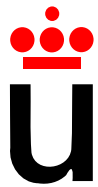
# Conceptos básicos de GNU/Linux

## Sistemas Operativos

Enrique Soriano, Gorka Guardiola

GSYC

1 de octubre de 2025

Universidad
Rey Juan Carlos

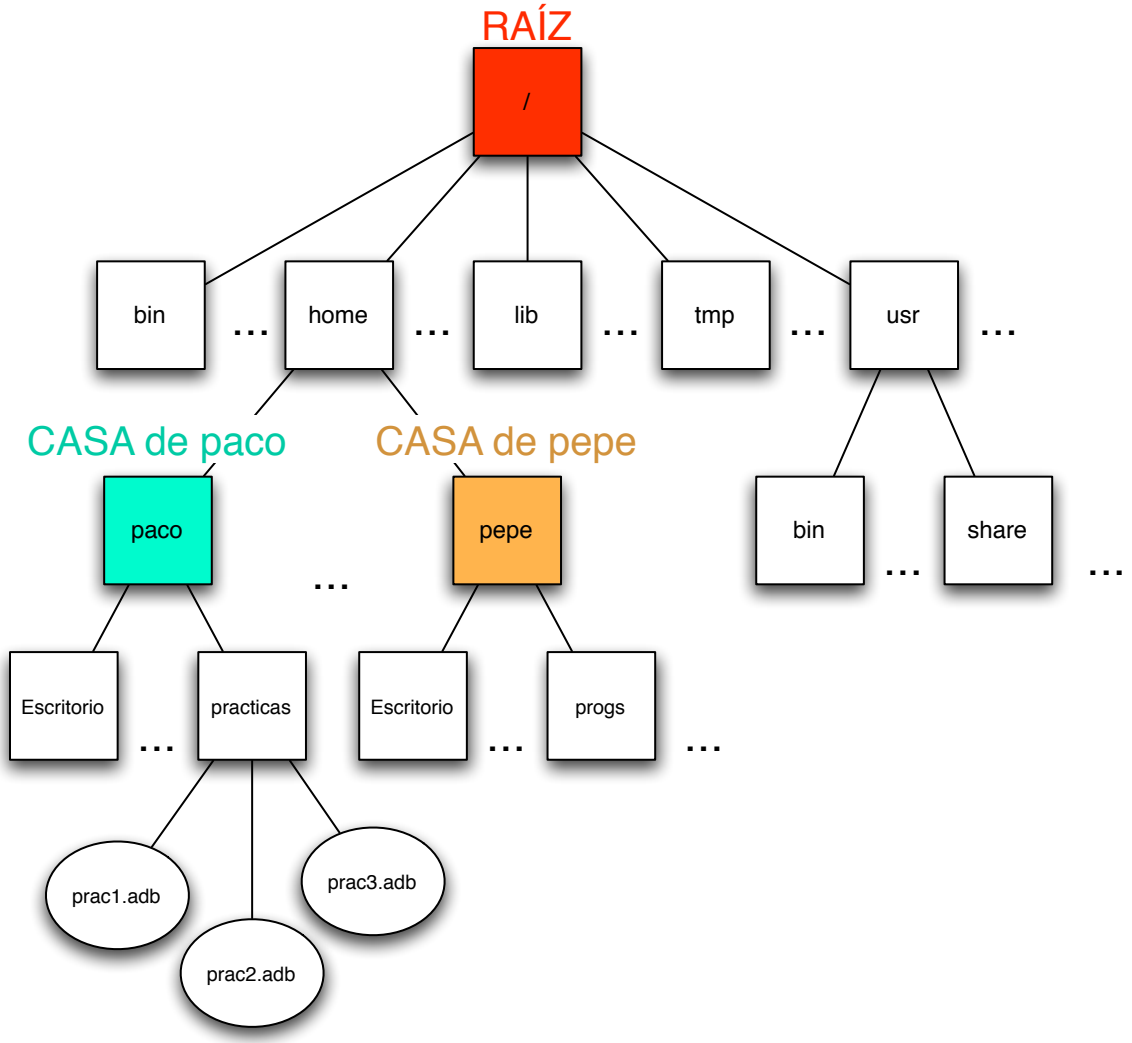
# Definiciones

- ***Comando* o mandato** (command): cadena de texto que identifica a un programa u orden.
- **Shell**: programa que te deja ejecutar *comandos*. Por lo general, permite crear programas (scripts) en un lenguaje propio. Hay distintos tipos de shells, usaremos `bash`. Básicamente, un shell hace esto:
  1. leer un línea de comandos
  2. sustituir algunas cosas en esa línea
  3. crear los procesos para ejecutar los comandos descritos por la línea
- **Prompt**: texto que indica que el **shell** está esperando una orden.
- **Usuario** (login name): el nombre de usuario, todos los programas ejecutan en nombre de un usuario.
- **root**: superusuario o administrador del sistema.

