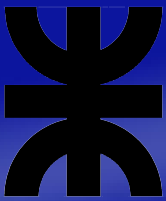


# Línea de comando

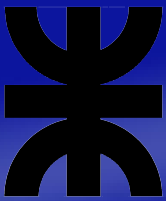


# Línea de comando

- Linux dispone de un intérprete de órdenes (terminal o Shell).
- El terminal hace de interfaz entre el usuario y el propio sistema operativo.
- Es una forma de acceder al sistema sin usar interfaz gráfica.

Para acceder a una terminal se puede hacer de dos formas:

- ♦ Una es con una aplicación como el terminal de GNOME, xterm o konsole de KDE, que son emuladores de la terminal dentro de una interfaz visual. Podemos acceder a la aplicación terminal (en ubuntu) desde Aplicaciones → Accesorios → Terminal
- ♦ Otra forma es salirse del entorno gráfico y acceder a un entorno completamente en modo texto. Para esto, debemos teclear Control+Alt+F1. Linux proporciona por defecto seis terminales de este tipo, de Control+Alt+F1 a Control+Alt+F6. Si queremos volver al modo gráfico lo hacemos con Alt+F7.



# Línea de comando

El terminal muestra un indicador (#, \$) de línea de ordenes esperando a que se introduzca un comando

Al abrir un terminal nos encontraremos en la ruta **/home/usuario**

**usuario@nombre\_pc: ~\$** -> usuario normal

**usuario@nombre\_pc: ~#** -> super usuario (administrador)

~ (vírgula) significa que estamos en la ruta **/home/usuario**

Si salimos de ahí veremos:

**usuario@nombre\_pc: ruta absoluta\$**



# Línea de comando

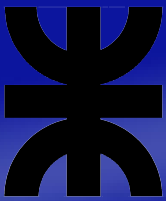
**man <command> (manual)**

Nos muestra el manual del comando

```
usuario@mipc:~$ man -printf
```

Busca la palabra clave printf entre las descripciones breves

Para salir del manual presionamos Q o Ctrl + F2



# Línea de comando

## **ls <opciones> <ruta> (list)**

Muestra archivos en una carpeta

ls (sin parámetros) lista archivos y directorios (en distintos colores) del directorio actual.

ls **-a** muestra todos los archivos y directorios incluyendo los ocultos

ls **-s** muestra el tamaño de cada fichero listado

ls **-l** muestra permisos, números de enlaces rígidos, nombre del propietario, grupo al que pertenece, tamaño, fecha de la última modificación

ls **-t** ordena los archivos por fecha de modificación, el más nuevo primero

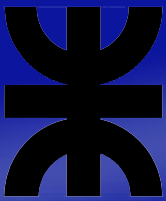
ls **-S** ordena el listado según el tamaño de los archivos

```
usuario@mipc:~$ ls -al
```

También podemos usar rutas relativas o absolutas

```
usuario@mipc:~$ ls -l /home/usuario/car1/car3
```

El archivo `.` hace referencia a sí mismo y el `..` al directorio padre



# Línea de comando

**cd** <opciones> <ruta> (change directory)

Cambiar el directorio de trabajo (change directory)

**cd** . directorio actual (. es del mismo directorio)

**cd** (sin parámetros) lleva al HOME de tu usuario

**cd** ~ lleva al home de tu usuario.

**cd** .. sube un directorio, directorio padre

**cd** / lleva al directorio raíz

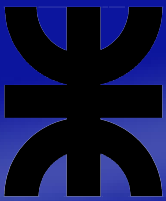
**cd** - lleva al último directorio en que hayas estado

usuario@mipc:~\$ cd

usuario@mipc:~\$ cd ~

usuario@mipc:~\$ cd ..

usuario@mipc:~\$ cd /



# Línea de comando

**mkdir <opciones> <nombre del directorio>** (make directory)

Crear un directorio

**-m** modo permisos en octal (usuario/grupo/otros usuarios)

**-p** crea el directorio padre y los subdirectorios si no existen, si existe solo crea el sub directorio

Ejemplos:

Crear el directorio car 6 en /home/usuario

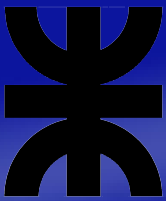
usuario@mipc:~\$ mkdir car6

Crear el directorio car 6 en /home/usuario

usuario@mipc:~\$ mkdir /home/usuario/car6

Crear el directorio car 6 en /home/usuario con permisos de lectura y escritura para el usuario y para el resto ninguno

usuario@mipc:~\$ mkdir -m600 car6



# Línea de comando

**rm <file>** (remove)

Borrar directorios y archivos.

Archivos:

rm sin parámetros borra archivo sin pedir confirmación

rm **-f** borra el archivo sin pedir confirmación, e ignora los archivos inexistentes (no muestra mensaje de error)

rm **-i** pide confirmación al borrar el archivo

Directorios:

rm **-r** <directorio> : borra el directorio y todo su contenido.

rm **-R** <directorio> : borra el directorio y todo su contenido en forma recursiva

rmdir <directorio> : borra el directorio si está vacío

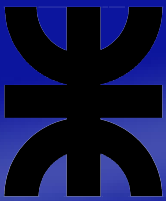
```
usuario@mipc:~$~/car1$ rm arc1.txt
```

```
usuario@mipc:~$~/car1$ rm arc1.txt
```

```
usuario@mipc:~$~/car1$ rm -f arc1.txt
```

```
usuario@mipc:~$~/car1$
```





# Línea de comando

**cp <opciones> <fichero origen> <destino> (copy)**

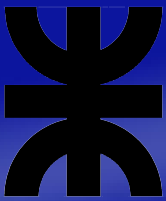
Copiar un archivo o directorio en el directorio especificado

- i pregunta antes de sobrescribir un archivo
- n no sobrescribe archivos existentes
- f si el archivo de destino ya existe y no puede ser leído lo borra y lo intenta copiar de nuevo
- p mantiene los permisos y los propietarios de los archivos copiados
- u copia sólo cuando el archivo de origen es más reciente que el archivo destino, o cuando el archivo no existe.
- R copia el directorio y todo su contenido

```
usuario@mipc:~$ cp arc4.txt ../../car2
```

```
usuario@mipc:~$ cp arc4.txt /home/usuario/car2
```

```
usuario@mipc:~$ cp -i arc4.txt ../../car1
```



# Línea de comando

**mv <opciones> <origen> <destino> (move)**

Mover o renombrar archivos o directorios. El archivo es borrado y creado en la otra ruta, la ruta destino debe existir.

- f si el archivo destino existe, no pregunta y sobrescribe el archivo
- i si el archivo destino existe, pregunta si quiere sobrescribe el archivo
- u mover sólo cuando el archivo de origen es más reciente que el archivo destino, o cuando el archivo no existe.

Renombrar el archivo arch1.txt a arch2.txt

```
usuario@mipc:~$ mv arch1.txt arch2.txt
```

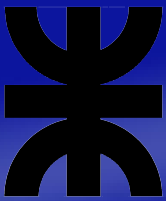
Mover el archivo arc4.txt a la carpeta car1

```
usuario@mipc:~$ mv arc4.txt ../../car1
```

```
usuario@mipc:~$ mv car4/*../car2
```

```
usuario@mipc:~$ mv car4 ../car2
```

```
usuario@mipc:~$ mv *.txt ../car2
```



# Línea de comando

## find, locate

Buscar un archivo dentro del sistema.

