Tema 1: Shell

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

Curso 2020/2021

Cristian Martín Fernández

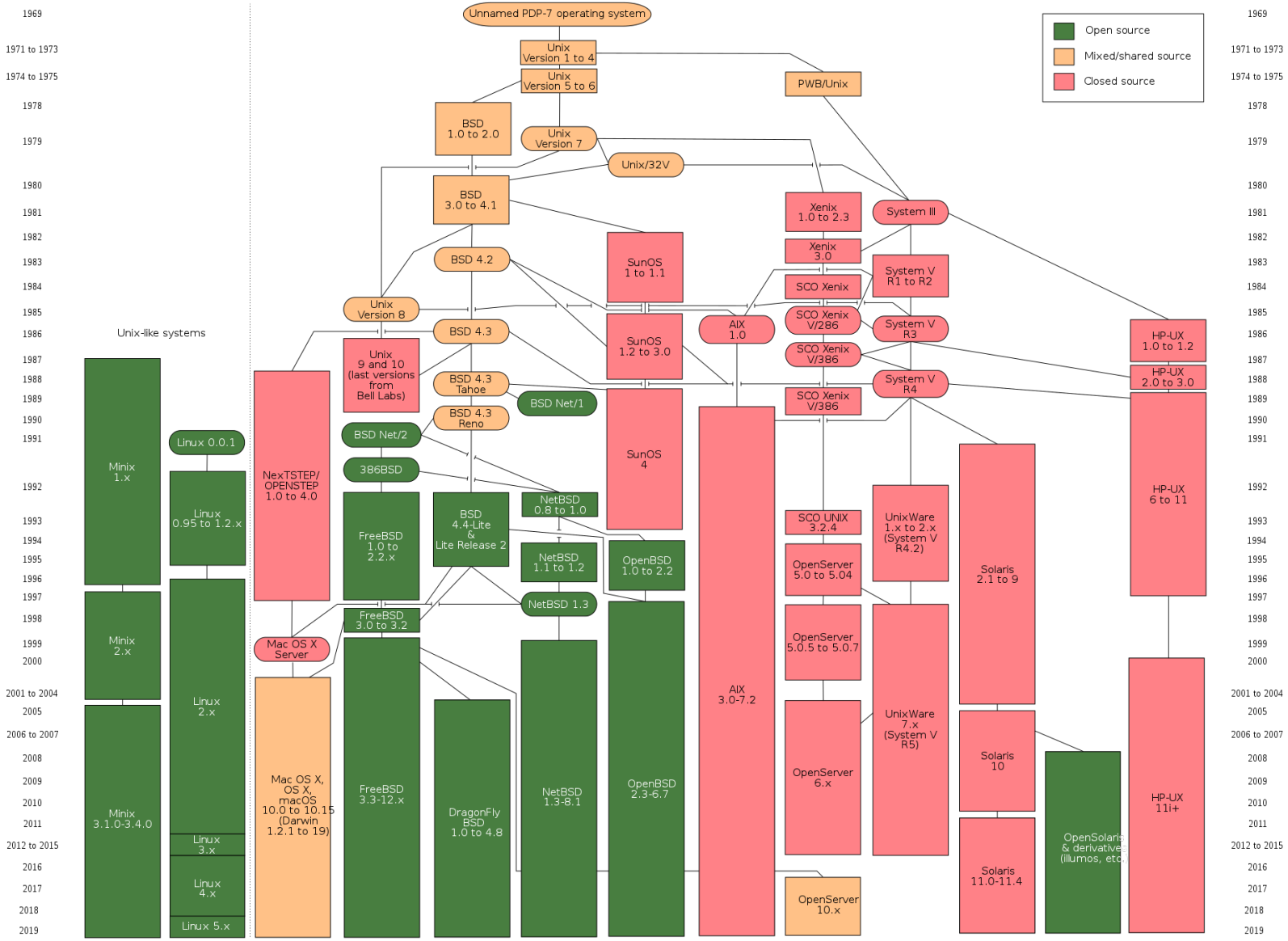
Contenido

- Introducción al S.O Unix
- Estructura de UNIX
- Unix desde el punto de vista de usuario
- El Shell

