

LINUX

Sumario

Administrar texto.....	2
Alias.....	4
Atajos.....	4
Comprimir y descomprimir.....	5
Crear y borrar directorios y archivos.....	5
Crontab:.....	5
Espacio en disco:.....	6
Enlaces:.....	6
Find.....	6
File.....	7
Gestión de usuarios.....	7
LDAP.....	7
Listar archivos.....	7
Puertos.....	8
Permisos.....	8
Procesos.....	9
Servicios.....	9
Red y parámetros del sistema.....	9
Directorios y archivos importantes.....	11
Volúmenes y gestión de discos:.....	12
Redireccionamiento.....	13
Vim:.....	13
Tmux.....	14
Otros.....	14
Instalación.....	15
Bash Scripting.....	15

Administrar texto

Awk:

Sirve para mostrar un texto específico de salida.

Para mostrar la línea que queramos. Ejemplo: `awk "NR==3"`

`awk '{print $1} : Me muestra el primer argumento de cada línea`

`awk 'NF{print $NF}' : Me muestra el último argumento de cada línea`

Con `substr(cadena, inicio, longitud)` extraemos una subcadena de texto. Es decir podemos coger las letras que queramos de cada palabra.

Ejemplo:

```
awk '{for(i=1;i<=NF;i++) print substr($i,1,1)}' | tr -d '\n'
```

Recorremos cada palabra con el `for` y luego con el `substr` elegimos la primera letra de cada palabra, luego quitamos los espacios.

Awk es muy extenso y tiene muchas más funciones.

Base64:

Sirve para codificar y decodificar archivos en base64

Codificar palabras: `echo -n "palabra" | base64`

Codificar archivo: `base64 "archivo"`

Decodificar archivo: `base64 --decode archivo.txt`

Cut:

Me corta la salida de un comando, se usa siempre con una opción, tiene usos muy útiles:

Ejemplo: `cut -d ":" -f 1,2,3 texto.txt`

Le indicamos el delimitador con `-d ":"` y que muestre las columnas 1,2 y 3 con `-f`.

Podríamos mezclarlo con `sort` para que nos ordene la salida:

```
cut -d ":" -f 1,2,3 texto.txt | sort
```

Podríamos mezclarlo con `grep` para que nos busque solo ciertos nombres:

```
cut -d ":" -f 1,2,3 texto.txt | grep Paco | sort.
```

Diff:

Sirve para diferenciar dos archivos.

Grep:

Filtra por un patrón de caracteres y despliega su contenido.

Podemos usarlo junto con `ls` para listar ficheros que contengan una palabra en su nombre:

`ls -lia | grep script`: Me listarían todos los ficheros que tienen script en su nombre.

Para no tener que hacer `cat data.txt | grep "hola"`, podemos hacer `grep "hola" data.txt` en archivos.

Añadiendo `^` obligamos a que la palabra empiece por eso, con `$` le decimos que termine con eso

Opciones:

-A: Te muestra x líneas por debajo de lo que digas (B para arriba)

-c: Número de líneas que contiene el patrón.

-E: Filtrado múltiple. Ejemplo: `grep -E "Entrada total|Salida total"`

-n: Muestra las líneas que coinciden y su número.

-l: En una carpeta, te dice los ficheros que contienen ese patrón.

-v: Te muestra las líneas que no contienen lo que pongas

Head y Tail:

Me muestra las primeras o últimas líneas

-n: Indicamos las líneas que queremos (con num negativo quitamos las últimas o primeras)

Ejemplo: `head -n -2` (Nos quita las dos últimas líneas)

Rev:

Sirve para revertir una cadena de texto.

Sed:

Sirve para modificar texto normalmente, lo más usado es con s:

`sed 's/perro/gato'` Con g al final me lo aplica a toda la línea
(Normalmente es solo al primer resultado)

`sed "3d" distros-deb.txt > distros-deb-ok.txt` Borra la línea 3

Si el sed es con variables se pone con comillas dobles.

Sort:

Sirve para ordenar la salida de un comando:

Por ejemplo un texto que tenga las líneas desordenadas con `sort texto.txt` mostramos las líneas ordenadas alfabéticamente. También podemos hacerlo con varios archivos a la vez:

`sort texto.txt texto2.txt`

Opciones:

-o: Redirige la salida a un fichero `sort -o textordenado.txt texto.txt` (Sería lo mismo que hacer `sort texto.txt > textordenado.txt`)

-c: Mira si el texto está ordenado y te avisa si no.

