Introducción a Unix

Unidad 1

Curso basado en uno propuesto por William Knottenbelt, UK, 2001

Daniel Millán, Nora Moyano y Evelin Giaroli

Facultad de ciencias aplicadas a la industria

2017

Introducción

1645: Blaise Pascal inventa la pascalina. Los datos se representaban mediante las posiciones de los engranajes. La pascalina es una de las primeras calculadoras mecánicas, que funcionaba a base de ruedas de diez dientes en las que cada uno de los dientes representaba un dígito del 0 al 9.
Wikipedia.

Las ruedas estaban conectadas de tal manera que podían sumarse números haciéndolas avanzar el número de dientes correcto.

1949: El EDVAC, una de las primeras computadoras de programas almacenados electrónicamente (binaria). *Wikipedia*.

Sistema Operativo Unix

1. ¿Qué es un sistema operativo?
2. Breve introducción a la historia de Unix.
3. Arquitectura del sistema operativo Linux.
4. Inicio/Salida de sesión en sistemas Unix.
5. Cambio de contraseña.
6. Formato general de los comandos de Unix.
7. El sistema de ficheros Unix.
8. Típica estructura de directorios Unix.
9. Manejo de archivos y directorios.
10. Enlaces a ficheros (directos/simbólicos).
11. Especificación de múltiples nombres de archivo.
12. Comillas y caracteres especiales.

¿Qué es un sistema operativo?

Un sistema operativo (OS) es un gestor (administrador) de recursos

Se presenta en forma de un conjunto de rutinas de software que permiten a los usuarios y a los programas acceder a los recursos del sistema de una manera segura, eficiente y abstracta.

CPU, tarjetas de red, discos de memoria, módems, impresoras, etc..

CPU: central processing unit.

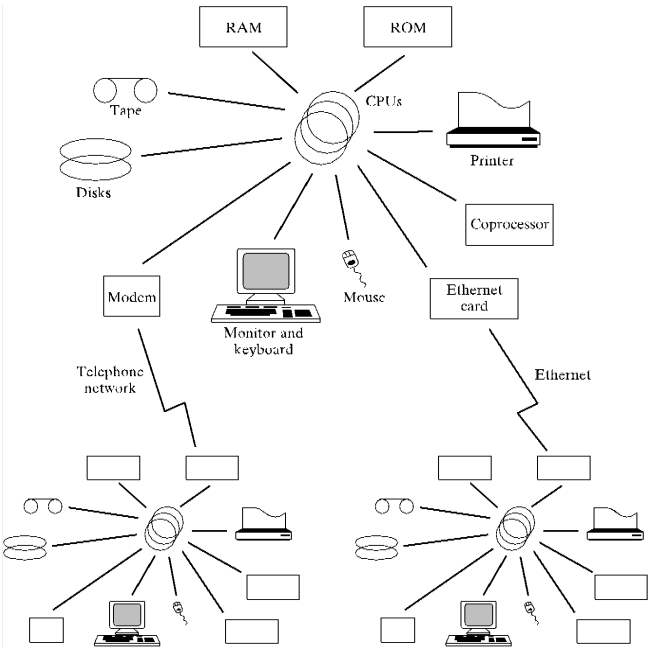
Unidad de Procesamiento Central.

El SO asegura un acceso seguro p.ej. impresora.

El USO fomenta el uso eficiente de la CPU mediante suspensión de operaciones de E/S.

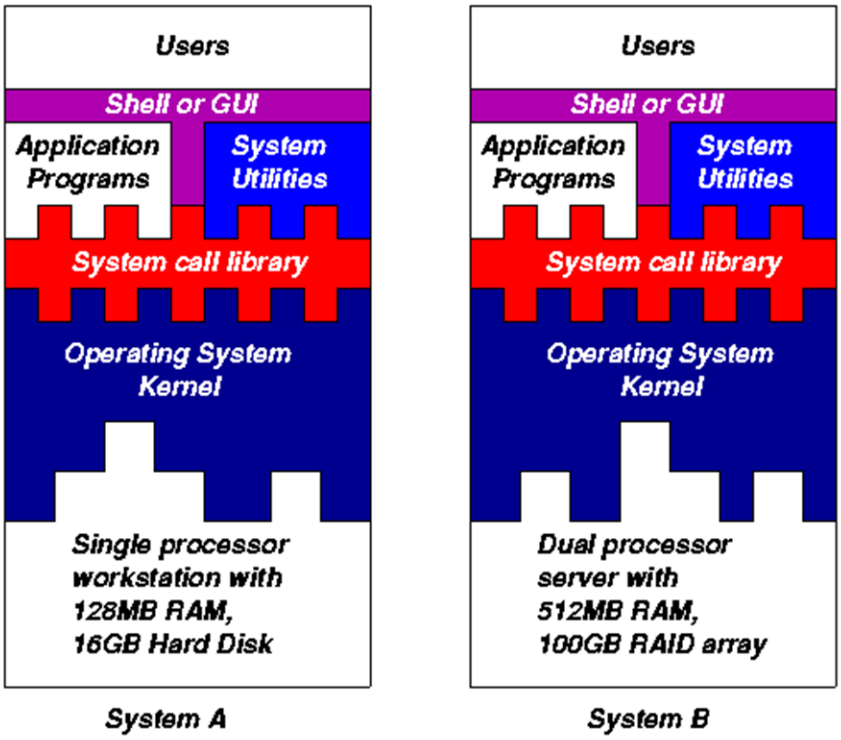
El SO proporciona abstracciones tales como archivos en lugar de posiciones de memoria en discos (detalles de hardaware están ocultos).

