

Repaso comandos Linux

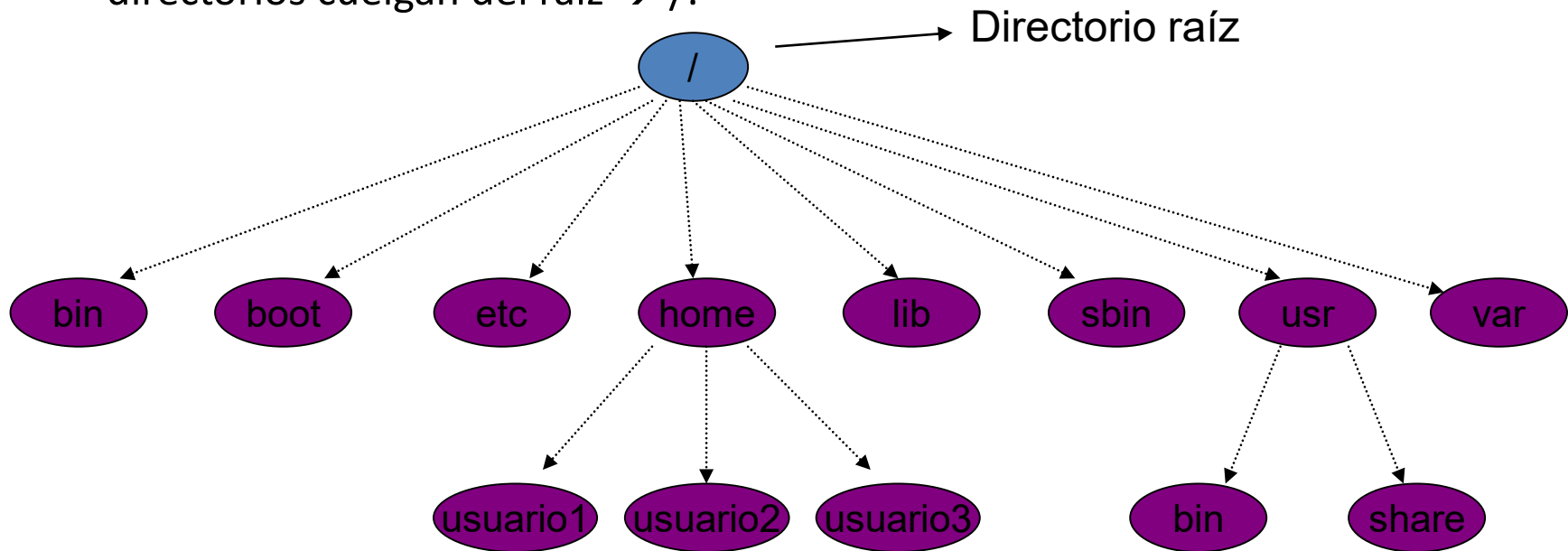
Contenidos

1. Principales directorios y su función
2. Características básicas
3. Manejo de ficheros y directorios
4. Expresiones regulares
5. Desconexión del sistema

1. Principales directorios y su función

1. Principales directorios y su función

- El sistema de ficheros de Linux tiene una estructura en árbol, donde todos los directorios cuelgan del raíz → /.



- Durante la instalación se crean unos directorios, siendo algunos de ellos:
 - **/bin** → programas, ficheros ejecutables por cualquier usuario.
 - **/sbin** → programas que sólo puede ejecutar el superusuario.

1. Principales directorios y su función

- **/boot** → ficheros necesarios para arrancar el sistema, excepto los de configuración. En este directorio se almacenan los ficheros correspondientes al cargador de arranque (grub), el programa de inicialización (init) y el núcleo del sistema operativo. Este último recibe el nombre de *vmlinuz* seguido por una serie de números. Por ejemplo, *vmlinuz-2.6.32-279.el6.i386*.

Las versiones del núcleo se numeran con 3 números, de la siguiente forma: *xx.yy.zz*.

- *xx*: Indica la serie principal del núcleo. Hasta el momento están la 1, 2 y 3. Este número cambia cuando la manera de funcionamiento del núcleo ha sufrido un cambio muy importante.
 - *yy*: Indica si la versión es de desarrollo o de producción. Un numero impar, significa que es de desarrollo, uno par que es de producción.
 - *zz*: Indica nuevas versiones dentro de una versión, en las que lo único que se ha modificado son fallos de programación o bugs.
- **/etc** → ficheros de configuración del sistema.
 - **/home** → directorios personales de los usuarios.
 - **/root** → directorio personal del superusuario.

1. Principales directorios y su función

- **/tmp** → ficheros temporales e información temporal de los programas.
- **/usr** y **/opt** → utilidades y programas instalados.
- **/dev** → ficheros asociados a dispositivos hardware. Cada dispositivo tiene un fichero asociado. Algunos ejemplos son:

crw-rw----. 1 root lp 6, 1 nov 15 18:26	/dev/lp1
brw-rw----. 1 root disk 8, 0 nov 15 18:26	/dev/sda
brw-rw----. 1 root disk 8, 1 nov 15 18:26	/dev/sda1
crw--w----. 1 root tty 4, 1 nov 15 18:26	/dev/tty1
- **/var** → ficheros de eventos del sistema o logs.
- **/proc** → directorio actualizado automáticamente por el sistema. En él se guarda información sobre el núcleo, los procesos y el hardware del sistema. Algunos de estos ficheros pueden ser manipulados por los usuarios y aplicaciones para comunicar al kernel cambios en la configuración.

