
- Memoria virtual.
- Varios sistemas de ficheros.
- Comunicación entre procesos (señales, pipes, IPC, sockets).
- Librerías compartidas y dinámicas.
- Permite trabajar en red TCP/IP.
- Estable, veloz, completo y rendimiento aceptable.
- Funcionalmente es muy parecido a UNIX.



- Mas de 100.000 usuarios.
- Actualizado, mejorado, mantenido y ampliado por la comunidad de usuarios

(modelo "bazar", contrapuesto al modelo "catedral").

Linux mantiene una estructura monolítica, pero admite módulos cargables.

3. Procesos y Threads

El manejo de procesos en UNIX es por **prioridad** y **round robin**. En algunas versiones se maneja también un **ajuste dinámico de la prioridad** de acuerdo al tiempo que los procesos han esperado y al tiempo que ya han usado el CPU. El sistema provee facilidades para contabilizar el uso de CPU por proceso y una pila común para todos los procesos cuando necesitan estarse ejecutando en modo privilegiado (cuando hicieron una llamada al sistema). UNIX permite que un proceso haga una **copia de sí mismo** por medio de la llamada **'fork'**, lo cual es muy útil cuando se realizan trabajos paralelos o concurrentes; también se proveen facilidades para el envío de **mensajes entre procesos (pipes, signals)**.

Los procesos no interactivos se denominan **daemons** o procesos background. Cuando se inicia un proceso, se le asigna un identificador **PID**, se guarda el proceso que lo lanzó **PPID**, el propietario que lo lanzó **UID** y el grupo de pertenencia **GID**, lo que definirá el perfil de permisos de acceso a los que tendrá derecho. Existe la posibilidad de alterar el usuario o grupo efectivo de permisos, durante la ejecución del proceso, mediante la llamada a **setuid o setgid**, siempre que se disponga de los permisos apropiados. También existe una bandera de permisos **setuid** asociada al archivo del programa que permite ejecutar éste, con los permisos del propietario del archivo en lugar de los del usuario que lo ejecuta.

LINUX combina multiprogramación y tiempo compartido. Es un sistema preemptive, con planificación dependiente de la categoría del proceso: tiempo real o proceso ordinario. Sobre procesos ordinarios emplea la técnica de planificación por prioridad dinámica (la asignada inicialmente más una variable asignada por el sistema). En tiempo real se aplica un planificador FIFO non-preemptive para ciertas tareas de igual prioridad o Round Robin preemptive en otros casos, también de tiempo real.

4. Gestión de Memoria

Los primeros sistemas con UNIX nacieron en máquinas cuyo espacio de direcciones era muy pequeño (por ejemplo 64 kilobytes) y tenían un manejo de memoria real algo complejo. Actualmente todos los sistemas UNIX utilizan el manejo de memoria virtual siendo el esquema más usado la **paginación por demanda** y **combinación de segmentos paginados**, en ambos casos con páginas de tamaño fijo. En todos los sistemas UNIX **se usa una partición de disco duro para el área de intercambio**. Esa área **se reserva al tiempo de instalación** del sistema operativo. Una regla muy difundida entre administradores de sistemas es asignar una partición de disco duro que sea al menos el doble de la cantidad de memoria real del ordenador. Con esta regla se permite que se puedan intercambiar flexiblemente todos los procesos que estén en memoria RAM en un momento dado por otros que estén en el disco. Todos **los procesos que forman parte del kernel no pueden ser intercambiados** a disco. Algunos sistemas operativos (como SunOS) permiten incrementar el espacio de intercambio incluso mientras el sistema está en uso (en el caso de SunOS con el comando 'swapon'). También es muy importante que al momento de decidirse por un sistema operativo se pregunte por esa facilidad de incrementar el espacio de intercambio, así como la facilidad de añadir módulos de memoria RAM a la computadora sin necesidad de reconfigurar el núcleo.

Cada proceso dispone de su propio espacio de direcciones, organizado en segmentos según: **Text Segment**, código; **Data Segment**, datos; **Initialized Data**; **Uninitialized Data**; **Stack**.