Por ejemplo, un sistema operativo asegura un acceso seguro a una impresora al permitir programa de una sola aplicación para enviar los datos directamente a la impresora a la vez. Un sistema operativo fomenta el uso eficiente de la CPU mediante la suspensión de los programas que están en espera de operaciones de E/S para completar para dar paso a los programas que pueden utilizar la CPU de forma más productiva. Un sistema operativo también proporciona abstracciones convenientes (tales como archivos en lugar de las ubicaciones de discos) que aíslan a los programadores y usuarios de los detalles del hardware subyacente.



La Fig. presenta la arquitectura de un SO típico, se muestra cómo el SO presenta una interfaz uniforme de cara a usuarios y programas de aplicación sin tener en cuenta los detalles del

hardware subyacente.



La Fig. presenta la arquitectura de un SO típico, se muestra cómo el SO presenta una interfaz uniforme de cara a usuarios y programas de aplicación sin tener en cuenta los detalles del hardware subyacente.

- El **núcleo del** SO controla de forma directa el hardware subyacente
- El núcleo maneja dispositivos de bajo nivel, la memoria y la gestión del procesador
- Servicios básicos del núcleo están disponibles para programas de nivel superior a través de una biblioteca de **llamadas al sistema**
- **Los programas de aplicación** (procesadores de texto, hojas de cálculo) y **programas de utilidades del sistema** (buscador) hacen uso de las llamadas al sistema

Aplicaciones y utilidades del sistema se ponen en marcha mediante un **shell** (una interfaz de línea de comandos de texto) o una **interfaz gráfica de usuario** que proporciona una interacción directa (mouse

Breve historia de Unix

- UNIX ha sido un SO popular durante más de 4 décadas debido a que brinda un entorno
  - **Multiusuario**
  - **Multitarea**
  - **Estabilidad**

  - **Portabilidad**
  - **Capacidades de red de gran alcance**
- **1960**: General Electric + MIT + Bell Labs (AT&T) desarrollan MULTICS
  - SO multi-usuario y multitarea en ordenadores centrales (cajas grandes)
  - MULTICS: ***MULT****iplexed* ***I****nformation and* ***C****omputing* ***S****ystem*
- **1969**: Ken Tompson (Bell Labs)
  - Crea un SO basado en MULTICS pero más sencillo en una PDP7 (mini PC 1965)
  - UNICS: ***UN****iplexed* ***I****nformation and* ***C****omputing* ***S****ystem* ➔UNIX
  - Poca memoria y potencia llevan a utilizar comandos cortos: **ls, cp, mv…**
  - El lenguaje de programación en que fue escrito UNICS se llamaba B
- **1971**: Se une Dennis Ritchie
  - Crea el primer compilador de C y se reescribe el núcleo de UNIX en C (1973)
  - Mejora de la portabilidad
  - Se lanza la quinta versión de UNIX a las Universidades en 1974 (GRATIS)
- **1978**: Se separan dos grandes ramas: SYSV (AT&T y otras empresas) y BSD (Berkley Sofware Distribution de la UCB) ➔Incompatibles!

MULTICS: Sistema de Información Informática Multiplexado
Multiplexación es un método por el cual múltiples señales de mensaje analógica o flujos de datos digitales se combinan en una señal a través de un medio compartido. El objetivo es compartir un recurso caro.

MULTICS fracasaron (para algunos entusiastas MULTICS "fracaso" es quizás una palabra demasiado fuerte para usar aquí), pero inspiró a Ken Thompson, que era un investigador en los Laboratorios Bell, en escribir un SO más sencillo. Él escribió una versión más simple de MULTICS en un PDP7 en ensamblador y llamó a su intento de UNICS (Sistema de Información Informática y Uniplexed). Debido a que la memoria y potencia de CPU eran un bien escaso en esos días, UNICS (con el tiempo acortado a UNIX) utiliza comandos cortos para minimizar el espacio necesario para almacenarlos y el tiempo necesario para decodificar - de ahí la tradición de corta comandos UNIX usamos hoy en día, por ejemplo, ls, cp, rm, mv, etc.