2. Características básicas

2. Características básicas

2.1 Shell: definición y características

2.2 Ayuda

2.3 Path: definición

2.4 Ruta relativa y ruta absoluta

2.5 Redirecciones

2.6 Conectar varios comandos

2.7 Comodines

2.1 Shell: definición y características

- El shell es el intérprete de comandos: actúa como interfaz entre el usuario y el sistema.
- Hay diferentes intérpretes, algunos de ellos son:
 - Bourne Shell (sh)
 - Bourne Again Shell (bash) → el más usado
 - Korn Shell (ksh)
 - C Shell (csh)
- Una **orden** o **comando** es un fichero ejecutable del sistema. No se envía al sistema hasta que se pulsa la tecla Enter.

orden [-opciones] [argumentos]

- **Opciones:**
 - Van precedidas de un guión. Ej: ls -l
 - Pueden agruparse varias opciones. Ej: ls -la
 - El orden de las opciones no es significativo.
- **Argumentos** o **parámetros:**
 - Van separados por espacio o tabulador.

2.1 Shell: definición y características

- Características:
 - Se **distingue** entre mayúsculas y minúsculas.
 - Permite el **autocompletado** de comandos y ficheros. Se pueden teclear las primeras letras de un comando o nombre de fichero, pulsar la tecla tabulador, y dejar que el sistema complete el comando.
 - Se pueden ejecutar **varios comandos** sin que estén conectados entre sí de ningún modo, tecleándolos en la misma línea y separándolos por punto y coma (;).
 - Permite **encadenar** varias **órdenes**.
 - Se puede **redireccionar** la **E/S**.
 - Dispone de un **historial** de comandos para poder recuperar los ya utilizados mediante las teclas del cursor.
 - Proporciona un lenguaje de programación: **shell scripts**.

2.2 Ayuda

- Hay varias maneras de ver la ayuda sobre una orden, siendo algunas de las más usadas:
 - **man orden** → Permite subir y bajar con las flechas y RePag y AvPag. Al salir (pulsando “q”), desaparece la información de la ventana de terminal. Los documentos de las páginas *man* están almacenados en un formato comprimido. El comando **man** descomprime y formatea estas páginas para su correcta visualización.
 - **info orden** → Muestra la información de forma parecida a como lo hace man.
 - **orden --help** → Muchos comandos disponen también de esta ayuda que se muestra de forma más resumida. La información permanece en la ventana de terminal.

2.3 Path: definición

- Cuando tecleamos una orden, el intérprete de comandos sigue una serie de pasos para encontrarlo:
 1. Busca el nombre de la orden y comprueba si es una orden interna.
 2. Comprueba si la orden es un *alias*, es decir, un nombre sustitutorio de otra orden.
 3. Si no se cumple ninguno de los casos anteriores, busca el programa correspondiente para ejecutarlo.
 4. Si el intérprete de comandos no puede encontrar la orden que hemos tecleado, muestra un mensaje de error.
- Para ejecutar un comando ubicado en el mismo directorio se realiza de la siguiente forma:
 - `./comando`indicando que está en el directorio actual (`.`), sino no encontrará el comando, a pesar de estar en el mismo directorio.
- Los directorios en los que busca las órdenes son los que están en una variable llamada PATH.

2.3 Path: definición

- Se puede consultar el valor actual de PATH con el siguiente comando:
 - `/usr/lib/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/alumno/bin`
 - Ejemplo:
- Para añadir un directorio al path, se puede redefinir la declaración completa o, simplemente, añadir el nuevo directorio con el comando:
 - **`PATH=$PATH:nuevo_directorio`**

2.4 Ruta relativa y ruta absoluta

- Dependiendo de la forma de acceder a una ubicación en el sistema de ficheros tendremos dos tipos de ruta:
 - **Ruta absoluta**: Secuencia de directorios que se ha de recorrer para acceder al fichero desde el directorio raíz.
 - **Ruta relativa**: Secuencia de directorios que se ha de recorrer para acceder al fichero desde el directorio actual.
- Directorios singulares:
 - / → raíz.
 - . → directorio actual.
 - .. → directorio padre.
 - ~ → *HOME* del usuario. (~ → AltGr + 4)
- Ejemplo. Para ir a /home/alumno desde /home/alumno/Documentos:
 - cd .. → ruta relativa
 - cd /home/alumno → ruta absoluta

2.5 Redirecciones

- El kernel abre para cada orden tres archivos:
 - `/dev/stdin` → entrada estándar (0): teclado.
 - `/dev/stdout` → salida estándar (1) : pantalla.
 - `/dev/stderr` → salida de error (2): pantalla.
- Las operaciones de redirección permiten cambiar la E/S estándar.
 - **Redirección de entrada:**
 - `orden < fichero` → toma el fichero como entrada de la orden. No se usa mucho.
 - **Redirección de salida:**
 - `orden > fichero` → guarda en el fichero el resultado de la orden, borrando lo que hubiera en el fichero.
 - `orden >> fichero` → añade al final del fichero el resultado de la orden.
 - **Redirección de salida de error:**
 - `orden 2> fichero` → guarda en el fichero los errores que se puedan producir al ejecutar la orden, borrando lo que hubiera en el fichero.
 - `orden 2>> fichero` → añade al final del fichero los errores que se puedan producir al ejecutar la orden.
 - También se pueden redirigir la salida estándar y la salida de error al mismo sitio, representándolas con el símbolo: `&`.