Es posible compartir código entre procesos mediante el empleo de **Shared Text Segments**. Dos procesos nunca comparten los segmentos de datos y de pila (salvo los thread), la

forma de compartir información se lleva a cabo mediante el empleo de segmentos especiales de memoria compartida **Shared Segments**.

Swapper (PID 0)

Page Daemon (PID 2)

Tabla de páginas = **core map**

Reemplazo: cada 250ms el page daemon mira si el número de marcos libres no baja de cierta cantidad, (aprox ¼ del tamaño de la memoria)

Si hay reemplazo el swapper desaloja uno o varios procesos que hayan estado inactivos durante más de 20s.

En LINUX, la memoria virtual se organiza en 3 partes o niveles de páginas.

dir virtual: (p1 p2 p3 off)

5. Entrada/Salida

Derivado de la filosofía de manejar todo como flujo de bytes, los dispositivos son considerados **como archivos** que se acceden mediante descriptores de archivos cuyos nombres se encuentran generalmente en el directorio '/dev'. Cada proceso en UNIX mantiene una tabla de archivos abiertos (donde el archivo puede ser cualquier dispositivo de entrada/salida). Esa tabla tiene entradas que corresponden a los descriptores, los cuales son números enteros obtenidos por medio de la llamada del sistema 'open'. En la tabla 2 se muestran las llamadas más usuales para realizar entrada/salida.

Llamada	**Función**
open	Obtener un descriptor entero.
close	Terminar las operaciones sobre el archivo
lseek	Posicionar la entrada/salida.
read,write	Leer o escribir al archivo (dispositivo)
ioctl	Establecer el modo de trabajo del dispositivo

Tabla 2 Llamadas al sistema de entrada/salida

En UNIX es posible ejecutar llamadas al sistema de entrada/salida de dos formas: síncrona y asíncrona. El modo síncrono es el modo normal de trabajo y consiste en hacer peticiones de lectura o escritura que hacen que el originador tenga que esperar a que el sistema le responda, es decir, que le de los datos deseados. A veces se requiere que un mismo proceso sea capaz de supervisar el estado de varios dispositivos y tomar ciertas decisiones dependiendo de si existen datos o no. En este caso se requiere una forma de trabajo asíncrona. Para este tipo de situaciones existen las llamadas a las rutinas 'select' y 'poll' que permiten saber el estado de un conjunto de descriptores.

Hay dos tipos de dispositivos: de bloque (discos, cintas...) y de carácter (terminales, impresoras...).

Para la E/S de red se emplean los SOCKETS (berkeley) que una vez abierto se trata como un fichero.

6. El Sistema de Archivos

En UNIX, un fichero es una secuencia de 0 o más bytes. No existe distinción entre ficheros ASCII, binarios, etc.

Se admiten nombres de hasta 255 caracteres. **No** siguen el esquema habitual de *nombre.extensión*

El sistema de archivos de UNIX, desde el punto de vista del usuario, tiene una organización jerárquica o de árbol invertido que parte de una raíz conocida como "/" (diagonal). Es una diagonal al revés que la usada en DOS. Se puede acceder a un fichero empleando un direccionamiento absoluto, desde el directorio raiz; relativo, desde el directorio actual; usando Enlaces (links), ubicaciones ficticias de ficheros que se encuentran en otro sitio.

Internamente se usa un sistema de direccionamiento de archivos de varios niveles, cuya estructura más primitiva se le llama 'information node' (i-node). El sistema de archivos de UNIX ofrece un poderoso conjunto de comandos y llamadas al sistema. En la tabla 3 se muestran los comandos más útiles para el manejo de archivos en UNIX.

UNIX	**Utilidad**
rm	borra archivos
cp	copia archivos
mv	renombra archivos
ls	lista directorio
mkdir	crea un directorio
rmdir	borra directorio
ln	crea una 'liga simbólica'
chmod	maneja los permisos
chown	cambia de dueño Tabla

Tabla 3 Comandos UNIX del Sistema de Archivos

Cada partición de un disco tiene una estructura:

Bloque Boot

Superbloque (n° de bloques, n° de i-nodes, comienzo de la lista de bloques libres)

i-nodes: uno por fichero (el número de ellos está limitado en la creación del sistema de archivos)

Bloques de datos.

