

Redirección redirección { se toma entrada de teclado ("here document").
o toma entrada de un archivo).

cat > fichero cat TSO & todo lo que entre por la entrada estandar (teclado) pasa a formar parte del fichero. Termina al escribir TSO (o' CTRL+D dle).

tee & (es transparente) pone la entrada que recibe en su salida, pero copia esa información en un fichero que se le proporciona en la llamada. (copia a medio camino).

Tuberías (pipes)

"I" & mecanismo de comunicación que une dos filtros mediante la bash. La salida estandar del primer filtro se hace la entrada (se redirige y no se imprime) estandar para el segundo.

Filtros

cvt

-c & cortes por columnas.
-f & cortes por campos (fields).

cvt -c1-10,21- & selecciona los caracteres 1 a 10 y del 21 al ultimo de la entrada estandar.

cvt -d & fija un delimitador.

cvt -d: -f1,5-7 fichero & selecciona los campos 1, 5, 6, 7 tomando como delimitador (:).
si no se pone, se supone que el delimitador es el carácter tabulador.

paste & Concatena por columnas cada linea.

paste f1 f2 f3 & imprime las columnas de cada fichero. Separadas por tabuladores.

paste -d=f1 f2 f3 & " ". Separadas por (=).

join & producto cartesiano de 2 ficheros ordenados y muestra los que tienen la misma clave.

join -o (out put) K(fichero).n(partे a mostrar) K2(fichero2).n2(partе a mostrar2)

join -1 2 -o 1.1 2.2 f1 f2 & -1 2 indica que f1 se use el 2do campo.

-o 1.1 2.2 & pide que la salida contenga el campo 1º (.1) de f1(1.) y el 2º campo (.2) de f2(2.).

comm & Compara conjuntos ordenados, presenta la salida en 3 columnas:

- Columna 1 presenta las líneas que solo están en el primer fichero.
- Columna 2 presenta las líneas que solo están en el segundo fichero.
- Columna 3 presenta las líneas que están en ambos ficheros.

comm -n₁ -n₂ & Oculta las columnas nombradas.

tac & Escribe las líneas en orden inverso (head -1 escribe el ultimo en vez del primero).

rev & Escribe cada linea en orden inverso (rev diras/head -1 escribe revertido).

uniq & Elimina las líneas que son repetidas consecutivas.

uniq -c & " ". Escribe cuantas veces se repite de forma consecutiva cada linea.

uniq -d & Solo escribe las líneas repetidas consecutivas.

sort → ordena líneas. Toma la ordenación de los caracteres ASCII.

sort -n → ordena números de menor a mayor.

sort -rn → ordena números de mayor a menor.

sort -u → ordena las líneas eliminando las que son iguales.

sort -r → ordena inversamente al alfabeto.

grep → Busca líneas que contengan la expresión regular.

grep x f1 → Muestra las líneas que cumplen x.

grep -v → Busca el negado de la expresión regular.

grep '' f1 → Permite buscar en el fichero con condiciones.

'x#' → termina en x.

'^x' → empieza en x.

'x.y' → el punto indica cualquier carácter.

sed → Es un editor no interactivo. Útil para realizar varias modificaciones simultáneas.
El fichero original no queda modificado, los cambios aparecen en la salida estandar.
Sed aplica todos los comandos linea por linea esperando por la 1^a.

sed -f f1 x → toma los comandos de f1 y los aplica en x y siguientes.

sed -e comando -e comando → toma los comandos de la misma linea de invocación.

[dirección [, dirección] [!]] función [argumentos]

• Si no se pone dirección se aplica a todos los términos.

• Las direcciones pueden ser números, # o expresiones regulares entre comillas(1).

• Exclamación (!) después de la dirección hace que se aplique al conjunto complementario (traducido como "aqui no"). ← no sirve sed -e '/[^u]ld' → hay cadenas vacías que hacen que se elimine todo.

sed -e 'parte donde'

Escribir o imprimir (p): duplica las líneas afectadas.

sed -e '3,5p' → imprime dos veces las líneas de la 3 a la 5. → Debajo de rango es la coma.

sed -e '1u/p' → duplica las filas que contienen la u.

Borrar (d):

sed -e '3d' → borra la linea 3.

sed -e '3!d' → borra las líneas que no sean la 3.

sed -e '3,\$d' → borra desde la linea 3 hasta el final.

sed -e '1,1s/d/ → borra todas las líneas que tengan 'd' si:

Sustituir (s):

sed -e '1brie \$/ste/XXX/g' → sustituye la 'e' de todas las filas que terminan por 'bre' por 'XXX' (la g indica global, osea todas las apariciones).

