.**Cat:** Muestra el contenido de archivos y crear ficheros.

– cat > [nombre fichero]

escribimos los datos que lleva el fichero

Control + D para cerrar el fichero y guardar

-n → Muestra el numero de lineas

**tr:** Cambia o borrar o sustituir caracteres por otros necesita la entrada obligatoriamente.

\n → Salto de linea

\t → Tabulador

-d → Permite borrar lista de caracteres y rangos de caracteres

tr -d [0-9] < fichero → Borraria todos los numeros.

-s → Borra la secuencia de caracteres seguidos y deja solo 1.

tr -s o < fichero → Borrara los caracteres que tengas varias o y dejaría solo 1. EJ → holoooooo = holo

**sort:** Es un comando filtro que nos redirige y no ordena los datos introducidos.

-r → ordena de orden inverso

-o → Remplaza el > : sort fich -o finch2

-c → Te dice si esta ordenado o no : cuando sale fuera de secuencia es porque no lo esta.

-u → Elimina las lineas repetidas dejando solo 1.

-f → Considera todas las letras como si fuesen mayusculas.

-n → Para ordenar los numeros.

**Para columnas:**

-k campo-comienzo [, campo-final] → Ordena a partir de campos.

-k3 → Ordenad a partir del campo 3 hasta el final de la linea

-k3,3 → Ordena el tercer campo

-k3,3 -k2,2 → Ordena primero el tercer campo y luego en campo 2

-k3,5 -k2,2 → Ordena desde el campo 3 al 5 y luego en campo 2

-k1,1 -k2,2 -k3,3 → Ordena el campo 1, luego el 2 y luego el 3

-t”separador” → Indica el separador que hay entre campo y campo

**more :** Muestra los archivos pausadamente.

- “numero” → muestras los archivos de la cantidad que le digamos.

+ “numero” → Muestra a partir desde el numero que le digamos.

| more → Va parando para poder ver todos los archivos.

**Head:** Visualiza las primera lineas de un fichero. Va de arriba hacia abajo

head - “numero” → nos saca las lineas correspondientes desde la primera hacia abajo

**tail:** Visualiza igual que el head pero desde abajo hacia arriba.

Tail - “numero” → Nos saca las lineas correspondientes desde abajo a arriba

**wc:** Nos muestra el numero de lineas, palabras y numero de caracteres, como ultimo da el numero de bytes

-l → muestra el numero de lineas

-c → Numero de bytes

-m → Numero de caracteres.

-w → muestra el numero de palabras

-lc → muestra los 2 parametros juntos.

**Cut :** extrae caracteres o campos de archivos

-c → Para cortar caracteres

-f → Para cortar campos. El carácter de separacion entre los distintos campos es, pode defecto, un espacio en blanco. Para indicar el separador se indica con -d

**grep:** Extrae lineas completas que cumplan las condiciones que le marquemos.

Grep “cadena”[-modificadores] → Las comillas se ponen cuando hay espacios

-E → Equivale al egrep

-i → Ignora mayusculas y minusculas

-c → Indica el numero de lineas que tiene la cadena

-v → Muestra las lineas que no contienen la cadena indicada

-w → Busca palabras completas, ya no como caracteres.

-e → Permite buscar varias condiciones en un archivo

grep -e “cadena” -e “cadena” fichero

-l → Solo aparece el nombre de los ficheros que contiene la cadena

-L → Solo aparecen los nombre de los ficheros que no contienen la cadena

**METACARACTERES: debe ir siempre entre ' '**

. → Significa cualquier carácter

^ → Indica el principio de una linea

$ → Indica el final de una linea

\* → Indica cero o mas repeticiones del carácter anterior

+ → Indica uno o mas repeticiones del carácter anterior, siempre se escapa. \+

\> o \b → Indican que la palabra termine con la condicion que indiquemos

grep 'sa\b' fich1 → Palabras que terminen en sa.

grep '\bsa' fich1 → Palabras que empiecen por sa.

[] → Lineas que contengan algun carácter que este introducido dentro del []

{} → Repeticion de n veces del carácter previo. Tiene que ir escapado siempre \{ \}

hola\{5\} → Repeticion de 5 veces del ultimo carácter.

Hola\{5,\} → De ese numero en adelante.

hola\{5,6\} → De 5 a 6 caraceteres.

**Ln:** Para crear enlaces duros.

-v → Nos muestra información sobre la creación del enlace.