2.6 Conectar varios comandos

- Tuberías:
 - Las tuberías (*pipes*) conectan la salida estándar de una orden con la entrada estándar de otra.
 - `orden1 | orden2`
 - Se pueden conectar más de dos órdenes.
 - Ejemplos:
 - `ls -l | more`
 - `cat file1 | grep -w hola | cut -d: -f5`
 - `ps -ef | tail -n+2 | tr -s " " | cut -d" " -f3`

2.7 Comodines

- Los **comodines** o **metacaracteres** son caracteres que se utilizan en lugar de otros caracteres que el sistema rellena. Los comodines más frecuentes son:
 - * → cualquier número de caracteres o ninguno.
 - ? → un solo carácter.
 - [] → un solo carácter de los indicados.
 - Ejemplos:
 - `ls file*` →
`file`
`file1`
`file55.sh`
`file.old`
 - `ls file?` →
`file0`
`file1`
 - `ls [a-z]*` → lista los ficheros cuyo nombre empiece con letra minúscula.

3. Manejo de ficheros y directorios

3. Manejo de ficheros y directorios

3.1 Gestión de directorios

3.2 Gestión de ficheros

3.3 Pagar y visualizar ficheros

3.4 Edición de ficheros

3.5 Búsqueda de ficheros

3.6 Enlaces

3.7 Compactado de ficheros y directorios

3. Manejo de ficheros y directorios

- **Nombre** de un fichero o directorio:
 - El nombre puede tener entre 1 y 255 caracteres.
 - Se puede utilizar cualquier carácter excepto la barra inclinada /.
 - No es recomendable emplear los caracteres con significado especial (= \ ^ ~ ' " ` * ; - ? [] () ! & ~ < >). Para usar ficheros con estos caracteres o espacios hay que introducir el nombre del fichero entre comillas.
 - Las letras mayúsculas y minúsculas se consideran diferentes.
- **Tipos** de ficheros:
 - **Ficheros ordinarios** (-):
 - De texto, imágenes, sonido, ejecutables, etc.
 - **Directorios** (d):
 - Archivos especiales que agrupan otros ficheros de una forma estructurada.
 - **Archivos especiales o de dispositivos** (b, c):
 - Representan los dispositivos conectados a un ordenador (impresoras, discos, terminal, etc.). Su tratamiento es similar a un archivo ordinario.
 - **Enlaces simbólicos** (l):
 - Ficheros que contienen la ruta de acceso a otros ficheros o directorios.
 - **Tuberías y sockets** (p, s):
 - Utilizados para la comunicación entre procesos.

3.1 Gestión de directorios

- **pwd**
 - Print working directory.
 - Muestra por pantalla la ruta completa del directorio actual.
- **cd [directorio]**
 - Cambia el directorio de trabajo. Va al directorio que se indique, bien con una ruta absoluta, bien con una ruta relativa.
 - Sin opciones ni argumentos → Va al directorio HOME del usuario.
- **ls [opciones] [directorio]**
 - Muestra el contenido de un directorio en orden alfabético.
 - Opciones:
 - -l → listado en formato largo, mostrando más información.
 - -a → lista también los ficheros y directorios ocultos.
 - -d → lista los nombres de directorios como ficheros, en lugar de mostrar su contenido.
 - -R → listado recursivo (lista todos los subdirectorios).
- **mkdir [opciones] directorio**
 - Crea un directorio.
 - Opciones:
 - -p → permite crear los directorios padres que no existan.
- **rmdir directorio**
 - Borra un directorio, siempre y cuando esté vacío.

3.2 Gestión de ficheros

- **cp [opciones] fichero1 fichero2**
 - Copia el contenido de fichero1 en fichero2.
 - Opciones:
 - -r → copia recursivamente. Para copiar directorios.
 - -i → pide confirmación antes de sobrescribir ficheros que existan en el destino.
 - -u → no copia un fichero o directorio si en el destino tiene fecha de modificación igual o más reciente.
 - -v → muestra por pantalla los ficheros copiados.
- **rm [opciones] fichero**
 - Borra ficheros y directorios.
 - Opciones:
 - -r → indica comportamiento recursivo. Para borrar directorios.
 - -i → pide confirmación antes de eliminar.
 - -v → muestra por pantalla los ficheros eliminados.
- **mv [opciones] fichero1 fichero2**
 - Mueve ficheros y directorios dentro del sistema de ficheros.
 - Se usa también para renombrar un fichero o directorio.
 - Opciones:
 - -i → pide confirmación antes de sobrescribir ficheros que existan en el destino.
 - -u → no mueve un fichero o directorio si en el destino tiene fecha de modificación igual o más reciente.
 - -v → muestra por pantalla los ficheros movidos.
- **touch fichero**
 - Crea un fichero vacío.
 - Si el fichero existe → modifica su fecha de actualización.