-r: Ordena al revés

Strings:

Busca caracteres imprimibles en un archivo. Ej: strings archivo

Uniq:

Elimina las líneas repetidas en un archiv y lo muestra. (Solo funciona si las líneas están juntas. A veces conviene hacer primero un sort. Ejemplo: sort data.txt | uniq -u)

Uso: `uniq archivo.txt`

-D: Imprime las líneas repetidas.

-u: Imprime las líneas no repetidas. (unique)

Tr:

Sirve para modificar unos caracteres por otros en un texto. Ejemplo: `tr '[a-z]' '[n-za-m]'`

Con -d podemos quitar caracteres: `tr -d \n` me quita los saltos de línea

Wc:

Sirve para contar las palabras en un texto.

-l: Cuenta las líneas

-c: Cuenta los caracteres

Xxd:

Pasa un archivo de Ascii a hexadecimal

Otros comandos son:fmt, fgrep, egrep

Alias

Comando: `alias comandocorto="comandolargo"`

Atajos

Ctrl+A: Inicio de comando

Ctrl+E: Fin del comando

Ctrl+C: Detiene el comando de forma segura

Ctrl+R: Histórico comandos con lo que indroduzcas

Ctrl+Flechas: Salta por las palabras

Ctrl+U: Quita lo que hay antes del cursor

Ctrl+K: Quita lo que hay después del cursor

Ctrl+Z: Fuerza la detención del comando (lo manda a segundo plano)

!& en línea de comandos: Referencia al último argumento

Comprimir y descomprimir

Con tar conservamos el metadata además de comprimir el archivo, también funciona mejor para directorios recursivos

Zip:

Comprimir: `zip archivo`

Descomprimir: `unzip archivo`

Tar:

-f para el nombre del archivo, -v verbose (más detallado), -x para extraer

Comprimir: `tar -cvf archivo`

Descomprimir: `tar -xvf archivo`

Gzip (extensión .gz):

Comprimir: `gzip archivo`

Descomprimir: `gzip -d archivo` o `gunzip archivo`

Crear y borrar directorios y archivos

Crear: `mkdir directorio`, con `mkdir direct(1,2,3)` creamos `direct1` `direct2` y `direct3`

`touch file.txt`, creamos `file.txt` vacía. Con `rmdir` borramos directorios vacíos.

`Cd` a secas te lleva a birgulilla.

Crontab:

Gracias al crontab podemos automatizar tareas en linux.

El crontab del sistema se encuentra en `/etc/crontab` y el de los usuarios en `/var/spool/cron`

Con `crontab -e` creamos un crontab y con `crontab -l` vemos los cronjobs existentes.

El formato al añadir tareas es:

* * * * * El asterico significa se ejecuta en todos + el comando que queremos

1: Minutos 0-59 > */5 = Cada 5 minutos

2: Horas 0-23 > 3-6 = Entre las 3 y las 6

3: Día 1-31 >

4: Mes 1-12 > */2 = En los meses que son divisibles por dos: Febrero...

5: Nombre del día 0-6 0=Domingo > 0,6 = Sábado y Domingo

Ejemplo: 0 0 * * 0 echo "Viva Asturias" >> /home/mcm/Asturias.txt = A medianoche los domingos añade Viva Asturias al fichero Asturias.txt

Espacio en disco:

Uso del espacio de disco: `df`

Para que no muestre el espacio en kbytes y sea más amigable usamos el parámetro más usado:
`df -h`

Para ver cuanto espacio de disco ocupa un directorio: `du Descargas`

Aquí el `-h` sirve para lo mismo: `du -h Descargas`

Para que no nos aparezcan los subdirectorios usamos `-s`: `du -hs Descargas`

También podemos mostrar el tamaño de cada archivo: `du -a`

La podemos canalizar en orden para ver una lista ordenada de archivos, el parámetro «-n» le dice al comando `sort` que considere la primera columna de números en la salida de `du` como una cadena numérica: `du -a Descargas | sort -n`

Enlaces:

Son útiles para crear accesos directos y poder crear varias ubicaciones para las carpetas.