Ken Thompson y luego se unió a Dennis Ritchie, el autor del primer compilador de C en 1973. Reescribieron el núcleo de UNIX en C - este fue un gran paso adelante en términos de portabilidad del sistema - y lanzó la quinta edición de UNIX a las universidades de 1974.

La séptima edición, publicada en 1978, marcó una ruptura en el desarrollo de UNIX en dos ramas principales: SYSV (Sistema 5) y BSD (Berkeley Software Distribution). BSD surgió de la Universidad de California en Berkeley, donde Ken Thompson pasó un año sabático. Su desarrollo fue continuado por los estudiantes de Berkeley y otras instituciones de investigación. SYSV fue desarrollado por AT&T y otras empresas comerciales. Versiones de UNIX basados en SYSV han sido tradicionalmente más conservador, pero mejor con el apoyo de sabores basados en BSD.

Las últimas encarnaciones de SYSV (o SVR4 Sistema 5 Release 4) y BSD Unix son en realidad

muy similares.

AT&T: American Telephone and Telegraph Company

BSD: memoria virtual, gestión de redes, utilidades como vi, csh, etc.

Ken Thompson y Dennis Ritchie. *Wikipedia.*

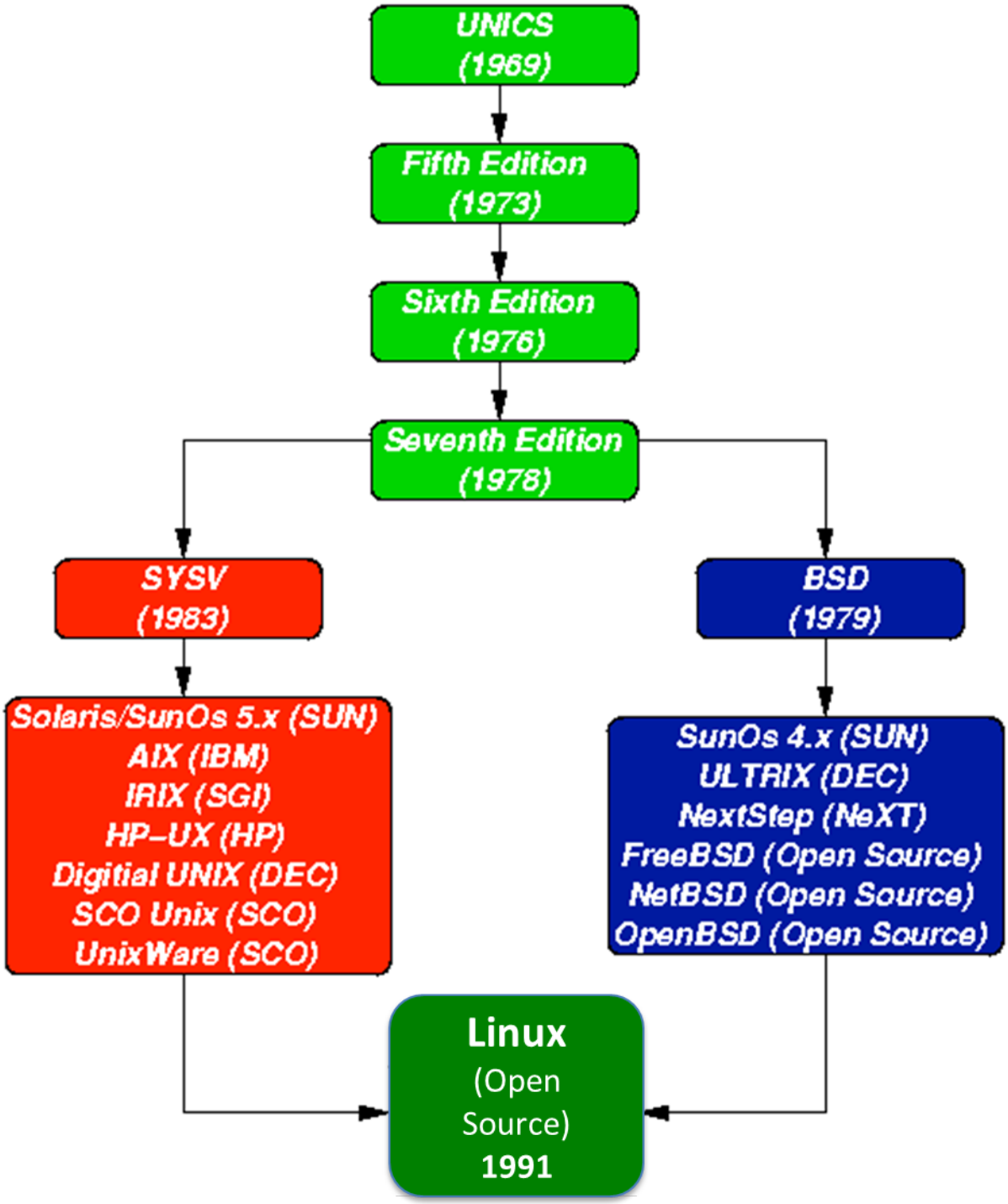
DEC PDP7. *Wikipedia.*

Feature	Typical SYSV	Typical BSD
kernel name	/unix	/vmunix
boot init	/etc/rc.d directories	/etc/rc.* files
mounted FS	/etc/mnttab	/etc/mtab
default shell	sh, ksh	csh, tcsh
FS block size	512 bytes->2K	4K->8K
print subsystem	lp, lpstat, cancel	lpr, lpq, lprm
echo command (no new line)	echo "\c"	echo -n
ps command	ps -fae	ps -aux
multiple wait syscalls	poll	select
memory access syscalls	memset, memcpy	bzero, bcopy

Algunas diferencias entre SYSV y BSD se encuentran en la

- estructura del sistema de archivos
- los nombres y las opciones de utilidades del sistema
- bibliotecas de llamada del sistema

Algunas diferencias menores se encuentran en la estructura del sistema de archivos, los nombres y las opciones de utilidades del sistema y las bibliotecas de llamada del sistema