3.3 Pagar y visualizar ficheros

- **cat [opciones] [fichero]**
 - Muestra el contenido de un fichero.
- **more [fichero]**
 - Muestra el contenido de un fichero página a página. Indica el porcentaje de fichero visualizado.
 - Para avanzar una línea → Enter.
 - Para avanzar una página → Barra espaciadora.
 - Para terminar antes de llegar al final → q.
 - Permanece la información en la ventana de terminal.
- **less [fichero]**
 - Muestra el contenido de un fichero página a página.
 - Para avanzar una línea → Enter.
 - Para avanzar una página → Barra espaciadora.
 - Permite subir y bajar con las flechas y RePag y AvPag o buscar patrones en el fichero, entre otras cosas.
 - Al salir (pulsando “q”), desaparece la información de la ventana de terminal.

3.4 Edición de ficheros

- En Linux existen muchos editores de texto disponibles (vi, vim, nano, Emacs, joe,...).
- El visual editor (**vi**) es el único que se encuentra en cualquier sistema Unix. Fue el primer editor de pantalla completa que existió y, aunque no es fácil de usar, es una herramienta muy potente.
- El **nano** es un editor bastante sencillo de usar.

3.5 Búsqueda de ficheros

- **find [ruta] [opciones]**
 - Busca los ficheros que cumplen las opciones especificadas. Comienza a partir de la ruta indicada y continúa por todos sus subdirectorios.
 - Si no se indica una ruta, busca a partir del directorio actual.
 - Opciones:
 - -name "*nombre*" → Se pueden usar metacaracteres (*,?,...).
 - -type *tipo* → f (fichero ordinario), d (directorio), l (enlace), etc.
 - -user *propietario* → Propietario de los ficheros.
 - -group *grupo* → Grupo de los ficheros.
 - -links *n* → Número de enlaces de los ficheros.
 - -perm *permisos* → Permisos exactos en formato octal o simbólico. Si se pone un menos (-) delante de los permisos, indica que al menos se han de tener esos permisos.
 - -maxdepth *n* → Indica cuántos niveles de subdirectorios se ha de descender. n=1 → directorio actual, n=2 → un nivel de subdirectorios.
 - Se pueden hacer acciones con los ficheros encontrados. Para hacer referencia a éstos, se usan las {}:
 - -exec *comando* \; → ejecuta el comando sin pedir confirmación.
 - -ok *comando* \; → ejecuta el comando solicitando confirmación.

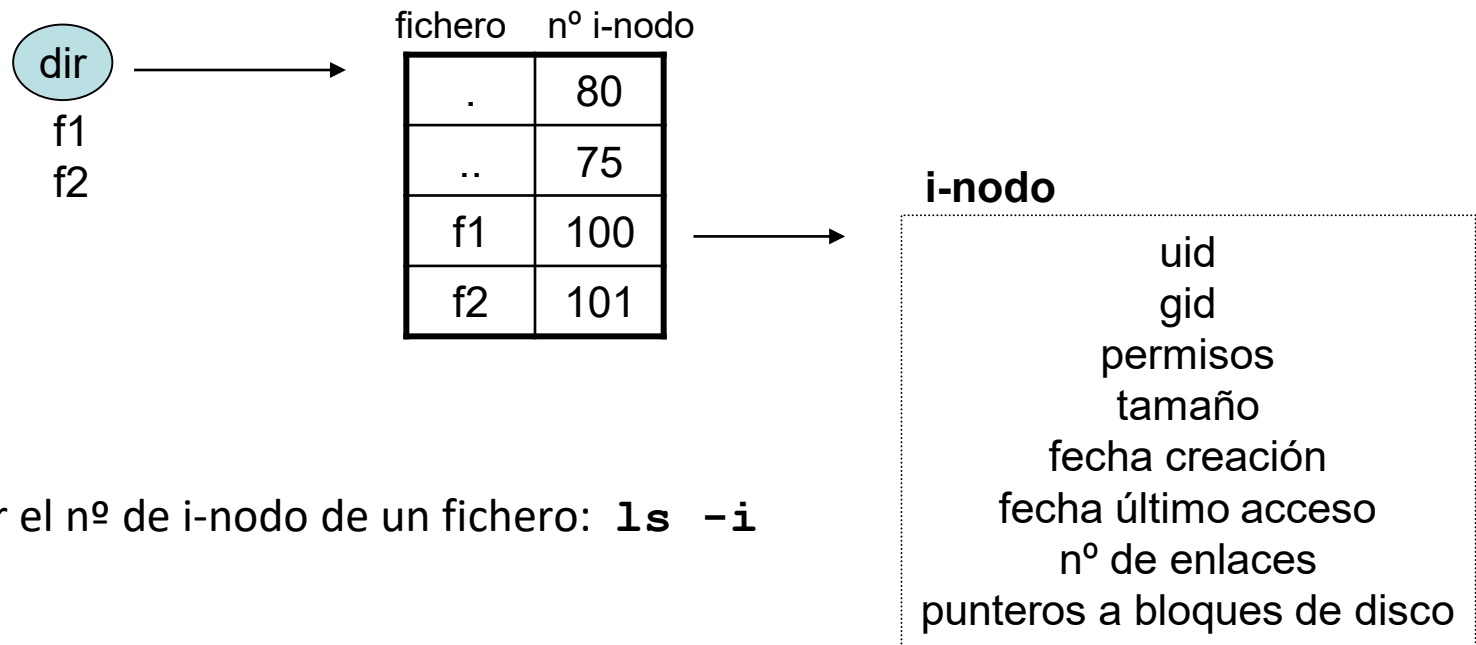
3.5 Búsqueda de ficheros

– Ejemplos:

- `find / -name "test*" →` partiendo desde el directorio raíz, muestra los ficheros (y directorios) cuyo nombre comience por *test*.
- `find / -maxdepth 2 -name "test*" →` partiendo desde el directorio raíz, muestra los ficheros (y directorios) cuyo nombre comience por *test*, mirando sólo en el directorio indicado y un nivel de subdirectorios.
- `find ~ -user alumno -type d →` partiendo desde el directorio home del usuario, muestra los directorios cuyo propietario sea *alumno*.
- `find -type f -name "*ejercicio*" →` partiendo del directorio actual, busca los ficheros ordinarios cuyo nombre contenga la palabra *ejercicio*.
- `find -type f -exec cp {} /tmp \; →` partiendo del directorio actual, busca los ficheros ordinarios y los copia en el directorio */tmp*.
- `find -perm 633 -ok rm {} \; →` partiendo del directorio actual, busca los ficheros con permisos iguales a 633 y los elimina, solicitando confirmación antes de eliminar.

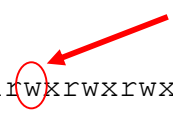
3.6 Enlaces

- En Linux, toda la información relacionada con un fichero (menos su nombre) se almacena en un **i-nodo**.
- Los directorios tienen una tabla con los nombres de los ficheros que contienen y el nº de i-nodo de cada fichero.