Un fichero puede ocupar varios bloques no necesariamente contiguos.

Un disco puede tener varias particiones con su propia estructura.

Dentro de la estructura de directorios de UNIX existen una serie de directorios comunes a todas las instalaciones que es preciso conocer:

/ directorio raiz, inicio del sistema de archivos.

/swap es un directorio no accesible por el usuario, que hace referencia al área de swap del sistema.

/tmp directorio de archivos temporales.

/dev directorio de dispositivos. En él se encuentran todos los dispositivos de E/S, que se tratan como archivos especiales.

/etc directorio para archivos de sistema diversos

/bin directorio para programas binarios (ejecutables)

/lib directorio de bibliotecas del sistema

/usr directorio de usuarios

/home directorio base a partir del cual se ubican los directorios por defecto de las cuentas de usuario.

Además del sistema de archivos clásico de UNIX, existen otros sistemas de archivos que pueden emplearse en UNIX y LINUX. Berkeley mejoró el sistema clásico con el **BFFS** (*Berkeley Fast File System*), que fue quien introdujo los nombres de archivo de 255 caracteres (el clásico solamente admitía 14). Berkeley tuvo que introducir cuatro llamadas al sistema nuevas **opendir, closedir, readdir y rewinddir** para que los programas pudieran leer los directorios sin conocer su estructura interna. Posteriormente estos cambios se añadieron a POSIX y a todas las versiones de UNIX.

LINUX comenzó empleando el sistema de archivos de MINIX, pero estaba limitado a nombres de 14 caracteres y a archivos de 64 MB de tamaño. La primera mejora vino de la mano de un sistema de archivos denominado **Ext** que permitía nombres de archivo de 255 caracteres y 2 GB de tamaño por archivo, pero era muy lento. La evolución vino de la mano del sistema de archivos **Ext2**, con nombres de archivo largos, archivos grandes y mejor rendimiento, siendo en la actualidad el sistema de archivos por defecto en LINUX. **Ext2** es muy similar a BFFS, con algunas diferencias menores.

Con objeto de compartir los archivos entre ordenadores conectados en una red, a menudo con diferentes sistemas de archivos, Sun Microsystems desarrolló el sistema de archivos de red **NFS** (*Network File System*), que se usa en todos los sistemas UNIX modernos, incluido LINUX.

NFS se basa en la idea de permitir que una colección arbitraria de clientes y servidores, compartan un mismo sistema de archivos. NFS puede ejecutarse entre ordenadores situados en redes distintas e incluso en un solo ordenador, que ejecuta las funciones de servidor y cliente al mismo tiempo. NFS implementa sus propios protocolos de comunicación entre los clientes y los servidores. En resumen, los equipos servidores exportan una lista de subdirectorios a compartir, lo que incluye toda la jerarquía que tienen por debajo y los clientes conectan esos recursos a sus propias jerarquías haciendo el acceso completamente homogéneo y transparente al usuario.

7. Seguridad

La seguridad en el entorno UNIX, se basa en la existencia de usuarios y grupos registrados. A cada usuario se le asigna un identificador **uid** (*User Identification*) y un **gid** (*Group Identification*) del grupo de pertenencia, en el archivo /etc/passwd los **uid** de valor inferior a **100** tienen unos privilegios especiales, siendo el usuario con **uid 0** el conocido como **root** o **superusuario** del sistema.

La protección de archivos en UNIX se maneja por medio de una cadena de permisos de nueve caracteres. Los nueve caracteres se dividen en tres grupos de tres caracteres cada uno.

```
rwx rwx rwx
 1   2   3
```

El primer grupo (1) especifica los permisos del dueño del archivo *owner*. El segundo grupo (2) especifica los permisos para aquellos usuarios que pertenecen al mismo grupo de trabajo que el dueño, *group*, y finalmente el tercer grupo (3) indica los permisos para el resto del mundo *others* o *world*. En cada grupo de tres caracteres pueden aparecer las letras RWX en ese orden indicando permiso de leer (**R**ead), escribir (**W**rite) y ejecutar (e**X**ecute). Por ejemplo, la cadena completa rwxr-xr-- indica que el dueño tiene los tres permisos (Read, Write, Execute), los miembros de su grupo de trabajo tienen permisos de leer y ejecutar (Read, Execute) y el resto del mundo sólo tienen permiso de leer (Read).