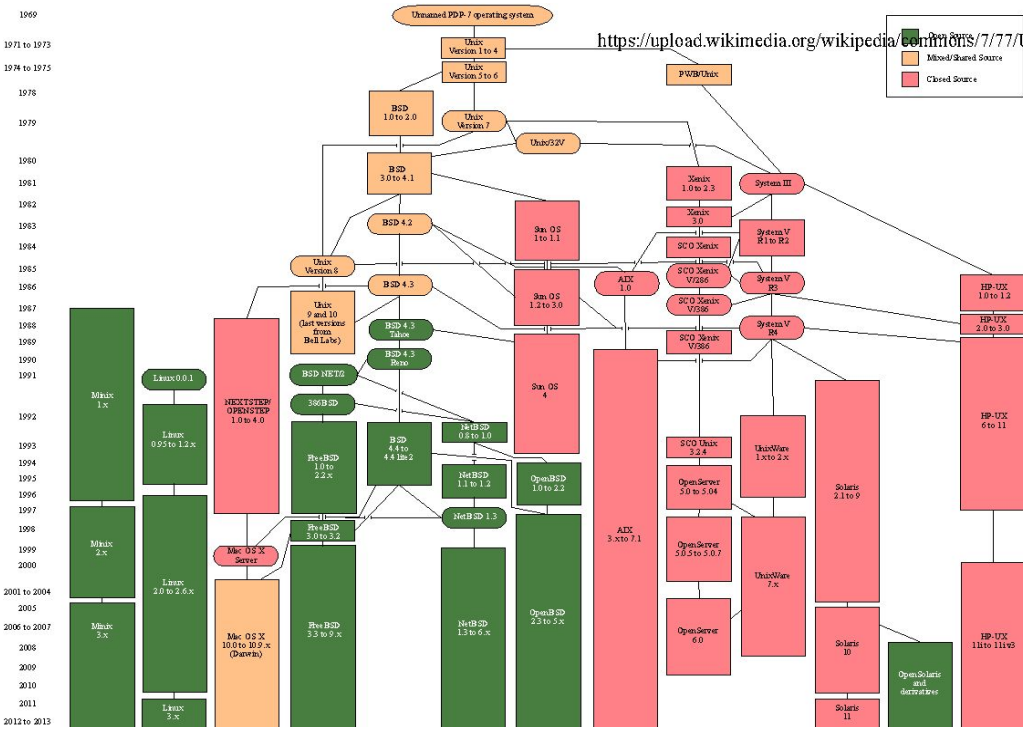
Significa que el código fuente del núcleo Linux está disponible gratuitamente para que cualquiera pueda añadir características y corregir deficiencias.

Lo que comenzó como un proyecto de una persona se ha convertido ahora en una colaboración de

miles de desarrolladores voluntarios de todo el mundo.

El enfoque de código abierto con éxito no sólo ha sido aplicada al núcleo del código, sino también a los programas de aplicación para Linux.

- **1991**: Linus Torvalds, un estudiante finlandés de Ciencias de la Computación diseña Linux un código abierto del SO UNIX para PC
  - No es SYSV ni BSD, pero incorpora características de cada uno (p.ej. al estilo SYSV archivos de inicio, pero con una disposición del sistema de archivos del tipo BSD)
  - Cumple con un conjunto de estándares de IEEE (*Institute of Electrical and Electronics Engineers*) llamado POSIX (*Portable Operating System Interface*)
  - Para maximizar la portabilidad del código, Linux *típicamente* soporta SYSV, BSD y llamadas al sistema de POSIX
  - Linux ha generado que miles de personas colaboren voluntariamente durante >25 años mejorando el núcleo y programas de aplicación
  - Diferentes distribuciones: Debian, Suse, RedHat, Ubuntu, etc
  - Portable a diferentes arquitecturas de procesadores como Intel, AMD, SPARC...
  - *Fácil* de usar e instalar y viene con un conjunto completo de utilidades y aplicaciones, incluyendo el sistema de gráficos X, entornos GNOME y KDE GUI, y la suite StarOffice (un clon de código abierto MS-Office para Linux)



Principios de diseño

- UNIX fue diseñado para ser un SO interactivo, multiusuario y multitarea:
  - **Interactivo** quiere decir que el sistema acepta órdenes, las ejecuta y se dispone a

esperar otras nuevas.