Para crear el enlace usamos `ln`, para que se borre el archivo necesitaremos borrar todos los enlaces.

Se puede mirar el número de enlaces que tiene con `stat link.txt`

Con `-s` le indicamos que es simbólico. Se rompen si se borra el archivo original.

Ej: `ln -s archexiste nombrenlacesim`

Podemos comprobarlo haciendo `ls -l` en la carpeta donde hemos creado el enlace.

Para eliminar los enlaces existentes usamos `unlink enlace`, aunque también podemos hacerlo con `rm enlace`.

Find

Comando para buscar: `find <dondebuscamos> <parámetros>`

- type: Por tipo: f fichero d directorio l enlace
- name: Por nombre fichero, -iname no distingue minúsculas y mayúsculas
- size: Por tamaño: Bytes c, Kilobytes k, Megabytes M, Gigabytes G. Ejemplo: +500M, -500M o 500M.
- cmin: Ve según los hayamos creado -5, hace menos de 5 min o +5 hace más. La c es creación, m modificación y a acceso.
- user -group -perm: Busca por propietario. Usuario, grupo y derechos de acceso.
- empty: Busca archivos vacíos.
- !: Busca al revés -executable

File

Sirve para ver el tipo de un fichero: file fichero.txt

Gestión de usuarios

Creamos usuarios con adduser o useradd. Los borramos con deluser. (Para grupo cambiamos por group)

Los usuarios los modificamos con usermod y los grupos con modgroup.

Con deluser --remove-all-files user

LDAP

Protocolo Ligero de acceso a directorios, se utiliza para la autenticación de servicios de directorio. Lo utilizan el correo electrónico y otros programas para buscar información en un servidor.

Comandos más usados:

ldapsearch: Para buscar

ldapadd: Para añadir usuarios

Ldapdelete: Para borrar usuarios

Listar archivos

Usamos el comando: ls

Para mostrar como una lista: ls -l

Para ordenar archivos por tamaño usamos

-a: Para mostrar archivos ocultos: ls -la

-r: Para invertir el orden usamos la: ls -lr

-t: Para ordenar por hora y fecha, se usa con la -r para ver los archivos más nuevos abajo: ls -ltr

-i: Para mostrar los inodos de los archivos o subdirectorios: `ls -lia`

-s: Para mostrar el tamaño en kb de los archivos

-R: Para listar también los directorios recursivos usamos: `ls -lR`

Para ver de manera más clara el tamaño de los archivos usamos -h: `ls -lh`

Para saber quién creó el archivo usamos `--author: ls --author -l`

Para mostrar el suid y el sgid usamos -n: `ls -n`

Puertos

Abrir: `ufw allow NPUERTO`

Cerrar: `ufw deny NPUERTO`

Para rango de puertos: `ufw deny NPUERTOINI:NPUERTOFIN/tcp` (si es necesario ponemos protocolo)

Puertos comunes número y para qué se usan:

[List of TCP and UDP port numbers - Wikipedia](#)

Permisos

Comando: `chmod 777 archivo`

Posición números: Usuario, grupo, otros.

Valores: 4 read, 2 write 1 ejecutar(x). Los sumamos para el número final.

-R para recursivo

Hay tres permisos especiales:

Sticky Bit (T en others): Solo pueden ser borrados o renombrados por el propietario del archivo, del directorio o el usuario root aunque el resto tengan permisos de escritura.

Añadiendo un 1 en el octal antes Ej: `chmod 1777 archivo`

Sgid (S en group): El usuario que ejecute el fichero tendrá durante la ejecución los mismos permisos que el grupo propietario

Añadiendo un 2 en el octal antes Ej: `chmod 2777 archivo`

Suid (S en owner): El usuario que ejecute el fichero tendrá durante la ejecución los mismos permisos que el usuario propietario

Añadiendo un 4 en el octal antes Ej: `chmod 4777 archivo`

Procesos

Ps:

`Ps aux:` Ver procesos del sistema

Los vemos con el comando `ps`. La información que vemos es:

`PID` - Número del proceso.

`TIME` - Tiempo que lleva corriendo el proceso.

`TTY` - Nombre de la terminal que controla el proceso.

`CMD` - Nombre del comando con el que se inició.