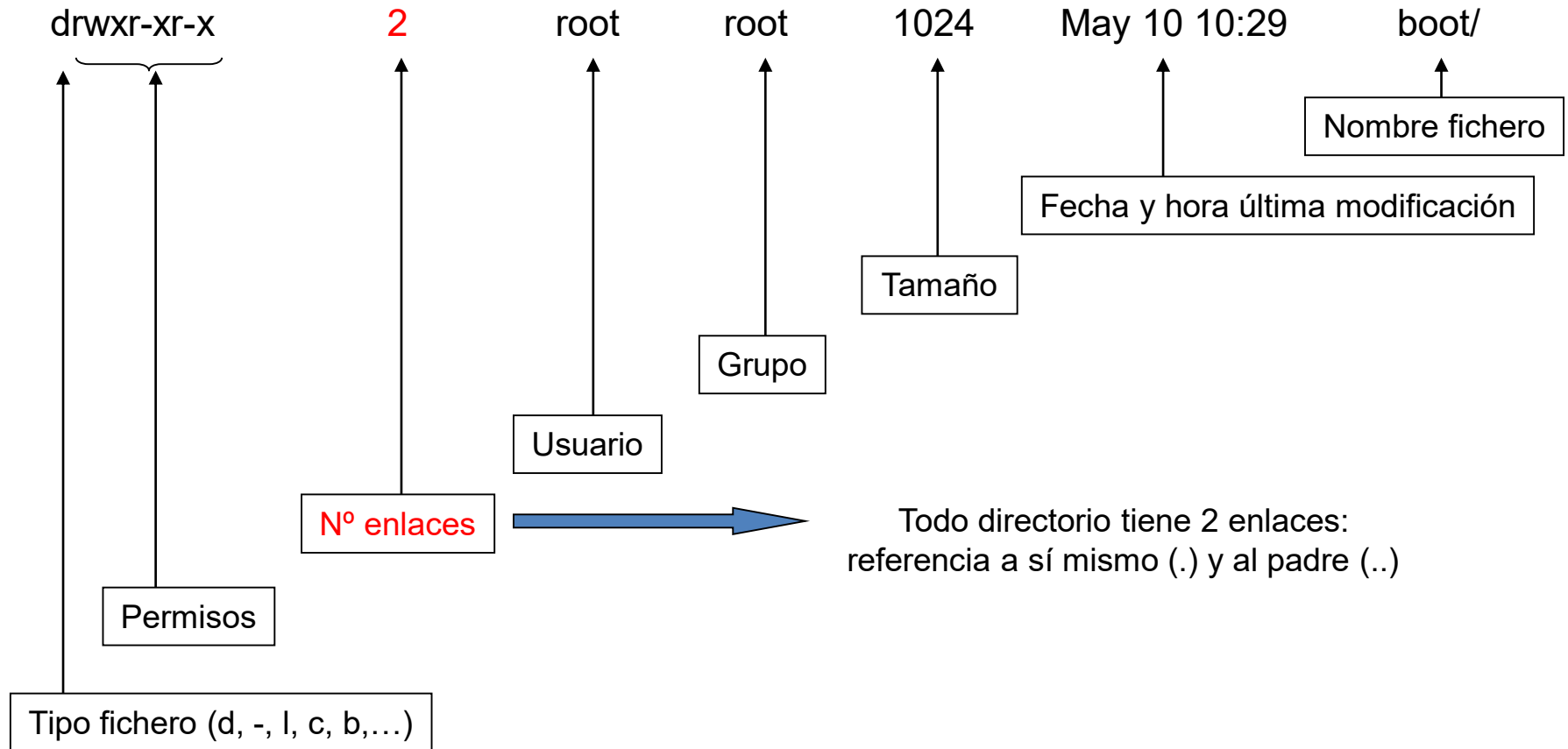


- Para ver el nº de i-nodo de un fichero: **ls -li**

3.6 Enlaces

- Un enlace es un nombre adicional para referirse a un fichero o a un directorio.
- Hay dos tipos de enlaces:
 - **Físicos o duros:**
 - Se crea aparentemente otro fichero, aunque sólo hay una copia real. Se asigna otro nombre para el mismo fichero.
 - El nº de i-nodo de los dos ficheros es el mismo.
 - Se incrementa el nº de enlaces.
 - El fichero aparece como fichero regular.
 - No se pueden crear enlaces duros de directorios.
 - Sólo se pueden crear entre ficheros del mismo sistema de ficheros.
 - **Simbólicos o blandos:**
 - Se crea un nuevo fichero que contiene un puntero al fichero original.
 - El nuevo fichero tiene un nº de i-nodo distinto.
 - No aumenta el nº de enlaces.
 - El fichero es de tipo “enlace”: `lrwxrwxrwx 1 ... link -> /tmp/test`
 - Los bits de permisos en un enlace simbólico no se usan (siempre son `rw-rw-rw-`). En su lugar, los permisos del enlace simbólico están determinados por los permisos del archivo "apuntado" por el enlace.

3.6 Enlaces

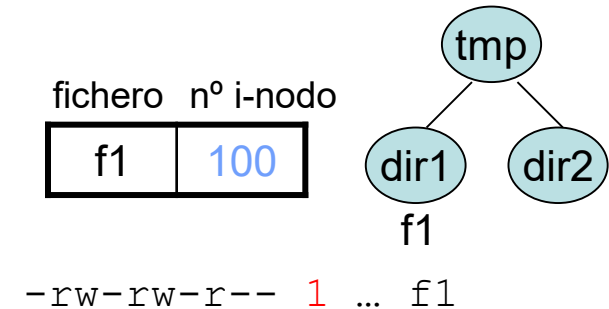


3.6 Enlaces

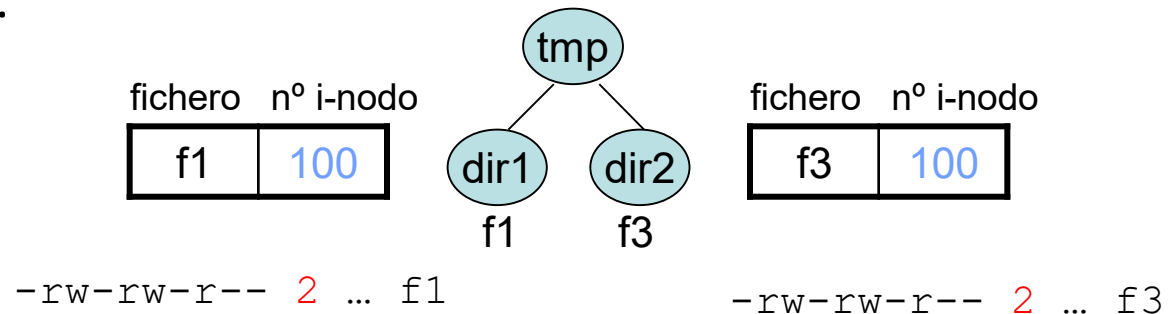
■ Cómo se crea un enlace:

□ Duro:

- `ln ruta nombre_enlace`



- `ln /tmp/dir1/f1 f3` → f3 es un enlace duro a f1. Es otro nombre para el mismo fichero (el nombre del fichero es con su ruta absoluta).

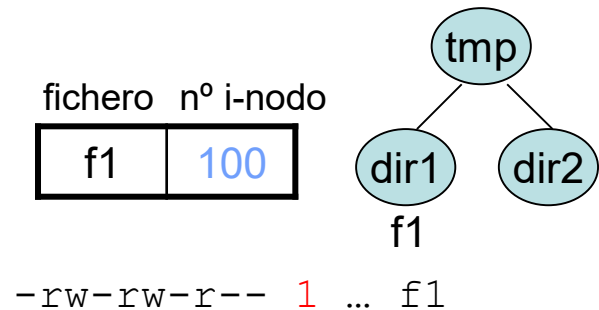


- Si se borra uno de los ficheros, se decrementa en uno el nº de enlaces pero no se borra ningún fichero.