INTRODUCCIÓN AL S.O. UNIX

- Los orígenes de UNIX se remontan a finales de los años 60 en los laboratorios Bell, de AT&T (sistema muy avanzado para la época).
- Ken Thompson, escribió lo que fue la versión primitiva de UNIX (Multics). Dennis Ritchie comenzó a colaborar con él.
- En 1973, el sistema operativo fue reescrito en (B->C) y usado en ordenadores PDP-11.
- AT&T cedió UNIX a las universidades para uso educacional.
- Su simplicidad y claridad desembocó en muchas versiones System V (AT&T) y BSD (Berkeley), incompatibles entre si, y las más utilizadas en los 90.

INTRODUCCIÓN AL S.O. UNIX



INTRODUCCIÓN AL S.O. UNIX

- **Interactivo**:
  - El sistema acepta órdenes, las ejecuta y se dispone a esperar otras nuevas.
- **Multitarea**:
  - Puede realizar varios trabajos, denominados procesos, al mismo tiempo.
- **Multiusuario**:
  - Más de una persona puede usar el sistema al mismo tiempo
- **Sistema simple y consistente** (ls A*, rm A*, etc.)
- **Flexible y potente**

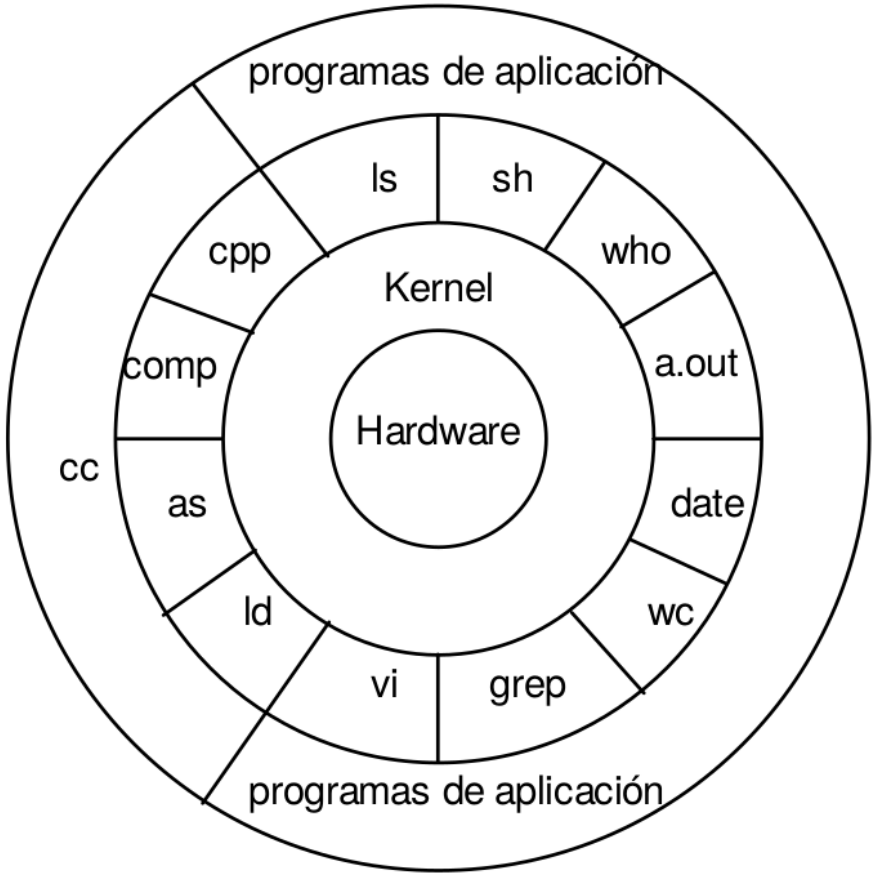
INTRODUCCIÓN AL S.O. UNIX

- UNIX fue diseñado por programadores para ser usado por programadores en un entorno en que los usuarios son relativamente expertos, como programadores.
- Este enfoque es muy distinto al de un sistema operativo de un computador personal.
- Herramientas de Unix: make, control de versiones, …

ESTRUCTURA DE UNIX

- La estructura de UNIX se amolda a un típico **modelo en capas**.
- El núcleo (**kernel**) del sistema interactúa directamente con el hardware y proporciona una serie de servicios comunes a los programas de las capas superiores.
- Programas como el intérprete de comandos (shell) sh y el editor vi interactúan con el kernel a través de llamadas al sistema, que permiten el intercambio de información entre el kernel y los programas.

