Introducción a Unix: Unidad 1

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

Daniel Millán, Nora Moyano & Evelin Giaroli

Facultad de Ciencias Aplicadas a la Industria, UNCuyo.

Junio de 2017

Introducción [Wikipedia]

La computadora u ordenador no es un invento de alguien en particular, sino el resultado evolutivo de ideas y realizaciones de muchas personas relacionadas con áreas tales como la electrónica, la mecánica, los materiales semiconductores, la lógica, el álgebra y la programación.

Se debe remontar a la edad del bronce (<4000 a.c.), inicio de las grandes culturas y ciudades, donde se dió origen a la escritura cuneiforme a la vez que se desarrollan los primeros sistemas para llevar un registro detallado de los intercambios comerciales bajo la forma de las llamadas "bullas de arcilla". Estas "bullas" eran una de las primeras formas de contabilidad de la administración. Eran unas bolas de barro en cuyo interior se albergaban unas fichas utilizadas en el registro de los trueques, siendo esa precisamente la función originaria para lo que se llevó a cabo la invención de la escritura.

Salteamos varios siglos y llegamos al término "cálculo" el cual procede del latín *calculus*, piedrecita que se mete en el calzado y que produce molestia como los cálculos renales. Precisamente tales piedrecitas ensartadas en tiras constituían el *ábaco* romano que, junto con el *suanpan* chino, constituyen las primeras máquinas de calcular en el sentido de contar (pero no mecánicas). En el siglo XVII el cálculo conoció un enorme desarrollo siendo los autores más destacados Descartes, Pascal y, finalmente, Leibniz y Newton con el cálculo infinitesimal que en muchas ocasiones ha recibido simplemente, por absorción, el nombre de cálculo.

Retomando el hilo original, Blaise Pascal es quien en **1642** concibe con 19 años la *pascalina o "rueda de pascal"*. En estas máquinas hechas a mano los datos se representaban mediante las posiciones de los engranajes e inicialmente solo realizaba sumas, y finalmente restas. 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. El mayor inconveniente fue su elevado coste debido a lo laborioso de su confección manual, se realizaron 50 de las que subsisten 9.

Ya en pleno siglo XX, finalizada la segunda guerra mundial y vista la gran importancia de contar con sistemas complejos capaces de resolver problemas combinatorios y criptográficos (Alain Turing, *Enigma*), se desata un gran interés por el desarrollo de computadores automáticos. En **1949** comienza a operar la EDVAC (*Electronic Discrete Variable Automatic Computer*) por sus siglas en inglés, concebida por John von Neumann. Construida por el laboratorio de investigación de **balística** de Estados Unidos de la Universidad de Pensilvania. A diferencia de sus precursoras no era decimal, sino binaria, y tuvo el primer programa diseñado para ser almacenado electrónicamente. La computadora realizaba operaciones de adición, sustracción y multiplicación automática y división programada. También poseería un verificador automático con capacidad para mil palabras (luego se estableció en 1024 = 2^{10}). El costo de la EDVAC fue justo por debajo de los 500 000 U$S.

For the record, la EDVAC requería de treinta personas por cada turno de ocho horas para poder ser operativa. Poseía físicamente casi 6000 válvulas termoiónicas y 12000 diodos. Consumía 56 kilowatts de potencia. Cubría 45,5 m² de superficie y pesaba 7850 kg.

Finalmente, y centrándonos en PCs de mesa, llegamos en los albores del siglo XXI a la primer PC que consta de una tarjeta gráfica NVIDIA con refrigeración líquida (agua) - *watercooled*, la Scan 3XS Cyclone PC que hizo sus primeros pinitos en el 2010 por la módica suma de £1647, incluyendo impuestos!