Leer (r):

sed -e 'f,5r fichero' imprime el contenido del fichero tras cada linea afectada.

Añadir(a): añade la palabra que ponemos a las líneas afectadas.

sed -e '1,5a PALABRA' añade PALABRA a las líneas 1-5.

tr - Espera dos parámetros. El primero la lista de caracteres a cambiar. El segundo es la lista de caracteres resultantes del cambio.
No admite un nombre de fichero como entrada.

tr aeiou aaaa = tr aeiou a → combina las vocales minúsculas por aes.

↑ cuando el segundo parámetro es más corto que el primero, se prolonga con la repetición del último carácter.

tr A-Z → cambia las mayúsculas por (-).

tr -c → Aplica la sustitución a los caracteres que no son los que indicamos.

tr -s → Quita las repeticiones de caracteres resultantes de los cambios.

tr -d → Indica que se borrarán los caracteres del primer parámetro (no hay opción de 2º parámetro).

Expresiones regulares

Generalmente describen las tiras de caracteres incluidas en una linea.

\ → cancela el significado de caracteres especiales.

^ → Inicio de linea.

\$ → Fin de linea.

. → cualquier carácter, salvo el salto de linea.

[] → Alternativa de caracteres [-] → rangos.

[^a-zA-Z] → cualquier carácter salvo los citados y el cambio de linea.

a* → también es una expresión regular. cero o más repeticiones de a.

.* → cualquier cosa (trozo de cadena de cualquier longitud, incluso cadena vacía).

\(a+b)\) → (Sub) expresión marcada en la que cero o más caracteres \ van seguidos de un carácter 'b'.

\2 → Referencia a la segunda (sub)expresión marcada.

..* → cualquier tira de caracteres, con al menos un carácter.

Programas del intérprete de comandos. Script.

El intérprete de comandos trata de forma especial una serie de caracteres:

* ? [-] <> | & # # ' " + \ ()

- Estos caracteres entre comillas sencillas pierden su tratamiento especial.
No se pueden incluir comillas sencillas entre comillas sencillas.

- Entre dobles comillas el intérprete sólo da tratamiento a: \$ \ ' .

- (\) el siguiente carácter al back este sí pierde su tratamiento especial.

Permisos

- Dueño (propietario) **M**
- Grupo **G**
- Otros **O**

- En ficheros, la mayoría de operaciones se asocian a tres tipos:

- Lectura (read) → **r**
- Escritura (write) → **w**
- Ejecución (execute) → **x**

Los permisos se pueden codificar en octal **uuu** (nº del 0 al 7).

Son la suma de los siguientes valores:

- r → lectura → **4**
- w → escritura → **2**
- X → ejecución → **1**

uuu
↑ ↑ ↑
Propietario Grupo Otros

664 ← r, w | r, w | r
775 ← r, w, x | r, w, x | r, x
622 ← r, w | w | w

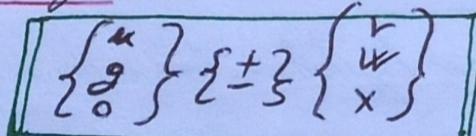
- Para los directorios se consideran 3 tipos de operaciones: lectura (r), escritura (w) y el uso del i-nodo (x).

Con el permiso de lectura podemos conocer los nombres de los objetos en un directorio.

Con el permiso x, dado un nombre en un directorio, podemos acceder al nodo y al objeto (fichero o subdirectorio).

chmod → cambiar los permisos.

chmod u+x f1 → añade al dueño permiso de ejecución de f1.



chmod a-w f1 → quita a todos el permiso de escritura.

chmod 644 f1 → pone los permisos de f1 **rwx-r--r--**

Permisos iniciales → permisos establecidos al crearse o ejecutarse.
umask → nos informa del valor de la máscara de creación.

umask 037 → cambia el valor de la máscara de creación a 037.

Los valores habituales de la máscara de creación son 002 y 022.

El comando umask es interpretado. El interpretador de comandos no lo delega en otro proceso se quiere cambiar el valor de una variable local.

Procesos

UID (user id): Propietario del proceso.

PID (process id): Identificador de proceso (al terminar el nº se reutiliza).

PPID (parent process id): identificador del proceso padre.

CPU: tiempos de CPU.

MEM: direcciones de memoria virtual.

TTY: cable por el que está conectado.

ps (process status) → Muestra el estado de los procesos asociados al terminal.

ps -f → Muestra el estado de sus padres e hijos.

ps -l → Indica la prioridad de procesos.

ps -ef → Muestra la lista completa de procesos (de todos los usuarios).

pstree → Muestra de un proceso todos sus hijos.