**-name** busca por nombre y distingue entre mayúsculas y minúsculas

**-iname** busca por nombre y no distingue entre mayúsculas y minúsculas

```
usuario@mipc:~$ find *.txt
```

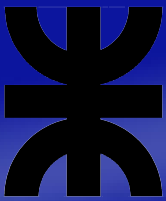
Buscar todos los archivos XXX1.txt dentro de la carpeta car1

```
usuario@mipc:~$ find ???1.txt
```

```
usuario@mipc:~$ find car1/*.txt
```

```
usuario@mipc:~$ find car1/ -size +50k
```

```
usuario@mipc:~$ find -name arc1.txt
```



# Línea de comando

**ps <opciones>** (process status)

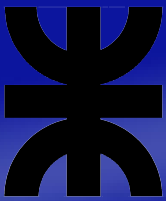
Mostrar el estado de los procesos

Muestra qué procesos están corriendo en nuestro sistema. Cada proceso está identificado con un número llamado **PID** (process ID).

- A** o **e** Lista los procesos de todos los usuarios
- u** Lista información de los procesos del propio usuario
- x** Lista procesos de todas las terminales y usuarios
- a** Lista los procesos de los usuarios

Para ver las opciones de ps, usamos `ps - - help`

```
usuario@mipc:~$ ps aux
```



# Línea de comando

Descripción de columnas de ps:

**User:** muestra de quien es el proceso

**PID:** Identificación del número de proceso.

**% Cpu:** porcentaje de la Cpu que está tomando este proceso

**% Mem:** porcentaje de memoria que está tomando el proceso

**VSZ:** cantidad de memoria virtual que está tomando el proceso

**RSS:** cantidad de memoria residente que está tomando el proceso

**Stat:** Estado del proceso y pueden ser:

**s:** durmiendo

**R:** proceso alojado en la CPU, ejecutándose

**D:** Ininterrumpible de dormir

**T:** El proceso tuvo un error o fue detenido

**Z:** proceso en Zombi

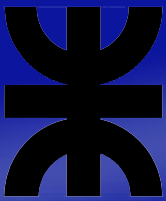
**START:** fecha en la que el proceso empezó

**TIME:** tiempo que el proceso lleva alojado en la CPU

**Command:** Nombre del proceso y sus parámetros de la línea de comando

**TTY:** Terminal que ejecuta el proceso

Un programa en ejecución es un proceso, un programa de usuario es un proceso, una tarea de sistema también puede ser un proceso .

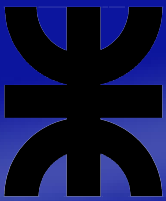


# Línea de comando

## Pstree

Mostrar todos los procesos jerarquizados en forma de árbol.  
usuario@mipc:~\$ pstree

```
init(1)-+-NetworkManager(4530)---{NetworkManager}(5168)
|-NetworkManagerD(4544)
|-acpid(4328)
|-atd(5328)
|-avahi-daemon(4586)---avahi-daemon(4587)
|-bonobo-activati(5662)---{bonobo-activati}(5666)
.
.|-gnome-terminal(7118)-+-bash(7123)-+-pstree(7524)
|                               | `--watch(7162)
|                               |--gnome-pty-helpe(7122)
|                               `--{gnome-terminal}(7124)
`-inetd(4927)
```



# Línea de comando

## **kill <PID>**

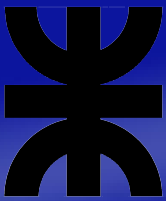
Eliminar un proceso kill

## **free**

Muestra información sobre el estado de la memoria del sistema, tanto la swap o de intercambio (páginas que han sido movidas a disco) como la memoria física. También muestra el buffer utilizado por el kernel.

usuario@mipc:~\$ free

	total	used	free	shared	buffers	cached
Mem:	2049900	814300	1235600	0	63444	457588
-/+ buffers/cache:		293268	1756632			
Swap:	2085884	0	2085884			



# Línea de comando

## **top**

Ordena en tiempo real los procesos según el consumo de CPU, memoria RAM, SWAP, entre otras características.