Esta PC de 64-bit constaba de un CPU Intel Core *i7* 920 pre-*overclocked*[1] a 4GHz (2.6GHz de fábrica, 4 núcleos - 8 threads), 6GB de RAM DDR3, 1TB HDD, y una tarjeta gráfica NVIDIA GeForce GTX 480, cuyo GPU también estaba pre-*overclocked* operando a 852MHz (701MHz de fábrica, 480 núcleos de CUDA).
Lo único es que venía con Win7 **:-(**

Para más detalles el lector interesado puede visitar la web:

http://hexus.net/tech/reviews/systems/25017-scan-3xs-cyclone-pc-first-watercooled-geforce-gtx-480/

[1] *overclocked: run (the processor of one's computer) at a speed higher than that intended by the manufacturers.*

Sistema Operativo Unix

Contenido

¿Qué es un sistema operativo? **4**

Breve historia de Unix **6**

Arquitectura del OS Linux **10**

Inicio/Salida de sesión SO Unix **12**

Cambio de contraseña **12**

Formato de órdenes de Unix **13**

El sistema de ficheros Unix **13**

Estructura de directorios Unix **14**

Manejo de archivos y directorios **15**

Enlaces a ficheros (direc/simbol) **16**

Múltiples nombres de archivo **16**

Comillas y caracteres especiales **17**

1. ¿Qué es un sistema operativo?

Un sistema operativo (OS, por sus siglas en inglés) 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 *hardware* del sistema de una manera segura, eficiente y abstracta.

Fig. 1.1: Interacción entre el sistema operativo con el resto de las partes.

Como ejemplo de recursos del sistema podemos mencionar: CPU, tarjetas de red, discos de memoria, módems, impresoras, etc..

La CPU es la unidad de procesamiento central (*Central Processing Unit*, por sus siglas en inglés)[2]. El CPU es el hardware dentro de un ordenador u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida (E/S) del sistema.

- El SO asegura un acceso seguro p.ej. impresora permitir que solo sea enviada información por un programa a la vez.
- El SO fomenta el uso eficiente de la CPU mediante la suspensión de los programas que están en espera de operaciones de E/S para completar su ejecución, y así dar paso a otros programas que pueden utilizar la CPU de forma más productiva.
- El SO proporciona abstracciones convenientes tales como archivos en lugar de las ubicaciones de los datos en las posiciones de memoria en los discos, estas aíslan a los programadores y usuarios de los detalles del hardware subyacente.

[2] Los primeros ordenadores, antes del EDVAC, tenían que ser físicamente recableados para realizar diferentes tareas, lo que hizo que estas máquinas se denominen "ordenadores de programa fijo". El EDVAC fue el primer ordenador en contar con una CPU ya que poseía la capacidad de almacenar programas.

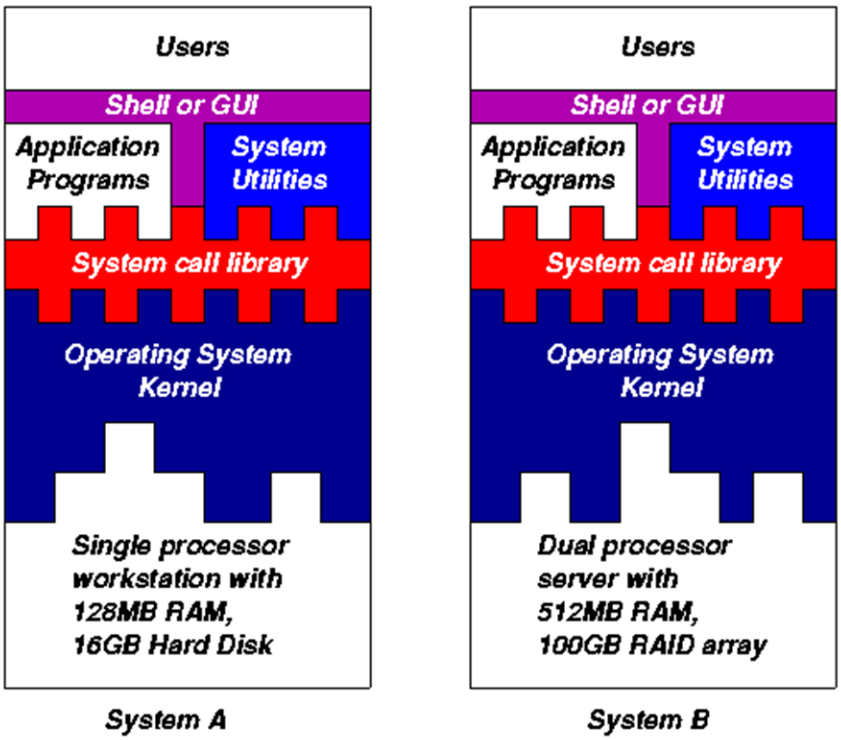


Fig. 1.2: Arquitectura genérica de un SO.