ESTRUCTURA DE UNIX



PARTES PRINCIPALES DE UNIX

- El **kernel** es la parte del sistema que gestiona los recursos del computador (el hardware).
- El **sistema de ficheros** organiza la información en disco y es de tipo jerárquico.
- El **shell** es un programa de utilidad que permite al usuario comunicarse con el sistema.

UNIX (PUNTO DE VISTA DEL USUARIO)

- Conexión y desconexión al sistema:
  - Aplicación login -> usuario + contraseña
  - UNIX conoce qué usuario es el dueño de cada fichero -> seguridad (acceso únicamente por usuarios autorizados).

UNIX (PUNTO DE VISTA DEL USUARIO)

- **/etc/passwd**
  - <nombre>,<password><uid>, <gid>,<descripción opcional> <carpeta>, <shell>

<nombre> Nombre de usuario
<password> Contraseña. Una «**x**» indica que está cifrada en /etc/shadow. Una «**!**» es que el usuario está bloqueado. Si tiene «**!!**» es que no tiene.
<uid> Identificador entre 0(root) y 65535.
<gid> Identificador de grupo (el cero ser reserva para el grupo root)
<carpeta> Directorio de inicio del usuario.
<shell> Intérprete de comandos del usuario. Si ponemos /bin/false el usuario no podrá ejecutar ningún comando del sistema

UNIX (PUNTO DE VISTA DEL USUARIO)

- **/etc/shadow**
  - <nombre><password cifrado><1><2><3><4><5><6>

<nombre> Nombre de usuario
<password> Contraseña.
1- Días transcurridos desde 1-1-1970 hasta el último cambio de contraseña
2- Nº de días que hay que pasar para poder cambiar la contraseña
3- Días máximos de validez de la cuenta.
4- Días que avisa antes de caducar la contraseña.
5- Nº de días con contraseña caducada antes de deshabilitar la cuenta.
6- Nº de días desde el 01/01/1970

EL SHELL

- Un shell es un programa de usuario más cuyas únicas habilidades son **leer del teclado, escribir en la pantalla y poder ejecutar otros programas**.
  - **Bourne shell**: es el shell tradicional de UNIX y existe en todas los sistemas (sh)
  - **C shell**: posterior al Bourne shell, fué diseñado para que los programas escritos en el lenguaje del shell tengan la apariencia de programas escritos en C (csh)
  - **Korn shell**: es un superconjunto del Bourne shell e incorpora algunas de las funciones más útiles del C Shell (ksh)

EL SHELL

- Cuando el shell lee una línea de comandos, extrae la primera palabra, asume que ésta es el nombre de un programa ejecutable, lo busca y lo ejecuta.
- Formato de las órdenes:
  - Nombre de orden – opciones – parámetros
    - `sort -r file`
- Las opciones (-) modifican el comportamiento normal de la orden
  - `ls -a -l -s` es equivalente a `ls -als`.
- Los parámetros aportan información adicional que será usada por la orden (ficheros, directorios, etc.)