8. La interfaz Hombre-Máquina

La interfaz por defecto del Sistema UNIX se denomina **shell**, es un intérprete de línea de comandos que se ejecuta cuando el usuario autorizado accede al sistema. El **shell** no es parte del núcleo del sistema operativo, por lo que pueden emplearse diferentes versiones.

Las versiones de shell más conocidas son: **Bourne shell**, estándar del System V; la **C shell**, típica del BSD y la **Korn shell**, versión ampliada de Bourne.

Una vez invocada, la shell presenta unas características que la hacen muy parecida a un idioma de programación interpretado:

- Variables
- Estructuras de control, **if**, **while**, ...
- Subprogramas
- Paso de parámetros
- Manejo de interrupciones

Estas características le dotan de capacidad para diseñar herramientas propias. Los archivos de comandos se denominan **shell scripts**.

El formato de las órdenes a la shell sigue el esquema:

comando *opciones* argumentos

Permite el empleo de caracteres comodín: ***** ; **?**

Todo comando o proceso lleva asociado 3 ficheros estándar: stdin, stdout, stderr.

Es posible redireccionar el origen o el destino de estos ficheros, secuenciar tareas (;), encolar tareas de modo que la salida de una sea la entrada de la siguiente (|), ejecutar multiproceso (&).

LINUX

En el entorno LINUX también existe el shell de línea de comandos, sin embargo el éxito mayor ha sido la incorporación de la interfaz gráfica de usuario o **GUI** (*Graphical User Interface*) basada en ventanas, similar a los sistemas Macintosh y Windows.

La característica más importante del entorno gráfico es su completa independencia del núcleo del sistema operativo, lo que permite seleccionar diferentes interfaces para trabajar. Por otro lado, cualquier problema derivado de la interfaz no afecta a la estabilidad del sistema, siendo suficiente con terminar el programa y relanzar nuevamente la interfaz para recuperar la funcionalidad.

A mediados de 1980 se creó una fundación para entornos de usuario gráficos independientes del sistema operativo que se llamó X-Windows. Las especificaciones X-Windows definen el método por el cual se pueden comunicar las aplicaciones con el hardware gráfico. También establecen un conjunto de funciones de programación bien definidas que podrán ser llamadas para realizar la manipulación básica de las ventanas.

La definición básica de cómo se dibuja una ventana y de cómo se manejan los clics del ratón no incluye la definición de cómo se deberían ver las ventanas. En realidad X-Windows en su estado natural no tiene apariencia real. El control de la apariencia se pasó a un programa externo denominado **gestor de ventana**. El gestor de ventana se preocupa de dibujar los bordes, usar el color y hacer que el entorno sea agradable a la vista. El gestor de ventana solo se requiere para usar las llamadas estándares al subsistema X-Windows para dibujar sobre la pantalla.

Sin embargo la dificultad de programar en este entorno propició la aparición, a finales de los 90, de dos grupos independientes que aportaron soluciones a los problemas de X-Windows: **GNOME** y **KDE.** KDE ofrece un gestor de ventanas nuevo y las bibliotecas necesarias para escribir aplicaciones de un modo mucho más fácil. GNOME ofrece un marco general para el resto de gestores de ventana y para las aplicaciones que trabajan con ellos. Cada uno tiene una idea diferente de cómo deben funcionar las cosas, pero debido a que trabajan por encima de X-Windows, no son enteramente incompatibles.

KDE (*K Desktop Environment*) o entorno de escritorio K, es un entorno de escritorio ligeramente distinto de los gestores típicos de ventanas. En lugar de describir cómo se ve la interfaz, KDE proporciona un conjunto de bibliotecas que, si se usan, permiten a una aplicación mejorar algunas características especiales que ofrece el gestor de ventanas (soporte de pinchar y arrastra, soporte de impresión estandarizado, etc.). KDE ofrece al programador una biblioteca mucho más sencilla de usar que el trabajo directo con la interfaz X-Windows. Una de las herramientas más potentes es el escritorio virtual, lo que permite tener varias pantallas efectivas al mismo tiempo.