En la Fig. 1.2 se muestran la arquitectura de un SO típico, se observa cómo el SO presenta una interfaz uniforme de cara a los usuarios y los programas sin tener en cuenta los detalles del *hardware* subyacente (p. ej. 1 ó 2 procesadores, características de la memoria, etc.). Se puede observar:

- El **núcleo del** SO controla de forma directa el hardware subyacente.
- El núcleo es un programa que permite controlar dispositivos, memoria y la gestión del procesador a bajo nivel (p. ej., interrupciones de dispositivos del *hardware*, compartir el procesador entre varios programas, asignar memoria a programas, etc.).
- Servicios básicos del núcleo del SO están disponibles para programas de nivel superior a través de una biblioteca de **llamadas al sistema**.
- **Los programas informáticos o aplicaciones** (procesadores de texto, hojas de cálculo) y **programas de utilidades del sistema** (programas simples pero muy útiles y eficientes que vienen por defecto con el OS, p.ej. buscadores de texto en archivos) hacen uso de llamadas al sistema. Las aplicaciones/programas y las utilidades del sistema se ponen en marcha mediante una ***shell*** (interfaz de órdenes de texto) o una **interfaz gráfica de usuario** que proporciona una interacción directa (*mouse*).

2. Breve historia de Unix

UNIX ha sido un OS popular durante más de 4 décadas debido a que brinda un entorno con excelentes capacidades, tales como:

- Multiusuario
- Multitarea
- Estabilidad
- Portabilidad
- Altas prestaciones para trabajo en red

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

UNIX ➔ *No user friendly*

A continuación presentamos una muy simplificada historia de cómo UNIX se ha desarrollado (para tener una idea de lo complicadas que pueden resultar ser las cosas consulte el sitio web http://www.levenez.com/unix/).

- **1960**: General Electric + MIT + Bell Labs (AT&T[3]) desarrollan MULTICS[4].
  - OS multi-usuario y multitarea en ordenadores centrales (unas cajas grandes).
  - MULTICS: ***MULT****iplexed* ***I****nformation and* ***C****omputing* ***S****ystem.*
- **1969**: Ken Thompson (Bell Labs)
  - Crea un OS 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 órdenes cortas: **ls, cp, mv…**
  - El lenguaje de programación en que fue escrito UNICS se llamaba B.
  - UNICS fue creado como un OS para jugar *Space Travel*.
- **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 (Berkeley Software Distribution de la UCB) ➔Incompatibles!

[3] AT&T: *American Telephone and Telegraph Company*
[4] MULTICS: Sistema de Información Informática Multiplexado.