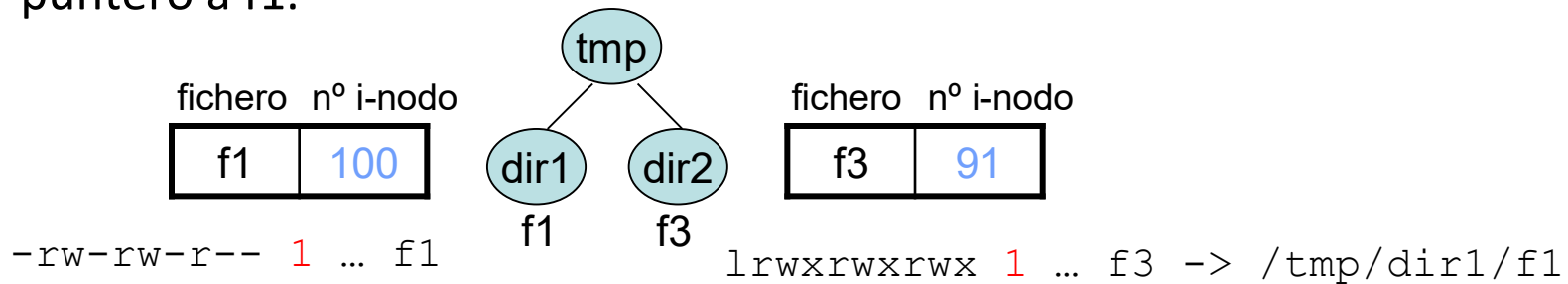
3.6 Enlaces

□ Simbólico:

■ `ln -s ruta nombre_enlace`



- `ln -s /tmp/dir1/f1 f3` → `f3` es un enlace simbólico a `f1`. Se crea un nuevo fichero `f3`, con diferente i-nodo, que contiene un puntero a `f1`.



- Si se borra el fichero `f1` o se mueve a otro sitio, el enlace se queda apuntando a algo que no existe. Cuando esto ocurre, se resalta en rojo el nombre.
- Para eliminar el enlace, se borraría el fichero `f3`.

3.6 Enlaces

- Algunos usos:
 - Enlaces duros:
 - Si un usuario no tiene permiso para acceder a un fichero porque no tiene acceso al directorio en el que se encuentra, se puede crear un enlace a este fichero en un directorio accesible para ese usuario.
 - Por compatibilidad con aplicaciones heredadas que usan directorios antiguos para copiar sus ficheros.
 - Enlaces simbólicos:
 - Son como los accesos directos de Windows.
 - Para tener en el HOME del usuario un enlace a los directorios en los que trabaja. Así es fácil recopilar todos los datos en caso de tener que cambiar de sistema.

3.7 Compactado de ficheros y directorios

- **tar [c|t|x|z|j]f fichero.tar fich1 fich2 ...**
 - Guarda varios ficheros juntos en un solo fichero (normalmente .tar).
 - Opciones:
 - -c → crea.
 - -x → extrae.
 - -z → comprime y descomprime con gzip.
 - -j → comprime y descomprime con bzip2.
 - -f → referencia a un fichero (esta opción siempre la última).
 - -t → muestra el contenido.
 - Ejemplos:
 - Empaquetar:

```
[alumno@server1 tmp]$ ls
file1 file2 file3
[alumno@server1 tmp]$ tar -cf todos.tar *
[alumno@server1 tmp]$ ls
file1 file2 file3 todos.tar
[alumno@server1 tmp]$ tar -cf sololy2.tar file1 file2
[alumno@server1 tmp]$ ls
file1 file2 file3 sololy2.tar todos.tar
```

3.7 Compactado de ficheros y directorios

- Ver contenido (sin extraer):

```
[alumno@server1 tmp]$ tar -tf todos.tar
file1
file2
file3
[alumno@server1 tmp]$ tar -tf sololy2.tar
file1
file2
```

- Desempaquetar:

```
[alumno@server1 tmp]$ ls
sololy2.tar todos.tar
[alumno@server1 tmp]$ tar -xf todos.tar
[alumno@server1 tmp]$ ls
file1 file2 file3 sololy2.tar todos.tar
```

3.7 Compactado de ficheros y directorios.

- Para archivar un directorio o archivo:

tar -cvf backup.tar /etc

Este comando crea un archivo tar llamado backup.tar que es el archivo del directorio /etc.

- Para archivar un directorio o archivo y guardarlo en un dispositivo de almacenamiento:

tar -cvf /dev/fd0 /home/user1/HGB

Este comando archivará el directorio /etc y lo guardará en el disquete.

- Para extraer el archivo:

tar -xvf backup.tar

Este comando extraerá el archivo backup.tar

- Para listar los archivos en un archivo:

tar -tvf backup.tar

El comando anterior mostrará los archivos y directorios archivados en backup.tar.

3.7 Compactado de ficheros y directorios

- Comprimir con **gzip** (**.tar.gz** – **.tar.z** – **.tgz**):

```
[alumno@server1 tmp]$ ls
file1 file2 file3
[alumno@server1 tmp]$ tar -czf todos.tgz *
[alumno@server1 tmp]$ tar -czf sololy2.tar.gz file1 file2
[alumno@server1 tmp]$ ls
file1 file2 file3 sololy2.tar.gz todos.tgz
```

- Descomprimir:

```
[alumno@server1 tmp]$ ls
sololy2.tar.gz todos.tgz
[alumno@server1 tmp]$ tar -xf sololy2.tar.gz
[alumno@server1 tmp]$ ls
file1 file2 sololy2.tar.gz todos.tgz
```