-s → Para crear enlace simbolico

**unlink :** elimina enlaces ya sean duros o simbolicos.

**echo ${#variable}** → Muestra el numero de caracteres de la variable.

**-e** → Nos permite utilizar una serie de parametros:

\n → Salto de linea

\c → Elimina el salto de linea

\t → tabulador horizontal

**EXPRESION ALGEBRAICA**:

**let** → no se ponen espacios: c=let $a+$b

**c=$(($a + $b))** → Puede o no haber espacios

**c=`expr $a + $b`** → Espacios obligatorios, da error si una variable es alfanumérica

**Expr:** Nos permite operar con números. Los elementos deben estar siempre separados y debe ir entre ``

* Resultado=`expr $nu1 + $nu2`

**$(( … )):** Nos permite ejecutar una expresión algebraica.

* echo $((4+6))

**$[ … ]:** Igual que el anterior.

* echo $[4+6]

**bc -l** → Nos muestra los decimales

echo “$x / $y” | bc-l

echo “scale=3;$x / $y” | bc-l → El scale sirve para determinar los decimales.

**who:** Nos va a indicar el usuario que esta logueado en la maquina.

**Touch:** Crea archivos vacios.

* Touch archivo

**File:** Nos va a mostrar de que tipo es el archivo (vacio, ASCII text…).

* file archivo

**pwd:** Nos dice la ruta absoluta del directorio donde nos encontramos situados.

**chmod:** Para cambiar permisos de un archivo o directorio.

* Chmod 777 archivo
* Chmod +r archivo
* Chmod o-x,g-w archivo

**env o printenv:** Mostrar las variables de entorno.

**colores=(rojo azul verde):** Crear variable array

**echo ${colores[0]}:** Mostrar el primer valor de un array

**declare:** Nos permite crear una variable con unas determinadas características.

**export:** Nos permite exportar una variable para que tenga valor en otras bash siempre y cuando sean hijas de la bash actual.

* Export variable
* Export variable=valor

**alias:** Nos permite crear un “comando” a través de alias que ejecutara la orden que nosotros le indiquemos. Va siempre entre comillas simples ‘’

* alias dir=’ls –l’
* \dir  Ejecutario el comando dir sin fijarse en alias.

**find:** Nos permite buscar archivos por el nombre del archivo.

* Buscar propiedades man find.

SCRIPT:

**$0:** Muestra el nombre del archivo.

**$n:** Muestra el argumento numero n que le metamos al script. Para un argumento con numero mayor a 9 se especifica ${n}

**$\*:** Contiene todos los argumentos introducidos al script

**$#:** Contiene el numero de argumentos que es le han introducido al script.

**TEST:** Nos permite validar opciones de archivos, directorios, etc…

-a 🡪 archivo existe. -d 🡪 existe y directorio.

-f 🡪 existe y regular -h 🡪 existe y enlace simb.

-r 🡪 existe y lectura sobre el -s 🡪 existe y no vacio

-w 🡪 existe y escritura -x 🡪 existe y ejecución

*Archivo1* -nt *archivo2* 🡪 1 es mas reciente que 2, o 1 existe y 2 no.

*Archivo1* -ot *archivo2* 🡪1 mas antiguo que 2, o 2 existe y 1 no.

*Archivo1* -ef *archivo2* 🡪 1 es un enlace duro al 2.

**PROCESOS**

**pstree** → Para ver los procesos que se estan ejecutando

-p → el pid

-u → aparece nombre de usuario

-a → otra manera de expansion

-c → expandido

**top 🡪** Muestra los procesos en ejecución con actualización en tiempo real.

**ps 🡪** Muestra un lista de los procesos en ejecución.

-e 🡪 Muestra información de todos los procesos

-l 🡪 Muestra información en formato largo

-u <nombre> 🡪 Muestra los procesos del usuario <nombre>.

-o <formato> 🡪 Permite usar un formato mostrando solo los campos que se deseen:

ps -o pid,cmd,size

**Para ejecutar un comando en segundo plano se pone el comando seguido de un espacio y &**

**jobs** → Nos muestra los procesos que se esta ejecutando en segundo plano

-p → Muestra el PID

-l Muestra toda la información metiendo tanto el orden de trabajo como el PID

**% 🡪** Trabajo actual

**%- 🡪** Trabajo previo al actual

**%n** 🡪 Trabajo numero **n**

**fg** → trae a primer plano un proceso que esta en segundo plano o parado

**fg [[%]num\_trabajo]**

**bg** → Ejecuta en segundo plano un trabajo que esta parado o detenido

**bg [[%]num\_trabajo]**

**wait 🡪** Espera la finalización de procesos en segundo plano.

**disown 🡪** Suprime un trabajo de la lista de trabajos activos

**kill** → Sirve para enviar señales a los procesos.

Kill -señal (PID o %Nº de trabajo)

