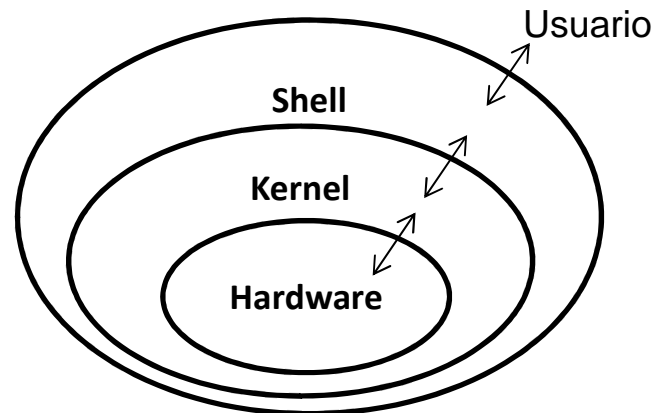


# Contenido

1. Introducción
2. Sesión
3. Ficheros
4. Organización de Ficheros
5. Comandos Básicos
6. Metacaracteres

# 1. Introducción (II)

- Unix es un Sistema Operativo:
  - **Multiusuario.**
  - **Multitarea.**
- Estructura Básica:
  - **Kernel:** Conjuntos de programas que controlan el acceso al ordenador, manejan su memoria y asigna los recursos del sistema a los usuarios, cuando así lo demandan.
  - **Shell (interfaz de usuario):** Es la parte que ve el usuario. Actúa como interprete, permitiéndole comunicarse con el S.O. Hay distintos tipos:
    - Tipo texto. Se accede a las funcionalidades del SO mediante comandos/órdenes de texto.
    - Tipo Gráfico. Basado en ventanas y acceso a opciones mediante pulsaciones de ratón.



## 2. Sesión

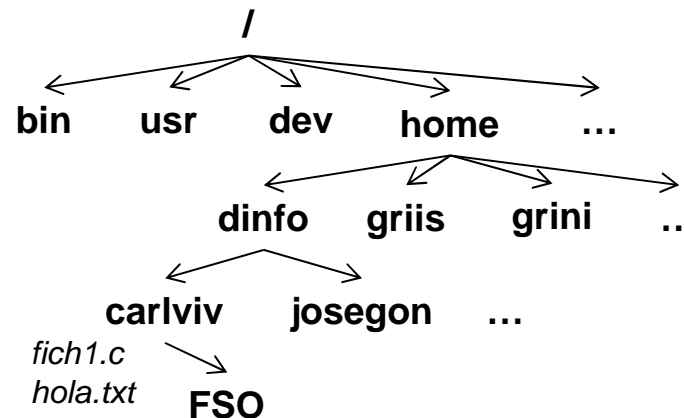
- Iniciar sesión:
  - **User Name** (nombre de usuario) + **Password** (clave)
    - Literal: Mayúsculas y minúsculas son diferentes.
- Shell tipo texto:
  - **Prompt**: Cadena de caracteres al comienzo de la línea de comandos.
    - Es configurable.
    - Por defecto, suele ser distinto en los distintos shell.
  - **Cursor**: Marca la posición en la que introducir texto.
  - Finalizar sesión: usar el comando **exit**.
  - Edición:
    - Borrado: usando las teclas de borrar habituales.
    - No siempre funcionan las flechas para movernos por la línea.
    - La tecla “tabulador” permite autocompletar nombres de comandos y nombres de ficheros y directorios.

## 3. Ficheros

- Concepto:
  - También llamado **Archivo**, es un conjunto de datos relacionados entre sí, es decir, es una selección de información homogénea, debidamente referenciada con un identificador (nombre) único.
- Nombre:
  - Pueden contener un número variable de caracteres.
  - Los caracteres pueden ser cualquiera menos los siguientes que tienen un significado especial:
    - \*, ?, [, ], ', ", \, ^, |, cualquier carácter ASCII inferior al 32 y superior al 127.
  - No existe regla, pero si convenciones normalmente aceptadas.
    - Ej., ".c" para ficheros fuente en lenguaje C, .txt para ficheros de texto, etc.
  - Las letras Mayúsculas y minúsculas son caracteres diferentes.
    - Los nombres de los ficheros deben ser introducidos literalmente.
  - Ficheros invisibles: su nombre empieza por el carácter ".". No son visibles al listar los ficheros de un directorio con las opciones por defecto del comando "ls".