GNOME (*GNU Network Object Model Environment*) al igual que KDE ofrece un entorno de escritorio completo y un marco de aplicaciones para desarrollo tan bueno como de fácil uso. A diferencia de KDE, GNOME **no es un gestor de ventanas**, proporciona bibliotecas de desarrollo y características de gestión de sesión que nosotros, como usuarios, no vemos. En lo más alto de su estructura está un gestor de ventanas que muestra la apariencia general del escritorio. El gestor de ventana por defecto es **Enlightenment**, pero hay disponibles varias opciones más. Desde el punto de vista del desarrollador, GNOME define sus interfaces con el resto del mundo mediante tecnología CORBA. Cualquier sistema de desarrollo que pueda comunicarse usando CORBA puede utilizarse para desarrollar aplicaciones compiladas en GNOME.

En KDE, el escritorio está compuesto por seis partes principales: el **Window Manager (kwin);** el **Desktop Manager (kdesktop)**; el **Panel (kicker)**; el **Control Center (kcontrol)**; el **Help Center (khelpcenter)** y el **Browser/File Manager/Universal Viewer (konqueror)**.

KDE se compone de subsistemas que realizan la mayor parte del trabajo. Los subsistemas más importantes son:

DCOP *Desktop Communication Protocol.*

KIO *Network Transparent I/O.*

SYCOCA *SYstem COnfiguration CAche.*

KParts Componentes integrados

KHTML librería HTML 4.0

XMLGUI Arquitectura dinámica GUI, basad en XML**.**

aRts Sistema Multimedia**.**

GNOME permite el empleo de cualquier gestor de ventana que se ajuste a sus especificaciones, en la actualidad algunos gestores más conocidos son: **Sawfish**, **Enlightenment**, **Window Maker**, **IceWM**, **Scwm** y **FVWM** (versión 2.3). El Administrador de ficheros de GNOME se denomina **Nautilus**. De toda la lista de paquetes disponibles para GNOME, algunos son parte integrante del propio entorno y otros solo se necesitan si se va a desarrollar haciendo uso de sus funcionalidades. Algunos de los paquetes de GNOME son:

ORBit es el proveedor de CORBA, es el que permite la comunicación entre las diferentes partes de GNOME.

GTK es la **Gimp ToolKit** (GIMP viene de GNU Image Manipulation Program que es una aplicación de manipulación de gráficos de código abierto). Permite la creación de botones, barras de desplazamiento y otros elementos básicos.

Imlib es una librería de dibujo (render), es la responsable de indicar a X-Windows como dibujar las cosas y como cargar imágenes.

gnome-libs contiene las librerías básicas para todas las aplicaciones GNOME.

glib contiene rutinas que emplean casi todos los programas de GNOME.

libxml permite interpretar los archivos en XML, a veces se conoce como **gnome-xml**.

libglade librería que lee ficheros XML, mediante libxml y lo convierte en interfaz de usuario.

libgtop sirve para dar a GNOME información sobre la organización de los sistemas de archivo y como el sistema controla lo que se ejecuta y cuando.

libghttp la librería HTTP de LINUX.

esound server es el encargado de manejar el sonido bajo LINUX.

gnome-print ofrece un entorno homogeneo de impresión a las aplicaciones.

bug-buddy programa para ayudar en la comunicación de errores (bugs).

gdk-pixbuf es una librería para la carga de imágenes, dibujo y carga de animaciones simples.

eog es un visor de imágenes pequeño y rápido.

bonobo permite incrustar contenidos de un programa dentro de otro (embeber).

oaf (*Object Activation Framework*), se emplea en las nuevas aplicaciones GNOME para buscar y ejecutar objetos CORBA.

gconf es la aplicación encargada de mantener las opciones de configuración

*** FIN DEL TEMA ***