EL SHELL

- La mayoría de las órdenes aceptan múltiples opciones y argumentos
  - `ls file.txt file1.txt`
  - `head -20 file.txt`
  - `head 20 file.txt`
  - `rm *.c`
  - `rm * .c`

REDIRECCIÓN

- El shell, como la mayoría de las órdenes de UNIX, emplean tres ficheros denominados “estándar”.
  - Entrada estándar, salida estándar y error estándar (salida de errores)
- En UNIX los dispositivos se tratan como ficheros, por lo que estos tres ficheros, si no se indica lo contrario, son dispositivos
  - La entrada estándar es el teclado, mientras que la salida y error estándar es el monitor
- Es posible encadenar varias órdenes de forma que la salida estándar de una orden se dirija hacia la entrada estándar de la siguiente (**interconexión por tuberías**)

REDIRECCIÓN

- Salida estándar:
  - `ls -al > listado`
  - `cat file.txt file1.txt > fich3`
  - `cat file.txt file1 >> fich3` (concatenación en fich3)
- Entrada estándar:
  - `cat < file.txt`
  - `sort < file.txt > file2.txt`
- Error estándar:
  - `ls 2>error.log`
  - `ls 2>/dev/null` (descartar información)

REDIRECCIÓN

- Cauces o tuberías (pipes) -> encadenar varias órdenes y/o ficheros
  - `who | wc -l`
  - `grep al *.txt | sort | head -20 | tail -5 > f_sal`
  - `ps aux | grep python` (muy común)
- La mayoría de las órdenes de UNIX son filtros (entrada-salida estándar), por lo que pueden usarse como operaciones intermedias en un cauce.
- ls sólo podrá estar al comienzo de un cauce y lp al final.

METACARACTERES

- Los metacaracteres se emplean para establecer una correspondencia con nombres o partes de nombres de ficheros. Los más empleados son:
- *: representa cualquier cadena de caracteres, incluyendo la cadena
- ?: representa a cualquier carácter simple.
- [ ]: una lista de caracteres encerrada entre corchetes especifica que la correspondencia es con cualquier carácter simple de la lista.
- -: el guión se utiliza dentro de los corchetes para indicar un rango de caracteres.
  - `ls *`
  - `ls file[0-1].txt`

PROCESOS

- UNIX es un sistema operativo multitarea y multiusuario, lo que implica que puede atender más de un trabajo a la vez, para cualquier usuario.
- Cada uno de estos trabajos se denomina **proceso**. Un proceso de puede definir como un programa en ejecución
- Los procesos se pueden ejecutar en primer plano (foreground), bajo el control de usuario, o en segundo plano (background).
- Identificado por el PID único
- También hay trabajos que se ejecutan siempre (daemons).

PROCESOS EN BACKGROUND

- El siguiente comando se ejecuta en background, pero cuando termina la sesión de consola, este muere:
  - `./programa.sh &`
- Una forma de evitar esto y dejar comandos en ejecución de forma prolongada es utilizar los siguientes comandos:
- `nohup ./programa.sh &` (no permite entrada de usuario)
- `screen ./programa.sh &` (es más pesado pero permite entrada y más funcionalidades, es necesario instalarlo)

