Comandos

- **echo** $PATH: imprime lista de directorios de comandos separados por dos puntos.
    - echo $?: devuelve el ultimo resultado del último comando (0 si salio bien, 1 si hubo error)
- **touch** [nombre]: crea un archivo
    - **touch -a --date=**"2018-03-06" [archivo] :Cambia la fecha de acceso
- **mv** archiname ar-modificados/archiname : mueve un archivo a otra carpeta
- **cat** [archivo]: imprime el contenido de un archivo
    - cat [archivoOri] **>**(o 1>) [ArchivoDes]: guarda PISANDO el contenido en un archivo (Usa el **STDOUT**)
    - cat [archivoOri] **>>** [ArchivoDes]: Guarda CONCATENANDO el contenido en un archivo (puede usarse con nombres entre “”. ej: echo “hola” >> “var-set”)
    - cat [archivoOri] **1>** [ArchivoDes] **2>** [archivoErr]: guarda salida en archivoDes y si hay un error guarda en archivErr. (usa **STDOUT** Y **STDERR**)
    - cat [archivoOri] **1>** [ArchivoDes] **2>&1**: Guarda la salida y el error en el mismo archivo
    - cat [archivoOri] **1> /dev/null 2>&1**: se deshace de la salida y el error
- **mkdir** [texto]: crea directorio
    - mkdir **-p** [dir1]/[dir2]/[dirx]: crea directorios recursivamente.
- **rm**: borra ficheros o directorios
    - rm **-r** borra recursivamente directorios (si el directorio no está vacío)
    - **rmdir**: borra directorio solo si están vacíos.
- **nano**: abre el nano
    - nano [archivo]: crea o abre un archivo
- **head** [archivo]:muestra primeras 10 líneas
    - head **-n** [num] [archivo]: n es la cantidad de líneas que muestra
    - tail -n+[num+1]: no muestra las num primeras lineas
- **tail** [archivo]: últimas 10 líneas
    - tail **-n** [num] [archivo]: últimas n líneas
    - head -n-[num]: no mostrara las num ultimas lineas
- **wc [archivo]**
    - **wc -c**: Cantidad de bytes del texto. devuelve número y nombre
    - **wc -m**: Caracteres
    - **wc -l**: Líneas
    - **$(find $1 | wc-l)-1** : cuenta cantidad de archivos (se debe restar uno por “.”)
    - **cat** copia | **wc -c**: hace lo mismo que arriba pero solo devuelve el número
    - **wc -l** < /etc/passwd: lee de STDIN el archivo y ejecuta el comando wc(cuenta las líneas) (Devuelve sólo el número, desconoce el origen de esta forma.)
    - **cat** /etc/passwd | **wc -l** : hace lo mismo que arriba pero con | (concatena comandos)
- **pwd:** te dice donde estas parado
- **env**: devuelve las variables de entorno una abajo de la otra (su ruta)
    - **env** | **sort**: devuelve pero ordenado alfabéticamente
- **ls -la**: lista los archivos con sus permisos
- **chmod** 777 [archivo o dir]: establece permisos para archivo

    - **chmod ugo+x** [archivo o dir]: se da permiso de ejecución a dueño,grupo u otros
    - **chmod a+x** [archivo o dir]: hace lo mismo de arriba
    - **chmod a-wx** [archivo o dir]: saca los permisos de w y x a todos
- **cut -c**[n]**-**[n2]: corta el contenido desde n a n2
    - **cut -d”**[delim]**” -f1**: toma el contenido antes del delim
        - usarlo asi para script $(echo $texto | cut -d":" -f7)
    - **cut -d”**[delim]**” -f2**: toma el contenido después del delim
    - **cut -d”**[delim]**” -f1,6,7**: toma el contenido de las columnas 1,6 y 7 delimitadas por delim
- **sleep** [numero]: espera [numero] segundos
- **top**: muestra los procesos en tiempo real (online)
- **ps -aux**: muestra los procesos ejecutados en ese momento (offline)
    - **ps -e**: Muestra todos los procesos.
    - **ps -ef**: Muestra todos los procesos en formato completo.