Kill → Envía una señal al proceso indicado y lo para.

Kill -9 pid → Obliga a la finalización del proceso.

sleep → Espera los segundos indicados y acaba

top → Muestra los procesos que consumen más CPU.

htop → Muestra los procesos, con interfaz gráfica y pudiendo filtrar.

• Procesos en background y foreground

La bash no espera a que los hijos esperen para que el padre continúe.

Se puede pasar de background a foreground y viceversa.

& → pasa comandos al background (segundo plano).

fg → pasa a primer plano (foreground)

fg %n → muestra el nº de planos que quieras

bg → pasa a segundo plano

sleep 5 & → en el foreground

sleep 5 & → en el background

AWK

Es un filtro para ficheros de texto. Es programable.

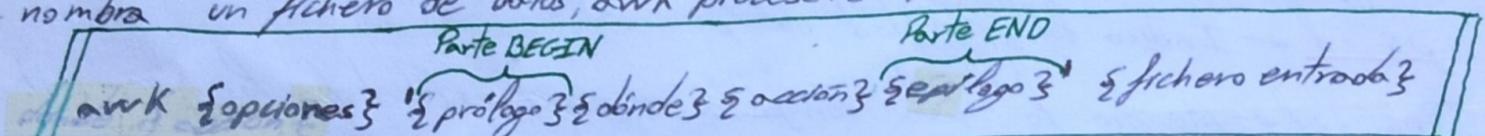
awk → programa ficheros, aplica el programa sobre cada línea de cada fichero nombrado.

awk '1;1;1' f1.f3 → Escribe dos veces cada línea de f1 y f3 que tenga una (a). Los comillas son para que el interprete de comandos (sh, bash, ms) no las vea.

El programa puede estar en un fichero. Se especifica con awk -f.

En los programas awk el punto y coma (;) se puede cambiar por el cambio de linea y viceversa

Si no se nombra un fichero de datos, awk procesará la entrada estandar.



- La parte prologo y sealogo puede tomar valores especiales BEGIN y END.

- Se ejecuta la acción asociada a BEGIN antes de tratar ninguna entrada.
- Para registros de entrada:
recorre la secuencia de pares donde - acción. Si cumple el registro se ejecuta la acción.
- Se ejecuta la acción asociada a END después de tratar el último registro de entrada.

Si se omite la parte donde se aplica la acción a todos los registros de entrada.

- La parte donde: [dirección, [dirección]] las direcciones pueden ser expresiones booleanas o patróns (patrones, expresiones regulares entre paréntesis ()). → expresiones aritméticas ($>$, $<$, $=$, \neq , $>=$, $<=$, $!=$)

- Si se ponen dos patróns (expresiones regulares entre paréntesis), estos definen unos a modo de párrafos.

- El primer a modo de párrafo comenzará con el primer registro en que se encuentre (un ajuste a) la primera expresión regular. Acabará en el primer registro en que (después) se encuentre (un ajuste a) la segunda expresión regular.

- El segundo a modo de párrafo comenzará con el primer registro en que se encuentre (un ajuste a) a la primera expresión regular.

- Si se ponen dos expresiones booleanas, estas también definen unos a modo de párrafos.

- La parte acción es una secuencia de sentencias. Normalmente las secuencias van en líneas distintas (pueden ponerse en la misma línea separándolas por ;).

- Sentencias de salida print → formato impreso.
printf → formato explícito.

- Sentencias de asignación.

- Sentencias de control de fluxo:

- if-else

- while

- for (; ;), for (in)

if [condition]
then

fi

while [condition]
do

DONE

for VAR in ELEMENTO
do

DONE

awk considera el texto como una secuencia de registros y a cada registro como una secuencia de campos.

El tipo de dato predominante en awk es el tipo tira de caracteres. Sus valores son las secuencias de caracteres incluyendo la secuencia vacía.

Las constantes de tipo tira de caracteres se escriben entre comillas dobles.

El delimitador de registros por omisión es el cambio de linea.

El delimitador de campos por omisión es el carácter blanco.

Se puede cambiar el valor del delimitador de campos en la linea de invocación a awk o en el protago (BEGIN).

awk -F: → Cambia el separador de campos a (:).

BEGIN {FS=":"} → Cambia el separador de campos a (:).

Si el separador de campos es el carácter : y en una linea (en un registro) tenemos:

::: agua :: fuego

la tira agua es el cuarto campo, fuego es el sexto campo, hay seis campos y cuatro de ellos son tiras de caracteres vacíos.

si el separador de campos es el carácter blanco (:) y en un registro tenemos:

uuu agua uuu fuego

la tira agua es el primer campo, fuego es el segundo campo, y hay dos campos. Con este separador no puede haber campos cuyo valor sea la tira vacía.