Normalmente se usa `ps aux`, donde le decimos que lo muestre más detallado.

Para matar los procesos hacemos `kill -9 PID`, con `kill -1 PID` les hacemos reload.

Para correr un proceso en segundo plano le ponemos `&` al final. Lo enviamos a primer plano con `fg` (el último), sino hacer `fg nombreproceso`.

Wait:

Espera a que un proceso se termine de ejecutar (Útil en scripts)

Ejemplo: `wait 32`

Otros comandos son `top`, `htop`, `atop` `isof`, `nmon`, `iostat`, `sar`, `vmstat`.

Servicios

Comando: `systemctl (start,status,stop,restart,reload,enable o disable) nombreserv`

Con `reload` lo recargamos pero no lo reiniciamos del todo, con `enable` hacemos que se inicie en cada arranque.

Otra opción: `service nombreserv (start,stop,restart)`

Red y parámetros del sistema

SSH:

El archivo de configuración está en `.ssh/config`

Para conectarse a una máquina desde otra: `ssh usuario@IP(o nombre maq)`

`ssh-copy-id usuario@IP`: Con esto copiamos la clave para que en la próxima no nos pida contraseña

`ssh -t usuario@IP comando` :Con esto ejecutamos comandos directamente en la máquina

`ssh -D 9999 usuario@IP` :Crea un proxy

`ssh -X usuario@IP` :Me conecto y si por ejemplo ejecuto firefox, lo muestra en mi máquina.

`ssh -L Puertomimaq:IpMaq2:Puertomaq2 usuario@IPmaq` :Pegas 2 saltos de golpe si la segunda tiene firewall

`ssh -R 2020:localhost:22 usuario@externo` :Haces un ssh “al revés”.

Luego en tu maq haces `ssh root@localhost -p2020` y te conectas a la otra máquina

SCP sirve para pasar archivos y usa ssh (también existe rsync).

`Scp file.txt usuario@externo:/root/`

FTP:

Se usa para transferir archivos de una máquina a otra: `ftp`

`-p`: Puerto

`Ifconfig`: Ve config de interfaces

`Ip addr`: Ve IP e interfaces

`Hostname`: Ve nombre de host y dominio, `-i` y `-I`

Uname:

Sirve para ver parámetros del sistema. Opciones:

`-n`: Nombre del host

`-r`: Versión kernel

`-m`: Arquitectura de hardware

Traceroute:

Sirve para ver todos los pasos en una conexión de red.

Ejemplo: `traceroute tryhackme.com`

Para cambiar interfaz usamos `-i`

Ping:

- q: Lo hace quiet
- c 2: Lo hace 2 veces

Directorios y archivos importantes

/boot:

Carpeta del cargador de arranque, necesaria para iniciar la máquina.

/dev:

Archivos de los dispositivos conectados al sistema, por ejemplo los discos duros conectados al sistema

Allí nos encontramos archivos como:

/dev/null: Un sistema de archivos que no existe, lo que entra ahí se pierde para siempre. Se usa para redirigir salidas de comandos que no necesitamos.

/etc:

Archivos de configuración de programas propios del SO y de programas instalados por el usuario:

Allí nos encontramos archivos como:

/etc/hosts: Mapea los nombres de los equipos con las correspondientes IPS

/etc/hostname: Contiene el nombre de la máquina

/etc/passwd: Contiene información de los usuarios (nombre,UID,GID...)

/etc/crontab: Sirve para automatizar tareas en el sistema

/etc/resolv.conf: Sirve para resolver los nombres DNS

/home:

Donde los usuarios almacenan sus datos y archivos de configuración.

/root:

Directorio del superusuario administrador root.

/tmp:

Carpeta donde se almacenan archivos de forma temporal. Los archivos que no se usen en 10 días serán borrados automáticamente. También existe /var/tmp, que es lo mismo pero con 30 días.

/usr:

Contiene software instalado, librerías compartidas y datos de programa de solo lectura.

Allí nos encontramos archivos como:

/usr/bin: Comandos de usuario.

/usr/sbin: Comandos de administración del sistema.

/usr/local: Software local modificado.

/var:

Datos variables que necesita el sistema cada vez que arranca. Allí se pueden encontrar normalmente logs, archivos que cambian dinámicamente, caché de webs...