  - **Multitarea** significa que puede realizar varios trabajos, denominados procesos, al mismo tiempo.
  - **Multiusuario** significa que más de una persona puede usar el sistema al mismo tiempo.

- UNIX fue diseñado por programadores para ser usado por programadores en un entorno en que los usuarios son relativamente expertos y participan en el desarrollo de proyectos de software ➔ *No user friendly*

Arquitectura del SO Linux

Linux tiene todos los componentes de un SO tipo UNIX:

- Núcleo: facilita accesso seguro a distintos programas al hardware (tarjetas gráficas y red, discos duros, etc), decide que programas utilizan hardware y cuánto tiempo (multiplexado), BSD/SYSV llamadas sistema, etc
- Shells y GUIs:
  - Intérpretes de línea de comandos (shells) como en UNIX: **sh**: shell Bourne, **bash:** *Bourne again shell* y **csh**: *C shell*
  - Interface Gráfica (GUI, *Graphic User Interface*), gestores KDE y GNOME
- Utilidades del sistema: Herramientas poderosas que hacen una sola tarea extremadamente bien.
  - **cp** copia, **grep**: busca expresiones regulares (caracteres), **awk**: procesa datos definidos en archivos de texto, **sed**: editor de flujo de texto, demonios, etc
- Programas de aplicación:
  - **emacs**: editor de texto, **gcc/g++**: compilador de C/C++, **latex**: lenguaje de composición de texto, **xfig**: paquete de dibujo vectorial, **StarOffice**, etc

Linux tiene todos los componentes de un sistema operativo típico (en este punto que le gustaría hacer referencia a la figura 1.1):

Núcleo El kernel Linux incluye soporte para controlador de dispositivo para un gran número de dispositivos de hardware de PC (tarjetas gráficas, tarjetas de red, discos duros, etc.), las características del procesador y la memoria de gestión avanzada, y soporte para muchos tipos diferentes de sistemas de archivos (incluyendo disquetes DOS y la ISO9660 estándar para CDROMs). En cuanto a los servicios que presta a los programas de aplicación y utilidades del sistema, el núcleo implementa la mayoría de BSD y SYSV sistema de llamadas, así como las llamadas del sistema descritos en la memoria POSIX.1. El kernel (en forma binaria en bruto que se carga directamente en la memoria en tiempo de inicio del sistema) se encuentra típicamente en el archivo / boot / vmlinuz, mientras que los archivos de origen por lo general se pueden encontrar en /usr/src/linux.The última versión del fuentes del kernel de Linux pueden descargarse desde http://www.kernel.org .

Shells y GUI:Linux soporta dos formas de entrada de comando: a través de intérpretes de línea de comandos de texto similares a los encontrados en la mayoría de los sistemas UNIX (por

ejemplo, sh - el shell Bourne, Bash - la Bourne again shell y CSH - el shell C) ya través de las interfaces gráficas (GUI) tales como los gestores de ventanas KDE y GNOME. Si va a conectarse de forma remota a un servidor de su acceso será típicamente a través de un shell de línea de comandos.

Utilidades del sistema Prácticamente cada utilidad de sistema que se puede esperar encontrar en las implementaciones estándar de UNIX (incluyendo todas las utilidades del sistema descrito en la memoria POSIX.2) ha sido portado a Linux. Esto incluye comandos como ls, cp, grep, awk, sed, BC, wc, más, y así sucesivamente. Estas utilidades del sistema están diseñados para ser herramientas poderosas que hacen una sola tarea extremadamente bien (por ejemplo, grep encuentra dentro de los archivos de texto, mientras que los recuentos de wc el número de palabras, líneas y bytes dentro de un archivo). Los usuarios a menudo pueden resolver los problemas mediante la interconexión de estas herramientas en lugar de escribir un gran programa de aplicación monolítica. Al igual que otras versiones de UNIX, las utilidades del sistema de Linux también incluyen programas de servidor llamados demonios que proporcionan servicios de red y administración remota (por ejemplo, telnetd y sshd proporcionar facilidades de acceso remoto, LPD proporciona servicios de impresión, httpd sirve páginas web, crond ejecuta tareas de administración de sistemas regulares de forma automática). Un demonio (probablemente derivado de la palabra latina que hace referencia a un espíritu benéfico que cuida de alguien, o tal vez la abreviatura de "disco y seguimiento de la ejecución") por lo general se generó automáticamente al iniciar el sistema y pasa la mayor parte de su tiempo en estado latente (que está al acecho?) a la espera de que se produzca algún acontecimiento.