Los campos se consideran un caso particular de variable:

\$1 → es el primer campo

\$2 → es el segundo campo

:

\$99 → es el campo nonogestronoveneno

\$0 → es todo el registro

El valor de un campo inexistente es la tira vacía.

\$NF → contiene el último elemento de la linea

Algunas variables predefinidas de awk son:

NF → número de campos del registro actual.

NR → número de la linea que está siendo procesada

FILENAME → nombre del fichero de entrada
no entra

print → secuencia de expresiones y como imprime los valores de las expresiones.

Las comas en la secuencia de expresiones y como generan caracteres separadores de campos en la salida.

print FILENAME, NR, \$1, \$2 → Escribe el fichero en procesamiento, el número de registro, el primer y el segundo campos del registro. Intercala tres separadores de campos de salida.

substr → toma una cadena de texto (string), una posición de inicio (índice) y una longitud, y devuelve una subcadena de la cadena original. La posición inicial se cuenta desde 1.

substr("Hola mundo", 1, 4) → Hola
Estos funciones modifican la cadena original

length → devuelve la longitud de una cadena de texto. length("Hola Mundo") → \$ → 10

index → busca un carácter o subcadena dentro de una cadena de texto y devuelve la posición en la que aparece por primera vez. Si no se encuentra devuelve 0. index("Hola Mundo", "Mundo") → 5

split → divide una cadena de texto entre trazos según un separador especificado y los almacena en un array. Devuelve el número de trazos en que se divide la cadena.

split("Hola-Mundo", a, "-") → dividiría la cadena

en 2 y la mete en a.

cat & cat & asteriscos

lma - lmes - monday
marte - martes - tuesday

awk -F '15 print FILENAME, NR, \$1, \$2' cat & asteriscos
dastros & lma lmes
dastros 2 marte martes

print "fichero", FILENAME ":", NR, \$1, \$3 → escribe "fichero", el nombre del fichero que estás procesando seguido de :, el número de registros y los campos primero y tercero. Intercala separadores de campo en la salida (donde están los comas).

print NF, \$NF → escribe el número de campos en el registro y el último campo del registro. Si en un registro hay 4 campos al tratar ese registro NF valdrá 4. \$NF será \$4, es decir, el último campo.

print → equivale a print \$0. Escribe todo el registro.

print "" → escribe una linea vacía

printf → escribe siguiendo un formato que se expresa como una fija de caracteres y casi siempre es un literal (una fija).

Sentencias de control de flujo
if, if-else, while, for

Operadores lógicos

&& → y

|| → o

! → no

printf (formato, lista de parámetros)

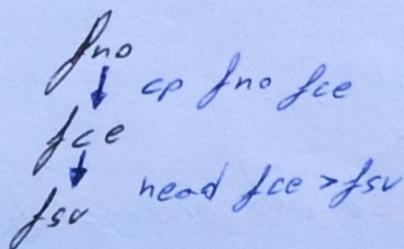
printf("%9s %9s %9s", \$3, \$2, \$1) → Deja 9 espacios para cada string



make

Facilita la automatización de ficheros.

El comando make se apoya en un fichero con reglas cuyo nombre usual es Makefile.



- Se recorre hacia arriba el subgrafo. Si la fecha de un fichero objetivo es anterior a la fecha de un fichero requisito se marca la acción como pendiente.
- Se recorre el subgrafo. Si acción estaba pendiente se ejecuta y se marcan como pendientes las siguientes dependencias.

make objetivo =

- make reglas
- si la fecha (objetivo) < max (fechas (reglas))
aplica acciones de regla de objetivo.

make tiene esquemas predefinidos si tenemos un fichero frente del lenguaje C 'prueba.c' con un programa sencillo y sin tener fichero Makefile, el fichero compilaria correctamente obteniendo el ejecutable prueba.

Es un error declarar en el fichero Makefile reglas que indican dependencias circulares.

```
b1a cp a b
a1b cat b b1b head -30 >a
```

Compresión

Los ficheros comprimidos mantienen la fecha de última modificación (y otros datos del i-nodo) del fichero original, para que al descomprimir la situación sea lo más parecida posible a la de partida.

tar → El formato tar es una forma pensada para agrupar y guardar (archive) uno o varios ficheros en cinta magnética (tape).

tar se usa con una de las tres opciones siguientes:

c → para crear objetos de tipo tar.

t → para examinar el contenido (nombres, tamaños,..., de los ficheros) en objetos de tipo tar.

x → para extraer todos o parte de los ficheros contenidos en objetos de tipo tar.