- **kill** [pid] : mata el proceso del PID
    - **kill -9** [pid]: busca dentro del pcb del proceso y mata los hijos antes del padre
- **sort** [archivo]: ordena el contenido e imprime
    - **sort -n** [archivo]: ordena numéricamente
    - **sort -rn** [archivo]: ordena al revés numéricamente
    - **sort -u**: ordena y quita repetidos
- **sed** "**s/**[caracter1]**/**[caracter2]**/g**" : reemplaza caracter1 por caracter2 (TODO VISUALIZACION, PARA QUE SE MANTENGA SE PONE “ -i ”)
    - **sed 's/’**[cadena]**’//g'** [archivo]: elimina cadena de un archivo (la cadena se pone sin comillas) (la salida es en texto)
    - **sed '/^$/d'** : borra lineas vacias
    - **sed s/^$/9/g | grep -v '9'** :usar si falla anterior. borrar lineas vacias (el 9 es cualquier caracter)
    - **sed 's/ //g'** : borra los espacios en blanco entre caracteres
    - **sed '2,5d'** [archivo]: borra de la línea 2 a la 5 del archivo (sed $desde","$hasta"d" $1)
    - **sed -i '2,5d'** [archivo]: borra las lineas y mantiene ese cambio en el archivo.
    - **sed 's/^/ /'**: inserta 3 espacios blancos al principio.
    - **sed -i 's/.*/\U&/' $1** : convierte el contenido de minuscula a mayuscula.
    - Borrar todos los espacios:

      while read linea; do

      echo "$(echo $linea | tr -d '[[:space:]]')" >> prutxt

      done<$1

      sed -i '/^$/d' prutxt

- **tar** -xzvf $HOME/practicas/tp1.tar.gz: descomprime un archivo en la ubicación
    - **tar -xzvf** $HOME/practicas/tp1.tar.gz -C [destino]: descomprime un archivo en destino
    - **tar -czf** listado.tar.gz listado: comprime un archivo

- **exit**: sale del script
- **stat** [archivo]: te da detalles del archivo
- **du** -hs [archivo o directorio]: devuelve el tamaño
    - du -b: devuelve el peso en kbytes (usar este)
    - du -b [archivo] | awk '{ print $1 }' : devuelve unicamente el peso del archivo // usar mejor “ cat copia | wc -c: “
- **basename** [archivo]: devuelve el nombre del archivo
    - basename -s [extensión] [dirArchivo] : devuelve nombre del archivo sin extensión. (extensión sin comillas)
- “cadena” **| tr '[:lower:]' '[:upper:]'** : reemplaza minúsculas por mayúsculas (SOLO MUESTRA)
- **History**: es un comando muy útil para averiguar los últimos comandos que se han ejecutado en un Servidor. El comando mostrará, por defecto, los últimos 500 comandos ejecutados
- **seq** n: muestra uno abajo del otro una secuencia de números
- **awk** '{ print $n }': imprime según columnas(n) en el contenido de un texto (usar cuando no se puede usar cut)
- cat archivo | sort | **uniq**: ORDENA Y SACA LÍNEAS REPETIDAS
  cat archivo | **uniq -c** : saca los repetidos y dice cuantos repetidos tenia cada palabra
- **man** [comando]: entra al manual
    - (dentro del manual) /[texto]: busca dentro del manual
    - ctrl+r: busca en comandos viejos
- **clear**: limpia la pantalla
- **“\n”:** poniendo esto en un texto genera un salto de línea
- **chown** usuario ruta-fichero: modifica el propietario de un archivo
    - **chown -R** usuario ruta-directorio: modifica propietario de directorio y sus archivos (para que solo lo haga al directorio sin sus archivos sacarle la -R)
- sudo **chgrp -R** grupo ruta-directorio: cambia grupo propietario recursivamente
- **sudo groupadd** grupo: crear grupo
- **sudo adduser** usuario: agrega usuario
    - **sudo passwd** usuario: cambia clave
- **sudo adduser** usuario grupo: asociar usuario al grupo
    - s**udo usermod - g** grupo usuario: usar este para cambiar grupo por defecto
- **who**: devuelve los usuarios conectados al sistema
    - **whoami**: devuelve mi usuario
- **free**: Devuelve info sobre memoria

A; B: Ejecuta B tanto si A salido bien o no
A && B: Ejecuta B si y sólo si A salido bien
A || B: Ejecuta B si y sólo si A salio mal
A &: Ejecuta A en background

STDIN: contenedor de entrada
STDOUT: contenedor de salida. la salida por defecto es imprimir por pantalla
STDERR: contenedor de salida de error

Variables de entorno:

variables de entorno: modifican el funcionamiento de los procesos en su ejecución. son globales para todo el entorno.