la palabra daemon fue utilizada en 1963 por primera vez, en el área de la informática, para denominar a un proceso que realizaba backups en unas cintas. Este proceso se utilizó en el proyecto MAC del MIT y en una computadora IBM 7094. Dicho proyecto estaba liderado por Fernando J. Corbató, quien afirma que se basó en el demonio de James Maxwell, este daemon era una especie de vigilante que residía en medio de un recipiente dividido en dos, lleno de moléculas. El vigilante o daemon se encargaba de permitir, dependiendo de la velocidad de la molécula, que éstas pasaran de un lado al otro. Los daemons de las computadoras actúan muy similar al daemon de Maxwell, pues realizan acciones según el comportamiento y algunas condiciones del sistema.

Programas de aplicación Distribuciones de Linux normalmente vienen con varios programas de aplicaciones útiles como estándar. Los ejemplos incluyen el editor emacs, xv (un visor de imágenes), gcc (el compilador de C), g ++ (un compilador de C ++), xfig (un paquete de dibujo), látex (un poderoso lenguaje de composición) y soffice (StarOffice, que es un MS clon de estilo -Oficina que puede leer y escribir archivos de Word, Excel y PowerPoint).

Inicio/Salida de sesión SO Unix

- **(TTY) terminales de texto:** Cuando se conecta a un ordenador UNIX de forma remota o al iniciar una sesión localmente usando una terminal de sólo texto, verá el símbolo:

  login: *pepe* (inicio de sesión)

- En este indicador, escriba su nombre de usuario y presione **Enter**. Recuerde que UNIX es sensible a mayúsculas (*pepe*, *Pepe* y *PEPE* son inicios de sesión diferentes).

  login: *pepe*

password: ****** (contraseña)

- Escriba su contraseña en el indicador y presione **Enter**. Tenga en cuenta que la contraseña no se mostrará en la pantalla al escribir.
- Si escribe mal su nombre de usuario o la contraseña recibirá un mensaje y se mostrará nuevamente **login**:
- De lo contrario, el intérprete de comandos muestra algo como esto:

  $ (el cursor parpadea ➔ esperando instrucción)
- Para salir de la sesión basada en texto, escriba **exit** en el intérprete de comandos (o si eso no funciona intente **logout**, y si eso no funciona pulse Ctrl-D).
- **Terminal Gráfica**
- Si usted está entrando en un ordenador UNIX a nivel local, o si está utilizando un inicio de sesión remoto que soporta gráficos, observará los campos de usuario y contraseña.
- Introduzca su nombre de usuario y la contraseña de la misma manera que antes (Nota: tecla TAB permite moverse entre los campos).
- Una vez que se ha identificado, se le presentará un gestor de ventanas que se ve similar a la interfaz de Microsoft Windows. Para que aparezca una ventana de intérprete de comandos busque los menús o íconos que mencionan las palabras **shell**, **xterm**, **terminal emulator**, or **console**.
- Para cerrar la sesión de un gestor de ventanas gráficas, buscar opciones de menú similares a **Exit** o **Log Out**.

Cambio de contraseña

- Una de las cosas que debe hacer cuando se conecta por primera vez en un SO tipo UNIX es cambiar su contraseña. El comando de UNIX para cambiar su contraseña es passwd:

  $ passwd
- El sistema le pedirá la contraseña anterior, a continuación, introduzca su nueva contraseña (por duplicado)
- Recuerde los siguientes puntos al elegir su contraseña:
  - Evitar caracteres que pudieran no aparecer en todos los teclados, por ejemplo, '£'.
  - El eslabón más débil en la seguridad informática suelen ser las contraseñas de los usuarios. No la facilite a nadie. Evite las palabras del diccionario o palabras relacionadas con sus datos personales (p.ej., nombre novia/novio o el nombre de su nombre de usuario).
  - La contraseña de poseer al menos 7 u 8 caracteres de longitud y tratar de utilizar una combinación de letras, números y puntuación.

Formato de comandos de Unix

- Una línea de comandos UNIX consiste en el nombre de un comando UNIX (en realidad el "comando" es el nombre de un programa de la **shell**, una utilidad del sistema o un programa de aplicación) seguido de sus "argumentos" (opciones y los nombres de archivo

de destino)

- La sintaxis general para un comando UNIX es:

  $ **comando –opciones objeto**

- Aquí **comando** puede ser entendido como un verbo, **opciones** como un adverbio y **objeto**, como los objetos directos del verbo.
- En el caso de que querer especificar varias opciones, éstas no siempre tienen que ser listadas por separado (las opciones comúnmente se pueden concatenar después de un único guión).