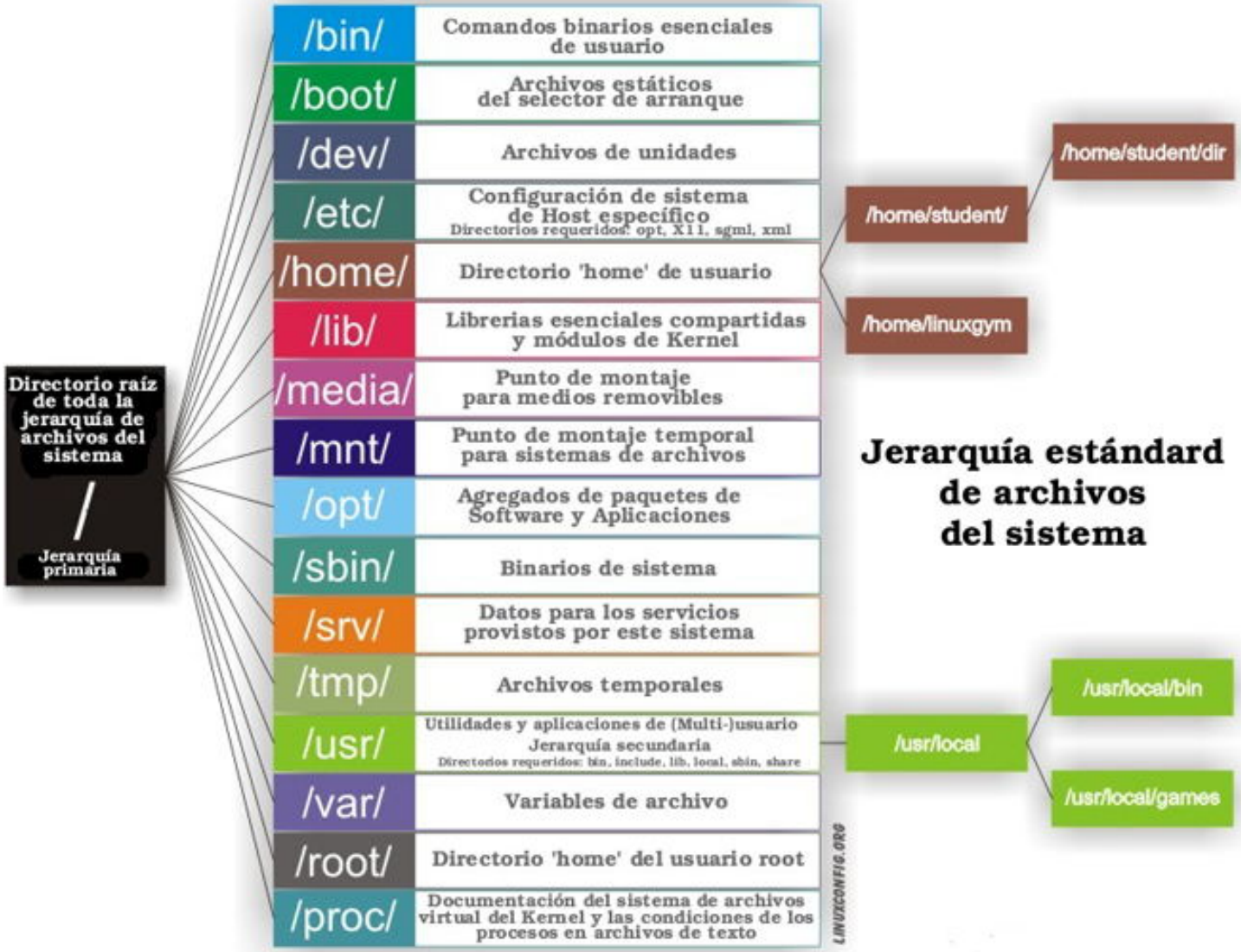
SEÑALES

- Las señales son un mecanismo que se usa para notificar a los procesos la ocurrencia de **sucesos asíncronos**.
- Estos sucesos pueden ser una división por cero, la muerte de un proceso hijo, la solicitud de terminación del proceso por parte del usuario, etc.
- Cuando se produce alguno de estos sucesos, el proceso en cuestión recibe una señal, lo que causa la interrupción de la ejecución del proceso con el fin de que la señal pueda ser tratada.
- Los procesos puede decirle al sistema la acción a tomar cuando reciban una determinada señal. Las opciones son
  - ignorar la señal
  - aceptar la señal
  - o dejar que sea el sistema el que trate la señal, que es la opción por defecto.

EL SISTEMA DE FICHEROS

- Un fichero UNIX es una secuencia de 0 o más bytes
- El sistema no distingue entre ficheros ASCII, binarios, o cualquier otro tipo. La interpretación del contenido de los ficheros se deja a los programas que los utilizan.
- Las mayúsculas y las minúsculas son significativas (RESUMEN, resumen, y ResumeN)
- Tipos de ficheros:
  - Ordinarios (regulares)
  - Directorios
  - Especiales o de dispositivo