Empaquetar

Estas opciones no van precedidas del carácter -, habitual en otros comandos.

- La opción f indica que a continuación se nombra el destino (del objeto de formato tar).

tar cf [/dev/rmtp] f1 f3 ← escribe en la cinta colocada en el dispositivo '/dev/rmtp' los ficheros f1 y f3.

tar cvf nombre directorio_patre → guarda en un fichero nombre.tar todo lo que haya hasta el directorio padre.

tar cvf nombre fichero1 fichero2 → Si se ejecuta tar desde el directorio de los archivos, solo se guardará el fichero(s) en el archivo tar, sin la ruta. Pero si ejecutas tar desde otro directorio e incluyes la ruta a los archivos, se mantendrá la estructura de directorios en el archivo tar. Esto significa que guardará la ruta completa a los archivos.

- La opción v viene de verbose (charlatán). Al crear un objeto tar, escribe el nombre de los objetos archivados. Al extraer ficheros o examinar un objeto tar escribe una linea por cada fichero semejante a la que escribe ls -l.

Listar

tar tvf nombre → lista el contenido del fichero nombre. Su formato debe ser tar.

Desempaquetar

tar xvzf nombre → extrae todos los ficheros contenidos en nombre. Su formato debe ser tar. Crea directorios cuando sea necesario.

tar xvzf nombre ruta/ruta.../fichero → extrae únicamente el fichero indicado con los directorios correspondientes de los que depende.

Comprimir

tar czvf nombre directorio-padre (.tgz por convenio, g-zip...) → Empaquetá toda la información del directorio padre comprimida.

Los archivos tar comprimidos con gzip, se deben usar vzf en lugar de zf para indicar que el archivo está comprimido.

tar tvzf " → se usa para listar el contenido de un archivo tar comprimido con gzip.

tar xvzf " → se usa para extraer un archivo tar comprimido con gzip.

VARIABLES LOCALES → Solo vive mientras la sesión está abierta.

var = \$(comando) → Variable de entorno var

\$var → Ejecutará el comando almacenado en la variable var. Obtiene el valor de la variable (\$), y su nombre.

export var = \$(comando) → Variable global, persiste tras la sesión.

var = "contenido" → Los valores de la variable son del tipo tipo de caracteres. Los comillas dobles son solo necesarios si queremos que el valor asignado tenga blancos, tabuladores u otro carácter especial.

El signo = debe seguir inmediatamente a la variable, para que el nombre de la variable no se confunda con el nombre del comando. También el valor a asignar debe seguir inmediatamente al signo =.

\$HOME → argumento por omisión del comando cd.

/home/capell/buscador/tfg/log → \$HOME/Bc/log

\$PATH → Es la lista de escritorios donde se buscan los comandos. Los directorios están separados por (:). Si la lista empieza en (:) esto equivale a buscar en primer lugar en el directorio actual (.). Hay un punto importante.

Sequencias

\$? → da el valor devuelto por el último comando ejecutado.
0 → todo ha ido bien
1 → ha ocurrido un error
2, 3, 4... → diferentes errores.

VARIABLES GLOBALES o de entorno → accesibles para todos los procesos.

export var = contenido

Los procesos hijos reciben una copia de estas variables.

unset var → elimina la variable durante la ejecución.

env → muestra las variables globales.

Las variables locales y globales distinguen entre mayúsculas y minúsculas. Además, los nombres de las variables no pueden empezar con un número y no pueden contener espacios.

Mientras las variables globales persisten en entornos nuevos e incluso al final de la sesión, las herencias son solo y exclusivamente de padres a hijos, una subshell que crea una variable de entorno no afectará al padre.

Sin embargo las variables locales se eliminan al cerrar la sesión y persisten en una subshell, igual que las globales, la herencia solo es de padre a hijo.

\$\$ → contiene el id del proceso (PID) del script actual. Util para crear nombres de archivos temporales únicos.

→ contiene el número de argumentos o parámetros que se pasaron al script.

#* → contiene todas las argumentos que se pasaron al script.

Diferencia entre ?* y *.

?* En la bash, el carácter ? coincide con cualquier carácter individual mientras que * coincide con cualquier número de cualquier carácter. Por tanto, ?* coincidirá con cualquier cadena que tenga al menos un carácter.

* En las expresiones regulares, el carácter . coincide con cualquier carácter individual, y el carácter * significa "cero o más del carácter anterior. Por tanto, .* coincidirá con cualquier cadena, incluyendo la cadena vacía.