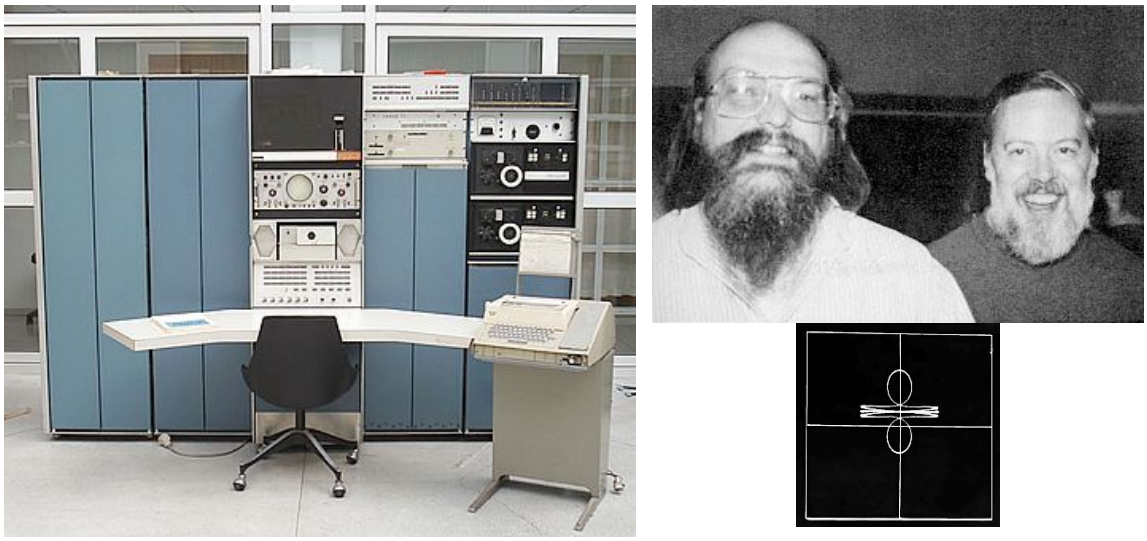


Fig. 1.3: *Izquierda*: PDP-7 mini PC producida por DEC en 1965 (foto museo en Oslo, Noruega). *Derecha arriba*: Ken Thompson y Dennis Ritchie. *Derecha abajo*: captura de pantalla del video juego *Space Travel*.

Algunos comentarios que pueden arrojar un poco de luz sobre los desarrollos mencionados anteriormente:

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 como los ordenadores de esa época.

Aunque a simple vista MULTICS fracasó (para algunos entusiastas de MULTICS "fracaso" es quizás una palabra demasiado fuerte), en realidad esto inspiró a Ken Thompson, que era un investigador en los Laboratorios Bell, a escribir un OS más sencillo. Él escribió una versión más simple de MULTICS en un PDP7 en código ensamblador y llamó a su intento como UNICS (Sistema de Información Informática y Uniplexed). Debido a que la memoria y potencia de CPU eran un bien muy escaso en esos días, UNICS (con el tiempo acortado a UNIX) utiliza órdenes cortas para minimizar el espacio necesario para almacenarlas y el tiempo necesario para decodificar estas - de ahí la tradición de órdenes cortos de UNIX que utilizamos hoy en día, por ejemplo: ls, cp, rm, mv, etc.

A Ken Thompson luego se unió Dennis Ritchie, el autor del primer compilador de C ya por 1973. Entre ambos reescribieron el núcleo de UNIX en C - este fue sin duda el gran paso adelante en términos de portabilidad del sistema - y fue cuando se 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: SYS**V** (Sistema 5) y BSD (Berkeley Software Distribution). BSD surgió de la Universidad de California en Berkeley (de las pocas públicas y de primera línea de EEUU), donde Ken Thompson pasó un año sabático y por lo visto muy productivo. 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 conservadoras, si bien han incorporado mejoras de desarrollos basados en BSD (memoria virtual, gestión de redes, utilidades como vi, csh, etc.).

Las últimas versiones de SYSV (o SVR4 Sistema 5 Release 4) y BSD Unix son en realidad muy similares.

Algunas diferencias menores entre SYSV y BSD se encuentran en la (ver Fig. 1.4):

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

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

Fig 1.4 : Diferencias entre SYSV y BSD.

En suma podemos describir los desarrollos de UNICS en el árbol que se muestra en la Fig. 1.5.