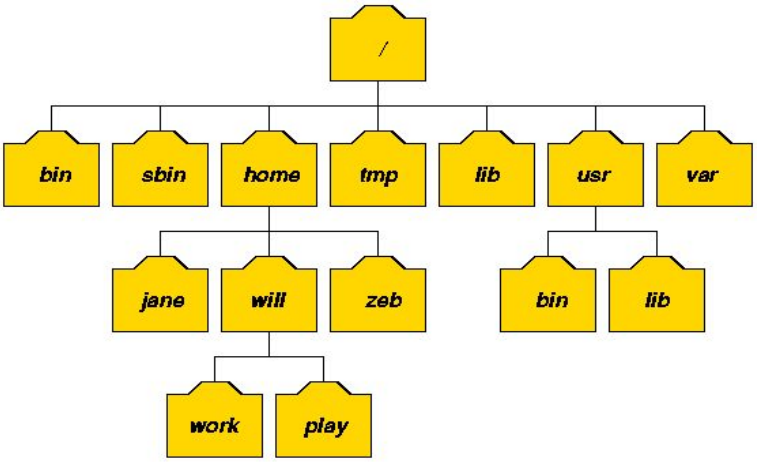
El sistema de ficheros Unix

- El sistema operativo UNIX se basa en el concepto de un sistema de archivos que se utiliza para almacenar toda la información que constituye el estado a largo plazo del sistema
- Cada elemento almacenado en un sistema de archivos de UNIX pertenece a uno de cuatro tipos:
  - **Archivos ordinarios**: contienen datos, texto, o información de programas. Para definirlos no utilizar ***,?, #** ni espacios (utilizar el guión bajo)
  - **Directorios**: carpetas que contienen archivos y otros directorios
  - **Dispositivos:** Para proporcionar que las aplicaciones tengan fácil acceso a los dispositivos de hardware, UNIX permite que sean manejados mediante archivos
  - **Enlaces**: es un puntero a otro archivo. Hay dos tipos de enlaces. **Enlace físico** es indistinguible de la del propio archivo. Un **enlace simbólico** es un puntero indirecto a un archivo, consiste en una archivo de directorio que contiene la dirección del archivo en el sistema de ficheros

Este estado incluye el núcleo del SO en sí mismo, los archivos ejecutables para los comandos soportados por el SO, información de configuración, archivos de trabajo temporales, datos de usuario, y varios ficheros especiales que se utilizan para dar acceso controlado a hardware del sistema y las funciones del sistema operativo.

Estructura de directorios Unix

- El sistema de archivos de UNIX se presenta como una estructura jerárquica de árbol que está anclado en un directorio especial de alto nivel conocido como la raíz: **/**
- Debido a la estructura de árbol, un directorio puede tener muchos directorios secundarios, pero sólo un directorio padre.



- **/** El directorio "raíz"
- **/bin** Utilidades del sistema de bajo nivel esenciales
- **/usr/bin** Utilidades del sistema de nivel superior y los programas de aplicación
- **/sbin** Utilidades del sistema superusuario (para realizar tareas de administración del sistema)
- **/lib** Bibliotecas de programas (llamadas al sistema que se pueden incluir en los programas por un compilador) para las utilidades del sistema de bajo nivel
- **/usr/lib** Bibliotecas de programas para programas de usuario de alto nivel
- **/tmp** Espacio de almacenamiento de archivos temporales (se puede utilizar por cualquier usuario)
- **/home** Directorios de los usuarios que contienen espacio de archivos personales de cada usuario. Cada directorio es el nombre de sesión del usuario.
- **/etc** Archivos UNIX de configuración y la información del sistema
- **/dev** Dispositivos de hardware
- **/proc** Un pseudo sistema de archivos que se utiliza como una interfaz para el kernel. Incluye un subdirectorio para cada programa activo (o proceso).

Manejo de archivos y directorios

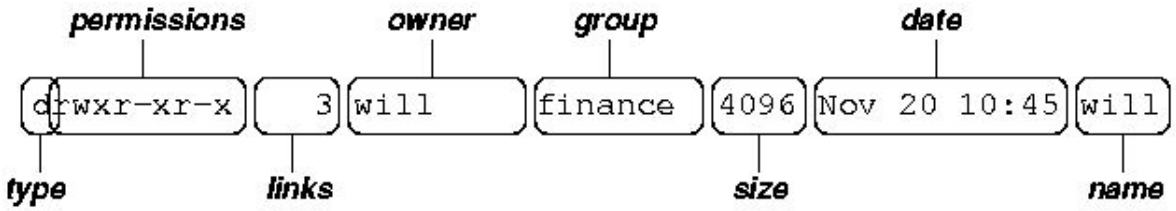
- Algunos de los comandos más importantes

pwd ruta absoluta a la ubicación actual en el sistema de archivos

ls muestra el contenido de un directorio (p/defecto el directorio de trabajo)

ls –a muestra todo incluso archivos ocultos (comienzan con .)

ls –a –l o equivalentemente **ls –al**



Tipo: **"d"** directorio, **"-"** archivo ordinario, **"l"** enlace simbólico

Permisos: tres derechos de acceso, lectura **"r"**, escritura **"w"** y ejecución **"x"**, y tres categorías de usuarios: propietario, grupo y otros (público en general).