# Ficheros y directorios

- Organizados en **árbol** $\rightarrow$ directorio **raíz** (root).
- Dos ficheros que están en distintos directorios son dos ficheros diferentes.
- **Directorio de trabajo**, pwd.
- **Directorio casa**, $HOME

# Árbol de ficheros



# Directorios en GNU/Linux

- /bin tiene ejecutables.
- /dev tiene dispositivos.
- /etc tiene ficheros de configuración.
- /home tiene los datos personales de los usuarios.
- /lib tiene las bibliotecas (código) que usan los programas ejecutables.
- /proc y /sys ofrecen una interfaz para interaccionar con el núcleo del sistema.
- /sbin tiene los ejecutables del sistema.
- /tmp sirve para almacenar los ficheros temporales, se borra en cada reinicio.
- /usr existe por razones históricas (tamaño de almacenamiento) y contiene gran parte del sistema: contiene directorios similares a los anteriores (/usr/bin o /usr/lib), con los datos y recursos para los programas de usuario (no del sistema).
- /var tiene los datos que se generan en tiempo de ejecución (cache, logs y otros ficheros que generan los programas).
- /boot continen los ficheros de arranque del sistema.
- /media y /mnt puntos de montaje
- /opt contiene ficheros para programas *de terceros*.

# Rutas (*path*)

- Ruta absoluta: serie de directorios desde el raíz separados por barras.
  - `/home/alumnos/pepe/fichero.txt`
- Ruta relativa: serie de directorios desde el directorio actual.
  - `alumnos/pepe/fichero.txt`
- .. : directorio padre.
  - `../pepe/fichero.txt`
- . : directorio actual.
  - `./fich1`
- Para indicar que queremos ejecutar un fichero del directorio actual:
  - `./miprograma`

# Texto plano

- ASCII: usa 1 byte para representar 128 caracteres (7 bits + 1 bit).
- ISO-Latin 1 (8859-1): usa 1 byte para almacenar caracteres (8 bits).
- UTF-8: puede usar 1,2, o más bytes. Compatible hacia atrás.
- Hay muchas otras.

# Ficheros de texto plano

Caracteres de control:

- El carácter `'\n'` indica nueva línea en el texto. Es una convención usada en todos los programas, bibliotecas, etc. en Unix.
- En otros sistemas operativos no tiene por qué ser así (Windows usa la secuencia `'\n\r'`).
- El carácter `'\t'` indica un tabulador.
- No hay carácter EOF: invención de los lenguajes.

# Manual

- Las páginas de manual se pueden consultar con el comando `man`: `man sección asunto`
  Por ejemplo: `man 1 gcc`
- Secciones de interés: comandos (1), llamadas al sistema(2), llamadas a biblioteca(3).
- Para buscar sobre una palabra: `apropos`.
  Por ejemplo: `apropos gcc`.

# Comandos básicos

- `cd`: cambia de directorio actual.
- `echo`: escribe sus argumentos por su salida.
- `touch`: cambia la fecha de modificación de un fichero. Si no existe el fichero, se crea.
- `ls`: lista el contenido de un directorio.
- `cp`: copia ficheros.
- `mv`: mueve ficheros.
- `rm`: borra ficheros.
- `mkdir`: crea directorios.
- `rmdir`: borra directorios vacíos.
- `date`: muestra la fecha.

# Comandos básicos

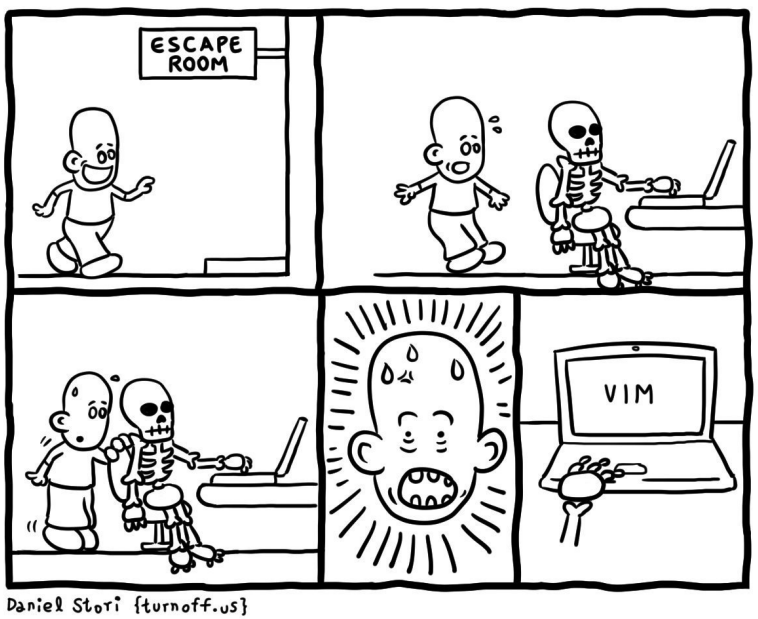
- `who`: muestra los usuarios que están en el sistema.
- `whoami`: muestra tu nombre de usuario.
- `sort`: ordena las líneas de un fichero.
- `wc`: cuenta caracteres, palabras y líneas de ficheros.
- `fgrep, grep`: buscan cadenas dentro de ficheros.
- `cmp, diff`: comparan ficheros.
- `cat`: escribe en su salida el contenido de uno o varios ficheros.
- `less`: permite leer un fichero de texto en el terminal usando *scroll*.