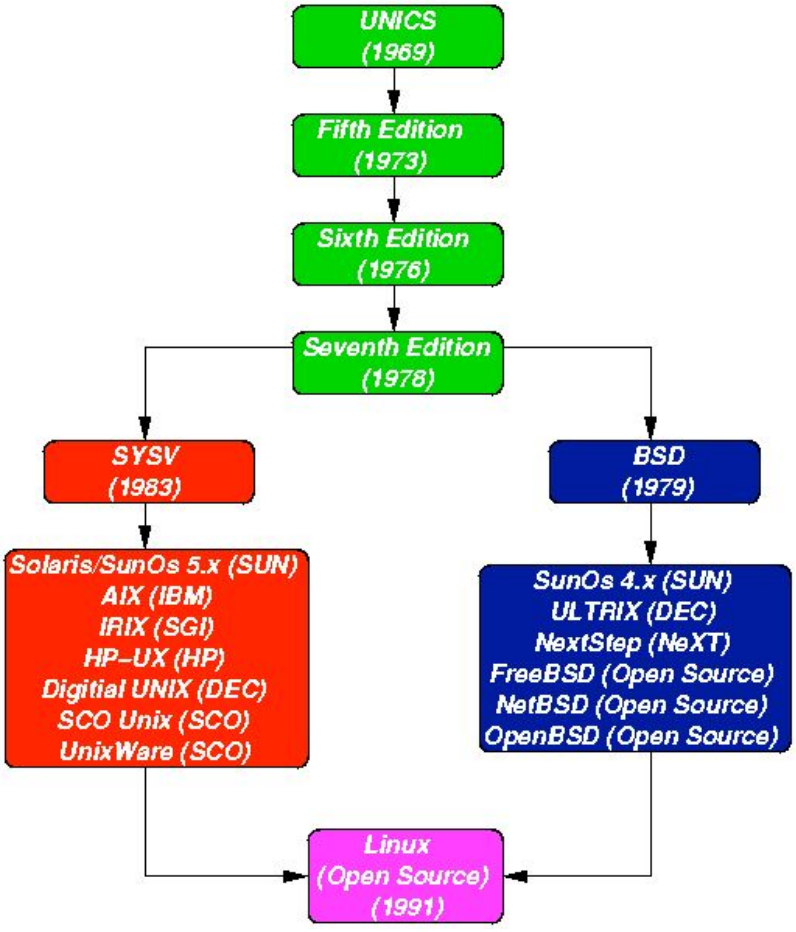


Fig. 1.5 : Árbol familiar de UNIX simplificado.

Para completar la historia nos falta comentar sobre el último cuadro de la Fig. 1.5, el OS **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).
  - 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 no sólo ha sido aplicado con éxito al núcleo del código, sino también a los programas de aplicación para Linux.