**PID:** Identificación del número de proceso

**USER:** usuario que corre la aplicación

**PR:** prioridad efectiva del proceso

**NI:** es un nivel para establecer la prioridad del proceso, mientras menor es mayor es la prioridad

**VIRT:** Total de memoria virtual usada (código, datos, memoria usadas por las librerías compartidas que utiliza el proceso y páginas que han sido movidas a disco (swap)

**RES:** Tamaño de segmento residente, es decir lo que ocupa nuestro proceso en RAM (código y datos únicamente)

**SHR:** memoria compartida que utiliza el programa, es la cantidad de memoria que se estima que este proceso está compartiendo con otros

**S:** Estado del proceso

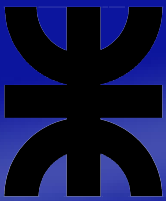
**% Cpu:** porcentaje de la Cpu que está tomando este proceso

**% Mem:** porcentaje de memoria que está tomando el proceso

**TIME+ :** tiempo de Cpu usado por el proceso desde que se creó

**COMMAND:** el comando que se está ejecutando





# Línea de comando

## **file <archivo>**

Mostrar el tipo de un archivo: file

usuario@mipc:~\$ file arc1.txt

## **cat <archivo>**

Mostrar el contenido de un archivo, cat

usuario@mipc:~\$ cat arc1

## **clear**

Corre pantalla del terminal hacia arriba y deja la última línea visible pero no borra las líneas anteriores

## **reset**

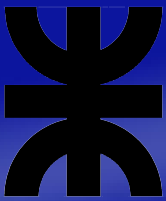
Borra lo que estaba escrito en la pantalla del terminal

## **cal**

calendario en pantalla

## **date**

fecha y hora actuales



# Línea de comando

## Comandos para apagar el equipo

**halt:** Apaga el equipo

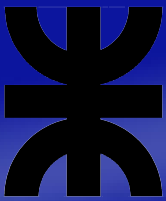
**reboot:** Reinicia el equipo

**shutdown -h now:** Apaga el equipo

**shutdown -r now:** Reinicia el equipo



# **GCC**



# GCC

## **GCC (GNU Compiler Collection)**

Es un conjunto de compiladores creados por el proyecto GNU

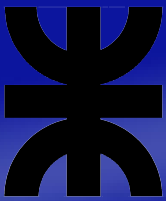
Es software libre, distribuido bajo la GNU (General Public License).

El desarrollo de GCC fue una parte del desarrollo de GNU.

En 1987 GCC fue el primer compilador portable para optimizar ANSI C liberado como software libre.

En 1992 llega la serie 2.0 en 1992, que añade la capacidad de compilar C++.

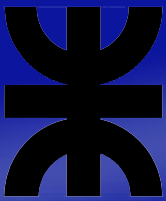
En 1997, se creó una rama experimental del compilador (EGCS) para mejorar la optimización y el soporte de C++. En 2001 llega la versión 3.0.



# GCC

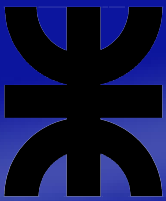
## Características de GCC

- Es un compilador portable y puede producir salidas para muchos tipos de procesadores. Además de procesadores usados en ordenadores personales, también soporta microcontroladores, DSPs y CPUs de 64 bits.
- Está escrito en C con un fuerte enfoque hacia la portabilidad, y puede compilarse a sí mismo, así puede ser adaptado a nuevos sistemas fácilmente.
- Tiene un diseño modular, permitiendo que el soporte para nuevos lenguajes y arquitecturas.



## Gedit

- gedit es el editor de texto por defecto de GUI en el sistema operativo Ubuntu.
- gedit incorpora una interfaz gráfica de usuario, podemos acceder (en ubuntu) desde Aplicaciones → Accesorios → Editor de texto , o desde una consola escribiendo gedit.
- Para abrir un archivo específico desde una consola:  
**usuario@mipc:~\$gedit** nombre de archivo



## Ejemplo de compilación y ejecución

```
/* hola */  
#include<stdio.h>  
  
int main() {  
    printf("Hola Mundo!\n");  
    return 0;  
}
```

### Compilamos el programa hola.c

```
usuario@mipc:~$gcc -o hola hola.c
```

### Ejecución

```
usuario@mipc:~$./hola
```

*Hola mundo!*

```
usuario@mipc:~$
```