# Comandos básicos

- `file`: da pistas sobre el contenido de un fichero.
- `od`: escribe en su salida el los datos de un fichero en distintos formatos.
- `head, tail`: escriben el las primeras/últimas líneas del fichero en su salida.
- `tar`: crea un fichero con múltiples ficheros dentro (comprimidos o no).
- `gzip/gunzip`: comprime/descomprime un fichero.
- `top`: muestra los procesos y el estado de sistema.
- `reset`: restablece el estado del terminal.
- `exit`: el shell termina su ejecución.

# Editor en modo texto: vi

- Hay múltiples editores para usar en el terminal (nano, pico, vim, emacs, etc.). Es necesario saber cómo editar texto en un terminal.
- **vi** es el editor clásico.
- **vim** es un editor basado en **vi**. Implementa un superconjunto. En ciertas distribuciones, `vi` es en realidad `vim`.
- Tiene fama de ser complicado (sobre todo salir de él) :)



# Editor en modo texto: vi

Hay múltiples editores para usar en el terminal (nano, pico, vim, emacs, etc.), pero **vi** es el editor clásico. Comandos para sobrevivir:

- Tiene dos modos: modo inserción (para escribir) y modo comando
  - **i** pasa a modo inserción
  - **ESC** pasa a modo comando
- En modo comando
  - **:q!** sale del editor sin guardar
  - **:x** salva el fichero y sale (también se puede con **:wq**)
  - **:w** salva el fichero
  - **:número** se mueve a esa línea del fichero
  - **i** inserta antes del cursor
  - **a** inserta después del cursor
  - **o** inserta en una línea nueva
  - **dd** corta una línea
  - **p** pega la línea cortada anteriormente
  - **h, j, k, l** mueve el cursor a izquierda, abajo, arriba, derecha

# Variables

- Variable de shell: son locales a la shell, los programas ejecutados por el shell no tienen dichas variables.
- Variable de entorno: los programas ejecutados por el shell sí tienen su propia copia de la variable, con el mismo valor.
- `mivar=hola` define la variable de shell con nombre *mivar*, cuyo valor será *hola*.
- `$mivar` el shell sustituye eso por el valor de dicha variable (si no existe, lo sustituye por nada).
- `export mivar` exporta la variable (ahora es una variable de entorno).

# Variables

- El comando `set` muestra todas las variables (de shell y de entorno).
- El comando `printenv` muestra las variables de entorno (también lo hace el comando `env`).
- El comado `unset` elimina una variable.
- Variables populares:
  - $PATH: la ruta de los programas.
  - $HOME: la ruta de tu directorio casa.
  - $USER: el nombre de usuario
  - $PWD: la ruta actual del shell
  - $LANG: configuración de localización (*locales*).
  - $LC_xxx: otras variables de localización (*locales*).

# Usando el terminal

Globbing (wildcards): caracteres especiales para el shell que sirven para hacer referencia a nombres de ficheros:

- **?** cualquier carácter.
- ***** cualquier secuencia de caracteres.
- **[ab]** cualquiera de los caracteres que están dentro de los corchetes (letra a o la letra b en el ejemplo).
- **[b-z]** cualquier carácter que se encuentre entre esas dos (de la letra b a la z en el ejemplo).

Para escribir caracteres especiales sin que haya sustitución:

- ' ' las comillas simples *escapan* todo lo que tienen dentro (ya no tienen un significado especial).
- " " las comillas dobles *escapan* todo menos algunas sustituciones (p. ej. las variables de entorno).

# Usando el terminal

- ↑ repite los comandos ejecutados anteriormente en la shell.
- **Tab** El tabulador completa nombres de ficheros.
- Ctrl+r deja buscar comandos que ejecutamos hace tiempo.
- Ctrl+c mata el programa que se está ejecutando.
- Ctrl+z detiene el programa que se está ejecutando.
- Ctrl+d termina la entrada (o manda lo pendiente).
- Ctrl+a mueve el cursor al principio de la línea.
- Ctrl+e mueve el cursor al final de la línea.
- Ctrl+w borra la palabra anterior en la línea.
- Ctrl+u borra desde el cursor hasta el principio de la línea.
- Ctrl+k borra desde el cursor hasta el final de la línea.
- Ctrl+s congela el terminal. Ctrl+q lo descongela. En muchas configuraciones, Ctrl+s ya no hace esto, pero hay que tenerlo en mente por si sucede.
- Ctrl+l: limpia el terminal.