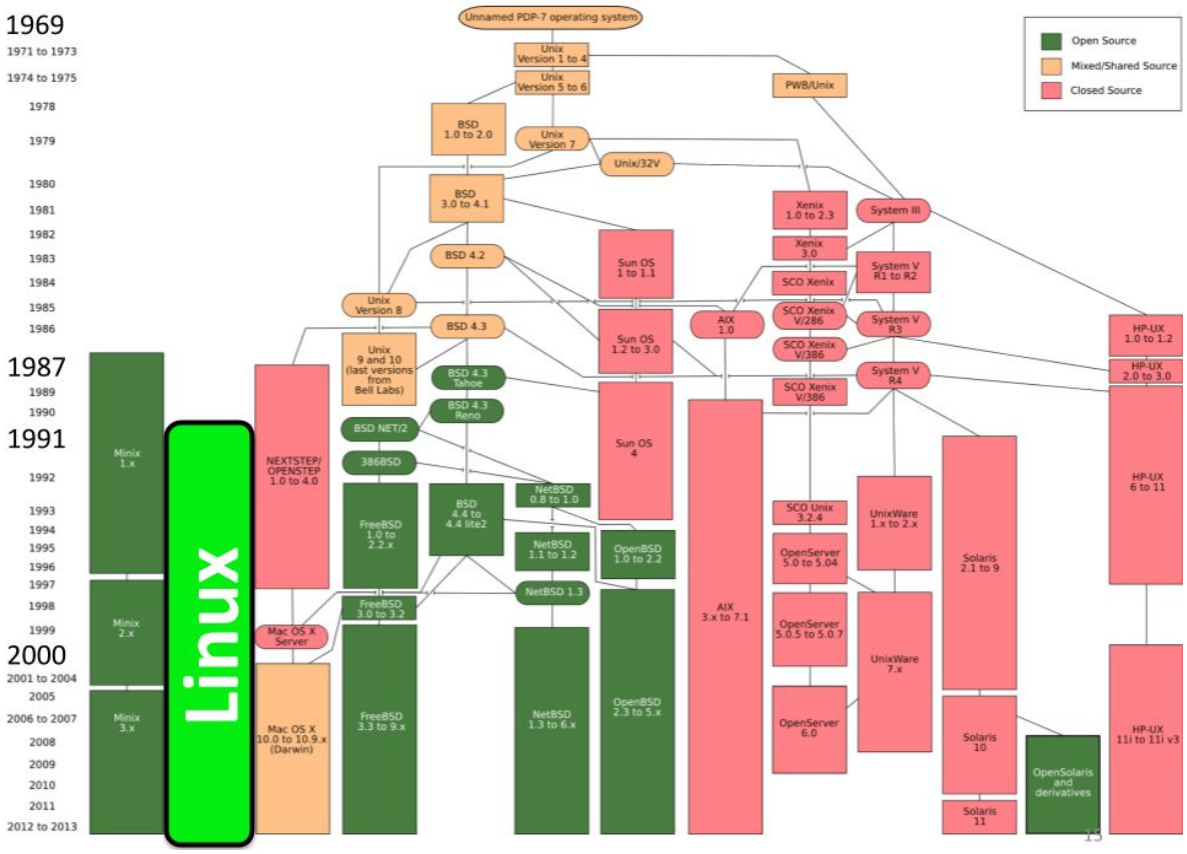


Fig 1.6 : Árbol familiar de UNIX y sistemas tipo UNIX, se destaca la aparición y el lugar que ocupa Linux.

3. Arquitectura del OS Linux

Linux tiene todos los componentes de un OS tipo UNIX (ver Fig 1.1):

- Núcleo: El núcleo (*kernel*) Linux incluye soporte para controlar un gran número de dispositivos del *hardware* de la PC (tarjetas gráficas, tarjetas de red, discos duros, etc.), así como características avanzadas del procesador y gestión de la memoria, y posee soporte para manipular muchos tipos diferentes de sistemas de archivos (incluyendo disquetes *floppy* DOS, la ISO9660 estándar para CDROMs, discos externos usb...). En cuanto a los servicios que presta a los programas de aplicación y utilidades del sistema, el núcleo implementa la mayoría de las llamadas al sistema de BSD y SYSV, así como las llamadas del sistema descritos en la memoria POSIX[5]. El *kernel* (código binario 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. La última versión del fuentes del kernel de Linux pueden descargarse desde http://www.kernel.org.

[5] POSIX (*Portable Operating System Interface*, X viene de UNIX) es una norma escrita por la IEEE. Dicha norma define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de órdenes (*shell*), y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones a nivel de código fuente.

- Shells y GUIs. Linux soporta dos formas de entrada de órdenes:
  - Intérpretes de línea de órdenes (*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.

  Si va a conectarse de forma remota a un servidor su acceso será típicamente a través de una *shell* mediante línea de órdenes.
- 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) ha sido portado a Linux. Esto incluye órdenes como ls, cp, grep, awk, sed, bc, wc y otros. Estas utilidades del sistema están diseñados para ser herramientas poderosas que hacen una sola tarea extremadamente bien.
  - **cp** copia, **grep**: busca expresiones regulares (caracteres), **wc:** cuenta el número de palabras, líneas y bytes dentro de un archivo, **awk**: procesa datos definidos en archivos de texto, **sed**: editor de flujo de texto, demonios, etc

  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[6] que proporcionan servicios de red y administración remota (por ejemplo, telnetd y sshd proporcionan facilidades de acceso remoto, LPD proporciona servicios de impresión, httpd administra páginas web, crond ejecuta tareas de administración de sistemas regulares de forma automática). Un *daemon* (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" en inglés) por lo general se ejecuta 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.
- Programas de aplicación. Las distribuciones de Linux (*distros*) normalmente vienen con varios programas de aplicaciones útiles de forma estándar. Algunos ejemplos:
  - **emacs**: editor de texto, **gcc/g++**: compilador de C/C++, **latex**: lenguaje de composición de texto, **xfig**: paquete de dibujo vectorial, **StarOffice** que es un clon del MS-Office que puede leer y escribir archivos de Word, Excel y PowerPoint (ahora *libreoffice*).

[6] La palabra daemon fue utilizada por primera vez en 1963, 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.