Para conocer más opciones de los comandos puedes utilizar **man** o **info.**

man ls manual de usuario en línea del comando **"ls"** de UNIX (muy conciso = duro).

Nota: Si está instalado puede resultar más útil el comando **info** (no estándar).

- Algunos de los comandos más importantes**:**

cd cambia el directorio de trabajo actual a un *path* que puede ser absoluto/relativo

$cd /home/pepe equivalente **$cd ~** equivalente **$cd**

mkdir crea un subdirectorio en el directorio de trabajo actual

rmdir elimina un subdirectorio del directorio actual siempre que esté vacío

cp se utiliza para hacer copias de archivos o directorios completos

mv se utiliza para cambiar el nombre de los archivos / directorios y / o moverlos de un directorio a otro

rm elimina los archivos especificados (directorios), no es posible recuperarlos

cat concatena el contenido de varios archivos y lo muestra por pantalla. Puede ser combinado con **>** redireccionando la salida a un nuevo archivo

more muestra el contenido del *fichero destino*, posee una función búsqueda **/**

less similar a more pero con características adicionales como desplazar hacia atrás

NOTA: Siempre se debe contar con los permisos necesarios!

Enlaces a ficheros (direc/simbol)

- Los enlaces directos (duros) e indirectos (suave o simbólico) de un archivo o directorio a otro pueden ser creados usando el comando **ln**
- $ ln *filename linkname*
  - crea otra entrada de *filename,* digamos *linkname* (enlace duro). Ambas entradas son idénticas (ambos tienen ahora un número de enlace de 2)
  - Si alguno se modifica, el cambio se refleja en el otro archivo
- $ln –s *filename linkname*
  - crea un acceso directo llamado *filename* (*linkname* es un enlace blando). El acceso

directo aparece como una entrada con un tipo especial (‘l’ ele)

- Se puede crear un enlace simbólico a un archivo que no existe, pero no un enlace duro
- Se pueden crear enlaces simbólicos a través de diferentes dispositivos de disco físico o particiones, pero los enlaces duros están restringidos a la misma partición de disco
- UNIX no suele permitir que los enlaces duros apuntan a directorios.

Múltiples nombres de archivo

- Múltiples nombres de archivo pueden ser especificados usando caracteres especiales de **búsqueda de patrones**. Las reglas son:
  - '?' coincide con cualquier carácter único en esa posición del nombre de archivo
  - '*' Coincide con cero o más caracteres del nombre de archivo
  - Caracteres encerrados entre corchetes “[]” coincidirá con cualquier nombre de archivo que tiene uno de esos caracteres en esa posición
  - Una lista de cadenas separadas por comas y encerrada entre llaves “{ }” se expandirá como un producto cartesiano entre caracteres circundantes
- Por ejemplo
  - **???** A todos los ficheros de tres caracteres en el directorio actual
  - **?ell?** nombres de archivo con cinco caracteres con “ell” en el medio
  - **el*** cualquier nombre de archivo que comienzan con “el”
  - **[m-z]*[a-l]** cualquier nombre de archivo que comienze con una letra desde la “m” a la “z” y termine de la “a” a la “l”
  - **{/usr,} {/bin,/lib}/archivo** se expande a: /usr/bin/archivo, /user/lib/archivo, /bin/archivo y /lib/archivo
- Tenga en cuenta que el shell de UNIX realiza estas expansiones (incluyendo cualquier coincidencia de nombre de archivo) de los argumentos de un comando *antes* de ejecutar el comando

Comillas y caracteres especiales

- Como hemos visto ciertos caracteres especiales (por ejemplo, '*', '-', '{', etc.) son interpretados de una manera especial por el shell.
- Para pasar argumentos que utilizan estos caracteres a los comandos de forma directa, tenemos que utilizar *comillas*
- Hay tres niveles que se pueden utilizar:
  - Trate de insertar un (****) frente al carácter especial
  - Use comillas dobles (**"**) alrededor de argumentos para prevenir la mayoría de las expansiones
  - Use comillas simples simples (') alrededor de argumentos para prevenir todas las expansiones

- Hay un cuarto tipo de *comillas* en UNIX. Las invertidas simples (`), se utilizan para pasar la salida de algún comando como un argumento de entrada a otro ➔
- Por ejemplo:

  $ **hostname**

  pepe

  $ **echo** esta máquina se llama `hostname`

  esta máquina se llama pepe