## 4. Organización de Ficheros (I)

- Una máquina puede contener cientos de miles de ficheros, por lo que para facilitar su gestión (acceso, asignación de nombre, búsquedas, etc.) es necesario establecer una organización ágil y eficiente.
- **Directorio:** es un fichero cuyo contenido son ficheros y otros directorios.
  - Los ficheros se agrupan en directorios.
  - Es igual al concepto de “Carpeta” en Windows.
  - Un directorio “contenido” dentro de otro se llama un **subdirectorio**, los cuales, obviamente, pueden contener a su vez más ficheros y subdirectorios.
  - Su identificador (nombre) sigue las mismas reglas que las indicadas para ficheros.
- **Estructura jerárquica de directorios.**
  - El hecho de que un directorio pueda contener a su vez directorios da lugar a una organización jerárquica de éstos.
  - Ejemplo:



## 4. Organización de Ficheros (II)

- Relación **padre/hijo**. Un directorio que contenga otros directorios se le denomina padre de todos los directorios contenidos en él, siendo los subdirectorios contenidos sus directorios hijos.
  - Ej. el directorio *dinfo* es hijo del directorio *home* y es padre de *carlviv* y *josegon*.
- Directorio **Raíz**. Es el directorio que está en la base de la estructura jerárquica. Es el único que no tiene padre. Su nombre es “/” y no se puede cambiar.
- Directorio de **trabajo** o **actual**. Es el directorio “abierto” o en el que se halla el usuario en un determinado momento.
  - Nombres genéricos de directorios asociados:
    - “.” → Directorio actual. Independientemente de cuál sea en un determinado momento podemos usar este nombre genérico para referirnos a él.
    - “..” → Directorio padre del directorio actual. Igual que antes, independientemente de cuál sea el directorio actual podemos usar este nombre para identificar a su padre.
- Directorio **HOME**. Es la forma genérica de referirse al directorio asignado a cada usuario de la máquina.
  - Su nombre es distinto para cada usuario. Ej. *carlviv*, *juangon*, etc.
  - El usuario tiene todos los permisos para gestionar su contenido.

## 4. Organización de Ficheros (III)

- **Ruta o vía de acceso** a ficheros/directorios. Es la manera de especificar de manera única un fichero/directorio.
  - En directorios distintos puede haber ficheros/directorios con el mismo nombre, por lo tanto, sólo el nombre no es un identificador único.
  - Tipos:
    - **Ruta absoluta.** Partiendo del directorio raíz se especifican todos los directorios por lo que se “pasa” hasta “llegar” al fichero o directorio destino. Como separador se usa “/”.
      - Ej. 1. Ruta absoluta al fichero *fich1.c*: `/home/dinfo/fich1.c`
      - Ej. 2. Ruta absoluta al directorio *grini*: `/home/grini`
    - **Ruta relativa.** Igual que antes pero partiendo del directorio actual.
      - Ej. Supongamos que el directorio actual es el directorio *home*, las rutas de los ejemplos anteriores quedarán ahora:
        - » Fichero *fich1.c*: `dinfo/fich1.c`
        - » Directorio *grini*: `grini`
      - Si usamos el nombre genérico “.”, nos estamos refiriendo el directorio actual o sea al */home*. Podemos usar esto en la ruta. Por ej., las rutas anteriores también podrían haber sido especificadas:
        - » Fichero *fich1.c*: `./dinfo/fich1.c`
        - » Directorio *grini*: `./grini`
      - En principio la ruta se especifica desde el directorio actual hacia “abajo” en la estructura de directorios. Sin embargo, usando el nombre genérico “..” podemos “ir hacia arriba”. Por ej., supongamos que ahora el directorio actual es *josegon*. Podemos referenciar los directorios (se haría igual para ficheros) *dinfo* y *grini* mediante rutas relativas de la siguiente manera:
        - » `../dinfo`
        - » `../../grini` (se pueden usar tanto “..” como se quiera en una ruta)