4. 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 órdenes muestra algo como esto:

  $ (el cursor parpadea ➔ esperando instrucción)

- Para salir de la sesión basada en texto, escriba **exit** en el intérprete de órdenes (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 órdenes 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**.

5. 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. La orden 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.

6. Formato de órdenes de Unix

- Una línea de órdenes de UNIX consiste en el nombre de una orden de UNIX (en realidad la "orden" 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 una orden de UNIX es:

  $ orden –opciones objeto

- Aquí **orden** 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).

7. 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 OS en sí mismo, los archivos ejecutables para las órdenes soportados por el OS, 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.

8. 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.

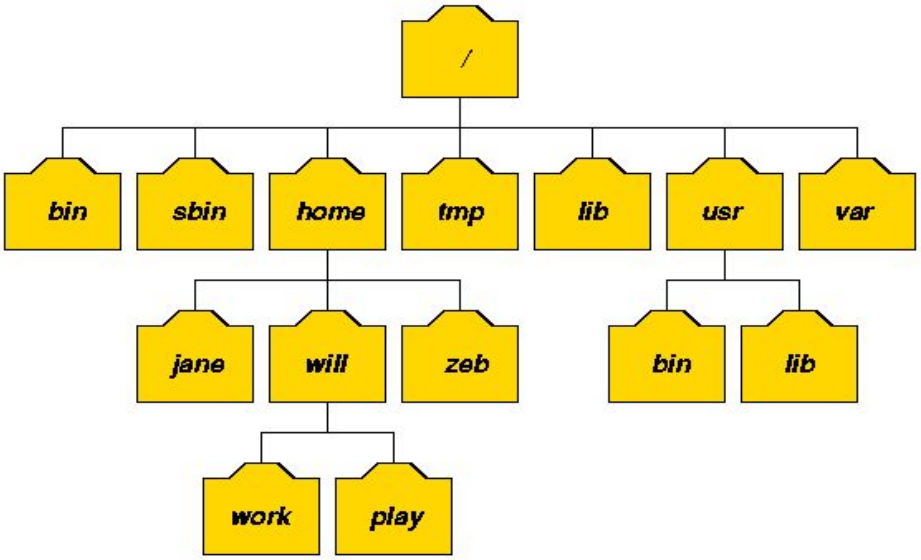


Fig1.7: Árbol de directorios de un típico sistema basado en UNIX.

- **/** 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).

9. Manejo de archivos y directorios

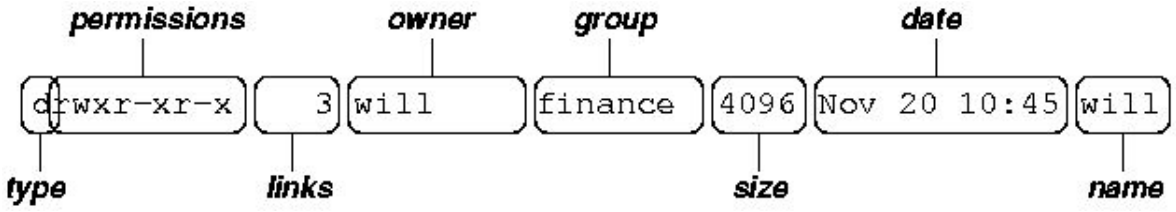
Algunos de las órdenes más importantes para la manipulación de archivos y directorios:

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 las órdenes puedes utilizar **man** o **info.**

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

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

A continuación se listan algunos de las órdenes 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.
mv	se utiliza para cambiar el nombre de los archivos / directorios y / o moverlos de un directorio a otro.
cp	se utiliza para hacer copias de archivos o directorios completos.
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 mediante **/**.
less	similar a ***more*** pero con características adicionales como desplazar hacia atrás.

Nota: Siempre se debe contar con los permisos necesarios!

10. 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 la orden **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.

11. 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 comience 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 una orden ***antes*** de ejecutar la orden.

12. 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 las órdenes 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 alguna orden como un argumento de entrada a otro.
- Por ejemplo:

  $ **hostname**

  pepe

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

  esta máquina se llama pepe