- Comprimir con **bzip2** (**.tar.bz2**):

```
[alumno@server1 tmp]$ ls
file1 file2 file3
[alumno@server1 tmp]$ tar -cjf solo2y3.tar.bz2 file2 file3
[alumno@server1 tmp]$ ls
file1 file2 file3 solo2y3.tar.bz2
```

- Descomprimir:

```
[alumno@server1 tmp]$ ls
solo2y3.tar.bz2
[alumno@server1 tmp]$ tar -xf solo2y3.tar.bz2
[alumno@server1 tmp]$ ls
file2 file3 solo2y3.tar.bz2
```

3.7 Compactado de ficheros y directorios

- Hay más posibilidades de compresión. Por ejemplo:
 - Con el comando **tar**, hay opciones para comprimir y descomprimir con xz, lzip, lzma, lzop,...
 - Comandos **gzip** y **gunzip**.
 - Comandos **zip** y **unzip**.
 - Comando **lha**.

4. Expresiones regulares

4. Expresiones regulares

4.1 Utilidad y sintaxis

4.2 Comando grep

4.1 Utilidad y sintaxis

- Las expresiones regulares son patrones que podremos utilizar para buscar o sustituir cadenas de texto.
- Están formadas por caracteres combinados con operadores.
- Operadores:
 - ^ → Principio de línea.
 - \$ → Final de línea.
 - [] → Conjunto de caracteres. Ej: [abc], [aeiou],...
 - [-] → Rango. Ej: [a-z], [A-Z], [0-9],...
 - . → Cualquier carácter.
 - * → Cero o más repeticiones del carácter anterior.
 - Por tanto:
 - . * → Lo que sea y las veces que sea (incluso ninguna).
 - .. * → Lo que sea y las veces que sea (al menos un carácter).

4.2 Comando grep

- **grep [opciones] patrón [fichero]**
 - Busca el patrón en el fichero y muestra la línea completa que lo contenga.
 - Opciones:
 - -i → no distingue entre mayúsculas y minúsculas.
 - -w → obliga a que **patrón** coincida solamente con palabras completas.
 - -v → muestra las líneas que NO contienen el patrón.
 - -n → muestra el número de línea junto con las líneas de salida.
 - El patrón puede construirse mediante:
 - Palabra o frase (entre comillas).
 - Expresión regular.
 - Ejemplos:
 - **grep tele file** → muestra las líneas que contienen “tele”.
 - **cat file | grep -w tele** → muestra las líneas que contienen “tele” como palabra completa.

4.2 Comando grep

- `grep "instituto de FP" file` → muestra las líneas que contienen el patrón indicado.
- `grep ^log file` → muestra las líneas que comiencen por "log".
- `grep -w ^login file` → muestra las líneas que comiencen por "login", siendo ésta una palabra completa.
- `grep -i adios$ file` → muestra las líneas que terminen en "adios", independientemente de si está en mayúsculas o minúsculas.
- `grep ^[a-z] file` → líneas que comiencen por una letra minúscula.
- `grep s[ao]l file` → líneas que contengan la palabra *sa/* o la palabra *sol*.
- `grep ^[a-z].*[0-9]$ file` → líneas que comiencen por una letra minúscula y terminen en un número, con el nº de caracteres intermedios que sea.
- `grep -w a.. file` → líneas que contengan una palabra que comience por "a" y tenga dos caracteres más.
- `cat file1 | grep -w ejemplo | cut -d: -f5`

5. Desconexión del sistema

5. Desconexión del sistema

- **shutdown [opciones] tiempo [mensaje de aviso]**
 - Detiene, reinicia o apaga el sistema de forma segura y ordenada en el momento indicado.
 - Notifica a los usuarios de este hecho y, además, se bloquea el sistema para que nadie más pueda acceder.
 - El único argumento obligatorio es **tiempo**, en el que se indica cuándo se debe realizar la operación. Se puede especificar de dos formas:
 - Indicando la hora en este formato → *hh:mm*.
 - Indicando los minutos que se ha de esperar → *+m*. Para cerrar inmediatamente → *now*, que equivale a *+0*.
 - Si no se especifica ninguna opción, cierra el sistema y se queda en modo monousuario (root).

5. Desconexión del sistema

– Opciones:

- -h → Detiene el sistema.
- -r → Detiene el sistema y reinicia la máquina.
- -P → Detiene el sistema y apaga la máquina.
- -c → Cancela un shutdown en curso.
- Se puede indicar al final un mensaje que se enviará a todos los usuarios.

– Ejemplos:

- `shutdown -r +10 Debido a tareas de mantenimiento, el sistema se reiniciará dentro de diez minutos`

- `shutdown -h now`
- `shutdown -P 22:30`

5. Desconexión del sistema

- **halt**
 - Detiene el sistema.
 - Equivale a: **shutdown -h now**
- **reboot**
 - Reinicia la máquina.
 - Equivale a: **shutdown -r now**
- **poweroff**
 - Apaga la máquina.
 - Equivale a: **shutdown -P now**
 - **halt** detiene el sistema operativo pero no apaga la máquina, en cambio **poweroff** sí que apaga el equipo. En ordenadores antiguos, con **halt** había que apagar el equipo después desde un botón.
- Todos estos comandos se tienen que ejecutar como superusuario.