-l → Nos muestra todas las señales

SEÑALES:

-9 → Mata a un proceso si o si

-STOP → suspende el proceso

-CONT → para revivir el proceso

-HUP → Relee los ficheros de cofiguracion

**killall** → Igual que el kill pero indicando el nombre del proceso

killall [señal] <nombre del proceso>

-i → Pide confirmacion al elimiar los procesos

**pgrep** → equivale al ps y un grep, busca procesos.

-l → Nos muestra tanto el PID como el nombre

**pkill** → Funciona exactamente igual que el pgrep pero permite mandar señales por el nombre del proceso.

**bash -x Archivo 🡪** Muestra línea a línea el scrip.

**COMPILAR:**

**g++ 🡪** Nos permite compilar un programa usando los objetos o archivos fuente del mismo.

Podemos compilar un programa directamente con:

g++ fuente.cpp

-c 🡪 Para crear el objeto ce un .cpp; (\*\*.o)

-o 🡪 Elegir el nombre del programa a crear

-g 🡪 Permite depurar el programa resultante.

-L<ruta> 🡪 Indicar la ruta donde se encuentran las librerías.

-l 🡪 indicar el nombre de la librería.

Se pone al final, el nombre debe ser lib\*.a y se especifica: libmates.a --- -lmates

-I<ruta> 🡪 Para especificar la ruta donde se encuentran las cabeceras.

**ar -rvs 🡪** Nos permite crear una librería (.a) a través de varios objetos (.o).

ar -rvs libmates.a sin.o cos.o tan.o

**make 🡪** Para ejecutar un makefile sobre un programa.

-f 🡪 especificar el nombre del makefile. Por defecto busca makefile y Makefile.

Podemos crear variables dentro del makefile para indicar los objetos (OBJS) o para indicar el compilador que usamos (CC), el directorio de las librerías (LIB\_DIR) o el directorio de las cabeceras. Para usarlas -- $(variable)

**$@ 🡪** Indica el nombre del objeto dentro de la orden.

**$< 🡪** Primera dependencia

**$? 🡪** Todas las dependencias de la regla que se an actualizado.

**$^ 🡪** Todas las dependencias del objeto

ESTRUCTURA:

objeto: dependencias

ordenes a ejecutar.

**DEPURACION:**

**gdb 🡪** Nos permite ejecutar la depuración de un programa

Para poder depurar un programa debemos usar la opción -g en el g++

-x 🡪 Carga todas las secuencias guardadas en un archivo .gdb y las ejecuta.

OPCIONES DENTRO DE GDB

**quit 🡪** Salir de gdb.

**run** 🡪 Inicia el programa.

**break “nombre función”** 🡪 Crea una parada cuando encuentre la función.

**break línea** 🡪 Crea una parada en la línea indicada.

**break línea if a > 10** 🡪 Parada en línea si a>10

**list 🡪** Muestra 10 lineas del programa.

**next 🡪** Va pasando de orden pero si se encuentra un subprograma, lo ejecuta sin mostrar el código.

**step 🡪** Igual que next pero si muestra el código de los subprogramas.

**set variable “nombre\_varaible” 🡪** Cambia el valor de la variable indicada.

**display variable** 🡪 Muestra el valor de una variable cada vez que se detiene en un punto de ruptura.

**print variable** 🡪 Muestra el valor de una variable en ese momento.

**attach PID 🡪** Permite depurar un programa en ejecución.

**shell “orden” 🡪** Ejecuta una orden desde gdb

**delete display id** 🡪 Elimina un display

**examine dirección** 🡪 Examina el contenido de una dirección de memoria. Se expresa en hexadecimal.

0x000f1

**show values** 🡪 Muestra el historial de valores de las variables impresas

**p/x $pc** 🡪 Muestra el contador de programa usando su dirección lógica.

**x/i $pc** 🡪 Muestra la siguiente instrucción que se ejecutará usando el contador de programa.

**disassemble** 🡪 Muestra el código ensamblador de la parte que estamos depurando.

**whatis variable** 🡪 Devuelve el tipo de una variable.

**info locals** 🡪 Lista todas las variables locales.