## 5. Comando Básicos (I)

- Sintaxis básica: `nombre_comando argumentos_de_la_orden`
  - Con **argumento de la orden** cualquier cadena de caracteres a la derecha del comando. No todos los comandos necesitan argumentos (`comando [args]`).
- Para directorios:
  - Comando **cd**. Cambia el directorio actual. Sin argumentos cambia al HOME.
    - Sintaxis básica: `cd directorio` ó `cd (cd [directorio])`
    - Ej.: `cd /home/dinfo`    `cd ../home/griis`    `cd FSO`
  - Comando **ls**. Muestra el contenido de un directorio. Sin argumentos: el actual.
    - Sintaxis básica: `ls directorio` ó `ls (ls [directorio])`
    - Ej.: `ls /home/dinfo`    `ls ../home/griis`    `ls FSO`
    - Opción interesante, listado largo: `ls -l directorio` (ó `ll directorio`). Muestra:
      - `dir/fich (d/-) permisos enlaces propietario grupo tamaño fecha_ultima_modificación nombre`
  - Comando **mkdir**. Crea nuevos directorios
    - Sintaxis básica: `mkdir directorio`
    - Ej. (dir. actual carlviv): `mkdir /home/dinfo/carlviv/nuevo_dir`    `mkdir tmp`
  - Comando **rmdir**. Borra directorios. Deben estar vacíos.
    - Sintaxis básica: `rmdir directorio`
    - Ej. (dir. actual carlviv): `rmdir /home/dinfo/carlviv/nuevo_dir`    `rmdir tmp`
  - Comando **pwd**. Muestra la ruta absoluta al directorio actual.



## 5. Comando Básicos (II)

- Para ficheros:
  - Comando **cat**. Muestra el contenido de un fichero. Sólo válido para ficheros de texto (ficheros codificados en ASCII).
    - Sintaxis básica: `cat fichero`
    - Ej.: `cat /home/dinfo/carlviv/fich1.c`      `cat hola.txt`
  - Comando **rm**. Elimina un fichero. **¡¡IMPORTANTE!!**, no hay manera de recuperarlo.
    - Sintaxis básica: `rm fichero`
    - Ej.: `rm /home/dinfo/carlviv/fich1.c`      `rm hola.txt`
  - Comando **cp**. Copia un fichero
    - Sintaxis básica.
      - Opción 1. Copia de un directorio a otro sin cambiar nombre: `cp ruta_origen/fich ruta_destino/`
      - Opción 2. Copia de un directorio a otro cambiando nombre: `cp ruta_origen/fich ruta_destino/nuevo_nombre`
      - Opción 3. Copia al mismo directorio con otro nombre: `cp ruta_origen/fich ruta_origen/nuevo_nombre`
    - Ej.
      - Opción 1: `cp fich1.c ../dinfo/home/grini/marfer/`
      - Opción 2: `cp fich1.c /home/grini/marfer/fich2.c`
      - Opción 3: `cp fich1.c fich2.c`
  - Comando **cmp**. Compara dos ficheros. Si no muestra nada es que son iguales.
    - Sintaxis básica: `cmp fich1 fich2`