EL SISTEMA DE FICHEROS

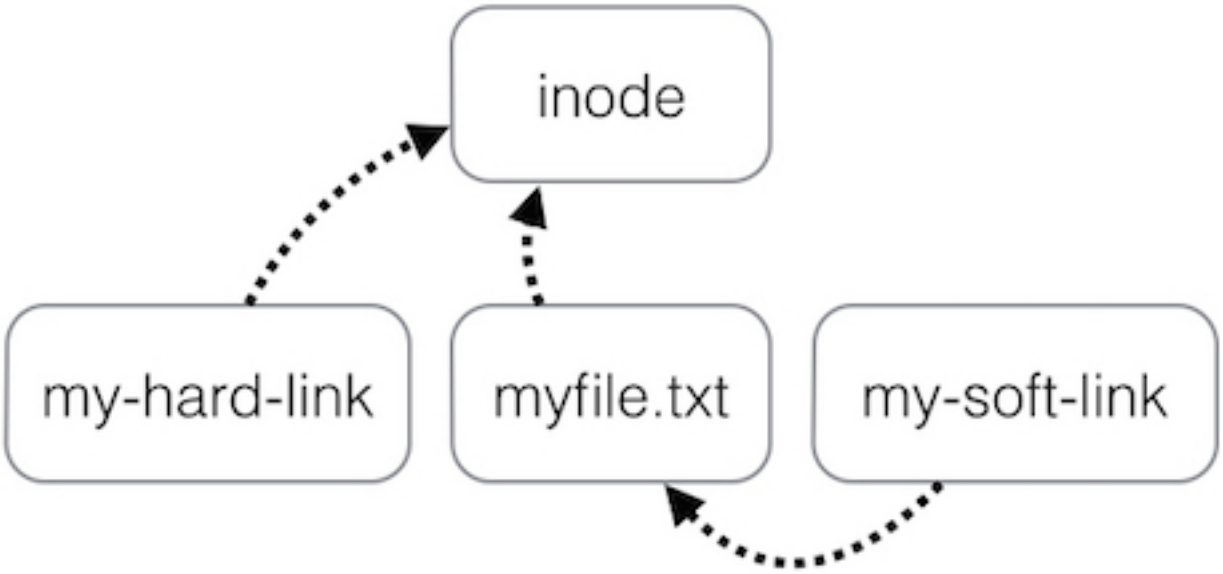


EL SISTEMA DE FICHEROS

- Directorio raíz (root): /
- Directorio actual: .
- Directorio padre: ..
- Direccionamiento absoluto:
  - `/usr/bin/ls`
  - `/home/cristian/users/practicas/.cshrc`
- Direccionamiento relativo:
  - `../../bin/ls`
  - `.cshrc`

EL SISTEMA DE FICHEROS

- Cada archivos y carpeta tiene un Inodo único.
- Enlaces simbólicos y duros:



EL SISTEMA DE FICHEROS

- Se podrá realizar un enlace duro de un archivo siempre y cuando el archivo esté en la misma partición del disco duro que pretendemos crear el enlace.
- Para borrar un fichero hay que borrar todos los enlaces duros.
- El acceso al contenido a través de un enlace duro es más rápido que en los enlaces simbólico.
- Los enlaces simbólicos se pueden realizar a ficheros o directorios y en diferentes particiones
- Los enlaces simbólicos ocupan más tamaño en disco