- [**comando] &:** ejecuta el comando en back y devuelve el número de PID
- **export** DATABASE_HOST=localhost : crea una variable de entorno(solo valida para la sesión)
- **unset** DATABASE_HOST: borra la variable de entorno
- echo “el valor de la variable es $HOME” : imprime el mensaje con el valor de la variable
- **HOME=/tmp ./script.sh:** corre el script cambiando el valor de la variable de entorno solo para la ejecucion de ese script

- **nano .bashrc:** archivo de inicio de sesión (se configura comandos que se inician al iniciar sesión)

<BUSCA DENTRO DEL ARCHIVO (devuelve las líneas que cumplan)

- **grep** [filtro] [archivo]: filtra dentro del archivo buscando por el filtro si es que encuentra ese filtro. caseSensitive
  - **cat [archivo] | grep [filtro]**: hace lo mismo que arriba
  - **grep -e** [filtro] [archivo]: filtra y devuelve la línea
  - **grep -i** [filtro] [archivo]: filtra ignorando mayúsculas o minúsculas
  - **grep -v** …: filtra todos los que no tengan el filtro dado
  - **grep -w** ... : filtra si cumple con el filtro de manera seguida (USAR ESTE PARA BUSCAR UN USUARIO EN PASSWD)
  - **grep -n** …: te agrega al principio de línea de respuesta, el numero de linea
  - **grep -c** ...: Cuenta las líneas de los que cumplen el filtro y te da el número
  - **grep -r** [filtro] [directorio]: Buscará recursivamente dentro de los archivos del directorio
  - **grep -r** [filtro] [directorio] **2>/dev/null**: lo mismo que arriba pero las que no estén disponibles las manda a null
  - **grep -i [^filtro]** [archivo]: busca según empiecen con el filtro (grep -i “^if” ejer1.sh)
  - **grep -i** [filtro$] [archivo]: buscan líneas que finalicen con el filtro (grep -i “filtro$” ejer1.sh)

<BUSCA ARCHIVOS (devuelve lista de ubicaciones de archivos)>

- **find** [directorio] **-type f -name** “passwd”: busca archivos filtrados con nombre “passwd” dentro de un directorio. recursivamente por default
  - find [directorio] -**maxdepth 2 -type f -name** [filtro]: buscará con máxima prioridad según el número (ejemplo 2)
  - find [directorio] -**mindepth 2 -type f -name** [filtro]: lo mismo pero en minimo
  - find [directorio] **-type f -name** [filtro] **-exec** [comando] **{} \;** : ejecutara el comando para cada resultado que encuentra el find. el “{}” simboliza el resultado, y se debe usar en los comandos. por ejemplo cp {} [direccionDest]
  - find [directorio] **-type f -name** [filtro] **-delete**: Borra lo que encuentra
  - find [directorio] **-perm 644**: devuelve el contenido del dir que cumpla con esos permisos

- find [directorio] **-type f -iname** “a*”: filtra archivos que empiecen con a
- find [directorio] **-type f -name** “*.txt”: filtra archivos con extensión .txt
- find $1 **-type f -size** +1k: filtra por archivos con mayor tamaño de 1k
- find $1 **-type f -mmin** -30 : filtra archivos modificados hace 30 min
- find $1 **-type f -used** -30: filtra archivos accedidos hace 30 min
- find $1 **-type f -mtime** +10: filtra archivos modificados hace más de 10 días
- find $1 **-type f -ctime** +10: filtra archivos creados hace más de 10 días
- find $1 **-type f -atime** +10: filtra archivos abiertos hace más de 10 días
- find $HOME/practicas/tp2 **-type f -executable -exec** cp **{}** execu **\;** :filtra ejecutables y los mueve a una nueva carpeta
- find $HOME/practicas/tp2 **-type f -not -executable -exec** cp **{}** no-execu **\;** : lo mismo de arriba pero no ejecutables (find $HOME/practicas/tp2 -type f -not -executable -exec cp {} $HOME/practicas/tp3/not-execu \;)
- find $1 **-type f -iname** "*.conf" -exec basename {} \; :imprime solo los nombres

tipos de ejecución:

- explícita (**bash** [script].sh)
- implícita con **./[**script].sh (requiere permiso de ejecución)
- implícita con ejecutable: poniendo **./**[dirección].. el punto indica donde estas parado

Permisos:

r → lectura → 100 (4)
w → escritura → 010 (2)
x → ejecución → 001 (1)

Dueño → u
grupo → g
otros → o

estructura:

-	rwx	rw-	r--	1	admi	admi	0	may 12 22:33	archivo1
	Dueño	Grupo	Otros	Enlaces	Dueño	Grupo	Tamaño	Fecha y hora	nombre
	111	110	100						

ejemplo: -rwxr-x--x = 761 -rwx-wx--x: 731 -rwxrwxrwx:777

Declaración de variables:

(SIEMPRE SIN ESPACIOS)

- a=”directorio”
- cont=0
- c=$(echo $a) //los comandos embebidos siempre van con “$()” (NO DEJAR ESPACIOS CUANDO ABREN Y CIERRAN PARÉNTESIS “(comandos)”)

Llamadas a variables:

- **$a , $b** .
- **“$a”, “$b” :** En strings
- Para operaciones numéricas se usa “ let ”
  let var1=var2+3 (sin usar los $)

Ingreso de datos:

- **read** a , **read** b

Parámetros:

- **$1,$2,$3**,...hasta $9 (se corresponden por posición)
- **$0**: contiene el nombre del script
- **$#**: contiene el número de parámetros con el que fue invocado
- **$$:** número de proceso
- **$***: contiene todos los parámetros de 1 al 9 (separados por un espacio)

para pedir por un parametro en particular

if [! $3]; then
fi

Comparadores:

- **Numérico**
  - [$a **-eq** $b] (igual)
  - **-ne** (no es igual)
  - **-gt** (mayor)
  - **-lt** (menor)
  - **-ge** (mayor o igual)
  - **-le** (menor o igual)

- **Texto**
  - **=** (igual)
  - **!=** (no igual)
  - **\>** (mayor)
  - **\<** (menor)
  - **-z** $a (es nulo)
  - **-n** $a (no es nulo)
  - if **[[$1 == doc*]];** then … : filtra por los que comienzan con “doc” (No poner comillas)
  - if **[[$1 == *.doc]];** then … : filtra por los que finalizan con “.doc”
  - if **[[$i =~ "[filtro]"]];** then ....: que contenga el filtro

- **Fichero**
  - -e “$archivo” (si existe)

- -f “$archivo” (si es archivo)
- -d “$archivo” (si es directorio)
- -s “$archivo” (si no tiene tamaño cero)

- **AND**
  - [“$1”] **&&** [“$2”]
- **OR**
  - `if [ "$fname" = "a.txt" ] || [ "$fname" = "c.txt" ] //OR`

Para corroborar que el parámetro exista y que no sea nulo
hay que poner:

```
if [ ! -e $1 ] || [ -z $1 ]; then
    exit
fi
```

Para no tener un numero de parametros definidos se debera recorrer cada parametro

for item in $*; do
    //… devolvera $1, despues $2, despues $3...
done

Estructuras condicionales:

- **IF:**
  - **if [condición]; then** (respetar espacios)
    …
    else
    …
    fi
  - **if** $b (si se creo correctamente el directorio) // if ! $1 (si no se pasó el primer param)
    …
    then
    else
    …
    fi

- **FOR:**

- for i **in** 1 2 3 4 5 6**; do** // **for** i **in {**1..28**}; do** //**for (**i=1; i<=6; i++**); do** //**for** i **in $(seq** 1 15**); do**
...Utilizando la herramienta perf, analice la ejecución de los scripts

done

for i **in** directorio**/*; do** (muestra rutas) //**for** i **in $(ls** directorio**); do** (muestra solo nombres)

...
done

- **WHILE:**

  - **while [** condición **]; do** (respetar espacios)
    ...
    done

  - **while read** linea**; do** (recorre cada linea del archivo “archivo1”)
    echo $linea
    done<archivo1

- **CASE:**

  - **case** $op i**n**
    1)
    ....
    bash [nombre].sh //se pòne esto para que vuelva a ejecutar el script
    ;;
    2)
    ....
    bash [nombre].sh
    ;;
    ***)**
    echo “salir” //sale del script ya que no lo llama de vuelta
    ;;

    esac

Archivos importantes:

- /**etc/passwd:** lista usuarios del sistema
  (nombreusuario:contraseña:idDelUsuario:IdDelGrupo:nombreUsuario:RutaHome:shell)
- **/etc/fstab**: contiene los dispositivos
- **/etc/group:** info de los grupos

Para obtener los nombre y shell de cada usuario →

for item in $(cat /etc/passwd | grep -w "bash" | cut -d":" -f1,7); do
done

shell o bash son un interpretador de comandos que te deja comunicarte con el SO

para crear scripts:

archivo**.sh -->**

#!/bin/bash
… codigo