## 5. Comando Básicos (III)

- Para ficheros y directorios (I):
  - Comando **mv**. Cambia de directorio y/o de nombre a un fichero/directorio. El original es eliminado.
    - Sintaxis básica.
      - Opción 1. Cambio de directorio: `mv ruta_origen/fich_dir ruta_destino/`
      - Opción 2. Cambio de directorio y nombre: `mv ruta_origen/fich_dir ruta_destino/nuevo_nombre`
      - Opción 3. Cambio de nombre: `mv ruta_origen/fich_dir ruta_origen/nuevo_nombre`
    - Ej.
      - Opción 1: `mv fich1.c ../dinfo/home/grini/marfer/` (sobre fichero)
      - Opción 2: `mv FSO /home/grini/marfer/FSO_2` (sobre directorio)
      - Opción 3: `mv hola.txt saludo.txt` (sobre fichero)
  - Comando **ln**. Crea un enlace entre un fich/dir origen y otro destino. Es similar a copiar, con la diferencia de que al copiar se crea un nuevo fichero en el disco, mientras al enlazar sólo “se crea un nuevo nombre”, el fichero físico en el disco es el mismo, pero podemos acceder a él mediante 2 (o más) nombre distintos.
    - Al enlazar cualquier modificación que se realice usando cualquiera de los nombres del fichero/directorio afectará al resto de nombres, ya que el fichero/directorio es el mismo.
    - Sintaxis básica: igual que para cp y mv. Ej.
      - Opción 1: `ln fich1.c ../dinfo/home/grini/marfer/` (enlace sobre ficheros. Mismo nombre)
      - Opción 2: `ln FSO /home/grini/marfer/FSO_2` (enlace sobre directorios. Distinto nombre)
      - Opción 3: `ln fich1.c fich2.c` (enlace sobre ficheros. Mismo directorio)

## 5. Comando Básicos (IV)

- Para ficheros y directorios (II):
  - Comando **chmod**. Cambia los permisos de acceso.
    - Permisos en UNIX. Sobre cada fichero/directorio se definen tres tipos de usuarios: propietario (u), grupo (g) y otros (o). Cada uno de ellos tiene tres tipos de permisos, que sobre un fichero tienen los siguientes significados:
      - Escritura (w). Si está activado (+) el fichero se puede modificar y/o borrar.
      - Lectura (r). Si está activado (+) se puede ver el contenido del fichero.
      - Ejecución (x). Si está activado (+) el fichero puede ser ejecutado. Sólo tiene sentido sobre ejecutables
    - Sintaxis básica: `chmod tipo_usuario±tipo_permiso fich/dir`
      - Si se pone “+” el permiso se da. Si se pone “-” el permiso se quita.
      - En *tipo\_usuario* se puede usar la opción “a” (*all*), que indica que el permiso se va a dar o quitar a los tres tipos de usuarios.
    - Ej.
      - Un solo permiso y usuario: `chmod u+x fich_dir`      `chmod g-w fich_dir`      `chmod o-r fich_dir`
      - Varios permisos y/o usuarios:
        - » `chmod a-w fich_dir`    `chmod ug+rw fich_dir`    `chmod go-x fich_dir`    `chmod u+w,go-w,o-xr fich_dir`
  - Modo octal. Es una manera más cómoda de indicar los permisos cuando se van a modificar varios a la vez.
    - Formamos la siguiente palabra con los permisos: `rw-rw-rw-`, donde los tres primeros se corresponden al propietario, los segundos al grupo y los terceros a otros. Ahora sustituimos cada letra por un dígito binario con el siguiente significado: 1⇒permiso dado y 0 ⇒permiso quitado. Se codifica en octal la palabra binaria resultante, siendo esto lo especificado en el segundo **argumento** de `chmod`: `chmod modo_octal fich/dir`.
    - Ej. `chmod 755 fich_dir` ⇒ el propietario tiene todos los permisos (7→111), el grupo y otros sólo los de lectura y ejecución (5→101).