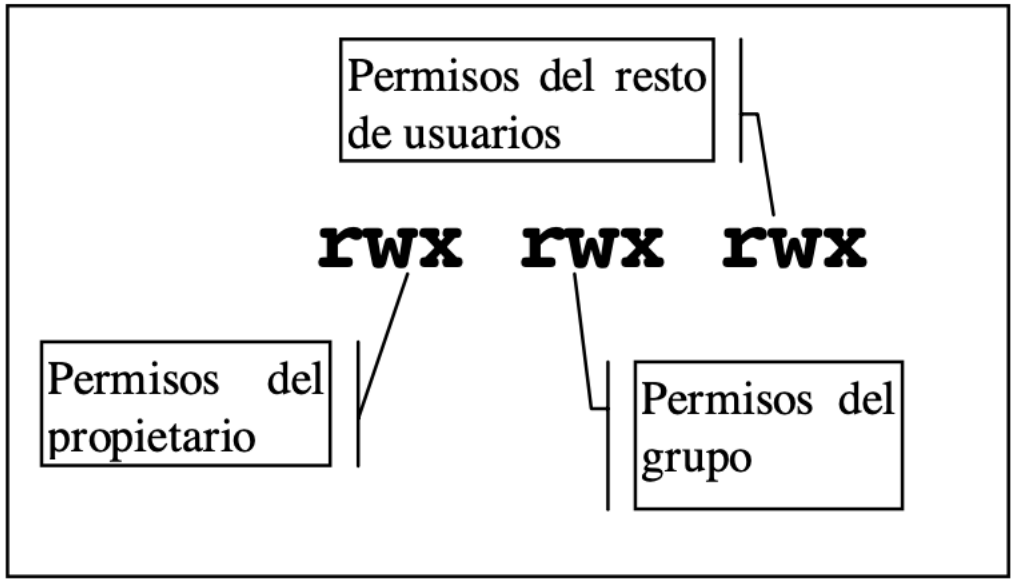
EL SISTEMA DE FICHEROS

- Enlaces duros
  - `ln file.txt fileduro.txt`
- Comprobamos el inode del fichero:
  - `ls -li`
- Enlaces simbólicos
  - `ln -s /home/cristian/shell/file.txt /home/cristian/enlacsim.txt`
- Con el comando cp y la opción –rl y opción –rs se pueden crear enlaces duros y simbólicos de múltiples archivos respectivamente

SEGURIDAD Y PROTECCIÓN

- Todos los ficheros en UNIX tienen un propietario, que normalmente es el usuario que los crea. Una vez creado, el propietario puede asignar permisos que permitan o impidan el acceso al fichero. Se distinguen tres tipos de usuarios:
  - **propietario**: la persona que crea el fichero, aunque es posible ceder la propiedad a otro usuario.
  - **grupo**: varios usuarios pueden formar parte de un mismo grupo.
  - **público**: el resto de los usuarios del sistema.

SEGURIDAD Y PROTECCIÓN



SEGURIDAD Y PROTECCIÓN

- Ejemplo:
  - r– r– r– (fichero solo lectura)
  - rw-r----- (lectura y ejecución propietario, lectura grupo).
- Modificar valores por defecto (666 ficheros 777 directorios):
  - umask [máscara]
- `chmod 700 file.txt` (permiso total (rwx al usuario propietario) Forma octal
- `chmod u=rwx,g=rwx,o= file.txt` (permiso rwx al usuario propietario y al grupo)
- Nunca usar **chmod 777!!**

COMANDOS ÚTILES

- Información sobre comandos:
  - `man comando`
- Mostrar información de una carpeta
  - `ls carpeta`
- Terminar un proceso:
  - `kill -9 proceso` (se puede usar la opción -15, menos agresiva)
- Primeras y últimas líneas de ficheros
  - `head/tail`
- Para ver los tipos archivos:
  - `file <dir o file>`

COMANDOS ÚTILES

- El comando **xargs** permite ejecutar una acción desde una lista de información.
- Por ejemplo, mostrar el contenido de los ficheros encontrados acabados en .txt
  - `find *.txt | xargs cat`
- AWK es una herramienta de procesamiento de patrones en líneas de texto.
- Mostrar el nombre de fichero y su tamaño:
  - `ls -l | awk '{ print $9 ":" $5 }'`
- Imprimir ficheros con tamaño mayor que 100:
- `ls -l | awk '$5 > 1000000 { print $8 ":" $5 }'`

COMANDOS DE BÚSQUEDA

- Buscar ficheros que contenga la c
  - `find -name *.c -type f`
- Buscar todos los ficheros del sistema cuyo tamaño sea mayor de 100 bloques:
  - `find / -size +100`
- Contar el número de directores en un directorio
  - `find /home/cristian/ -type d | wc -l`
- Buscar enlaces simbólicos con profundidad máxima 2
  - `find /home/cristian -maxdepth 2 -xtype l`

ESPACIO DISPONIBLE

- Espacio que ocupa un directorio
  - `du -s /home/cristian`
- Muestra el espacio disponible en el sistema de archivos
  - `df /`