## 5. Comando Básicos (IV)

- Comandos útiles:
  - Comando **man**. Muestra ayuda acerca de comandos. También muestra ayuda sobre funciones del lenguaje C.
    - Sintaxis básica: `man comando`
  - Comando **passwd**. Permite cambiar el password (clave) del usuario.
  - Comando **exit**. Cierra una sesión de trabajo.
  - Comando **who**. Muestra información acerca de quién está conectado en un determinado momento a la máquina e información de sus sesiones de trabajo
    - Si se escribe **who am i** se muestra nuestro nombre de usuario e información de las sesiones de trabajo que tengamos abiertas.
  - Comando **date**. Muestra información de la fecha del sistema.
  - Cambiar de shell: basta con poner el nombre del shell (interprete de comandos) que queramos abrir. Algunos tipos:
    - Bourne shell (`sh`).
    - Bourne-Again shell (`bash`) . Es el ejecutado por defecto en la máquina del laboratorio.
    - Korn shell (`ksh`).
    - C shell (`csh`).
    - Z shell (`zsh`).

## 5. Comando Básicos (IV)

- Opciones (modificadores) de comandos.
  - Los comandos vistos hasta ahora les hemos mostrado con su sintaxis básica (funcionamiento por defecto).
  - Casi todos los comandos Unix tienen una serie de opciones de ejecución que modifican ese funcionamiento por defecto o añaden funcionalidades extra.
    - El entorno de comandos UNIX es muy potente.
  - Sintaxis general de comandos UNIX:  
*comando [argumentos\_tipo\_opción] [argumentos\_de\_la\_orden]*
    - Los argumentos de tipo opción o modificadores, que indican las opciones con que se ejecuta ese comando, se introducen entre el nombre del comando y los argumentos.
    - La sintaxis general de los **modificadores** es: *-letra*.
      - Cada opción de ejecución lleva asociada una letra.
      - Se pueden introducir más de un modificador. El número de estos depende de cada comando.
  - Ejemplos:
    - `rm -f fichero` → Fuerza el borrado del fichero, aunque esté protegido contra escritura.
    - `rm -rf directorio` → Borra, sin preguntar, recursivamente todo el *directorio*.
    - `ls -lt` → Muestra un listado en formato largo ordenado por fecha de modificación.
    - `cat -n fichero` → Añade el número de línea al mostrar el fichero.
  - Con el comando **man** podemos ver todas las opciones que tiene cada comando.

## 6. Metacaracteres

- También llamados **caracteres comodín**, sirven para especificar en una única expresión un conjunto de ficheros o directorios.
- Tipos:
  - “?”. En una expresión se sustituye por cualquier carácter. Puede aparecer en cualquier punto de la expresión y si límite en su número.
    - Ejemplos.
      - *p?* → cualquier nombre que empiece por p y continúe por cualquier letra. Ej. *pe*, *pa*, *pr*, etc.
      - *??a???* → cualquier nombre con 6 letras, siendo la tercera una “a”. Ej., *inacio*, *jramos*, etc.
      - *cat hola.?* → mostrará el contenido de todos los ficheros cuyo nombre tenga la cadena “hola.” seguida de cualquier letra, como por ejemplo, *hola.c*, *hola.p*, *hola.t*, etc.
    - “\*”. Se sustituye en una expresión por cualquier cadena de caracteres, incluida la vacía. Puede aparecer en cualquier punto de la expresión y sin límite de número.
      - Ejemplos.
        - *p\** → cualquier nombre que empiece por *p* seguida de cualquier cadena de caracteres, como por ejemplo, *pepe*, *programa.c*, *prueba.txt*, etc.
        - *\*a\*.c* → cualquier nombre que contenga una *a* y acabe en *.c*. Ej., *programa.c*, *practica.c*, etc.
        - *cp \*.java ../* → copiamos todos los ficheros que acaben en *.java* al directorio padre del actual.
    - Se pueden usar ambos metacaracteres en una misma expresión.
      - Ejemplos.
        - *ls p\*.?* → Lista todos los ficheros de directorio actual que empiecen por *p* y acaben con un “.” más cualquier carácter.
        - *rm \*hola??* → Borra ficheros que empiecen por cualquier cosa, luego tengan la cadena *hola* seguida por dos caracteres más cualquiera.