/run:

Datos de los procesos iniciados desde el último inicio. Antiguamente era /var/run

Volúmenes y gestión de discos:

Un volumen lógico se compone de particiones lógicas asignadas a discos físicos (Donde se almacenan los datos en un LVM)

pvcreate: Creación de volúmenes LVM

vgcreate: Crea un grupo de volúmenes a través de un disco físico.

Ej: `vgcreate nombregrupo /dev/sdb(Disco1) /dev/sdc(Disco2 si es necesario)`

lvcreate: Crea un volumen lógico

Ej: `lvcreate -L espacioengigasG -n logicvolumename logicvolumegroup`

lsblk:

blkid:

`pvs -a:` Ver la información de los discos físicos.

Montar, desmontar y remontar discos:

Desmontar: `umount /home/projects`

Montar: `mount -o remount`

Para activar cuotas:

`quotaon -vu /home/backups`

`quotaon -vg /home/projects`

Redireccionamiento

Normalmente la salida de los comandos se muestra por pantalla (tanto la salida normal como los errores), para cambiar esto podemos:

Redireccionar salida estándar a un fichero: `ls -l > /home/salida.txt`

Con dos `>>` creamos también el fichero si no existe: `ls -l >> /home/salida.txt`

Para la salida de errores usamos `2>` creará el fichero si ya existe: `ls -l 2> /home/salida.err`

Si queremos enviarlo a un fichero que ya existe usamos `2>>`: `ls -l 2>> /home/salida.err`

Si queremos enviar toda la salida a un fichero: `ls -l &> /home/salida.txt`

Para añadirlo a un fichero que ya existe: `ls -l &>> /home/salida.txt`

Con esto le pasamos un sort al primer fichero mandamos la salida al segundo (tienen que ser ficheros diferentes):

```
sort < /home/salida.txt > /home/salidaordenada.txt
```

Con tee lo vemos en la pantalla y también lo envía al fichero:

```
ls | tee salida.txt
```

Recordamos que si no queremos la salida lo podemos redireccionar a `/dev/null`

Vim:

Editor de texto de linux, para editar archivos hacemos `vim archivo`, para ir a una línea directamente `vim archivo +num`, para ir directamente a una palabra `vim archivo +/Palabra`.

Para entrar en modo insertar pulsamos `a` o `i`, para salir de él pulsamos `esc`.

Comandos:

Borramos una línea con `dd`, una letra con `d l` y una palabra con `dw`.

Para guardarlo con otro nombre escribimos : `w nombre.txt`

G: Final de fichero

B: Inicio de línea

gg: Inicio de fichero

\$: Con esto vamos al final de línea

Esc+U: Deshacemos cambios

Ctrl+R: Rehacemos cambios

Para cortar texto, en modo comando hacemos `cc` para línea `cw` para palabra `cl` para letra.

Con `y` copiamos. Para hacer esto con varias líneas pulsamos en número de líneas antes.

Con la p pegamos texto

Para poner una línea debajo de otra en modo comando usamos:

```
:g/key/norm owhat ever you want
```

Para comparar dos archivos usamos `vim -d archivo1 archivo2` y con `ctrl+w w` nos movemos de un fichero a otro. Con `D+O` y `D+P` copiamos una línea de un archivo a otro.

`:r`: Con esto se le indicamos un archivo lo copiamos al archivo abierto.

También se pueden abrir archivos zip en vim, pulsando enter para entrar en cada uno de ellos.

Para ir a un archivo que está referenciado en otro podemos pulsar `gf` sin salir de vim.

Para correr comandos externos usamos `:!date`. No vale para hacer `:r !ls`, con lo que copiamos el `ls` de la ruta que estemos al archivo abierto.

`:help user` para abrir la ayuda de Vim

Tmux

Sirve para poder correr dos terminales a la vez.

`Tmux new -s "Sesion"`: Nueva sesión

`Ctrl+B` y luego `"` : Divide verticalmente (% horizontalmente)

`Ctrl+B` y luego `x` : Elimina el panel

`Ctrl+B` y luego espacio: Rota de horizontal a vertical

Otros

Xargs:

Sirve para leer la salida de un comando y pasárselo a otro.

Ejemplo: `find . -type f | xargs file`

Time:

Al principio de un comando vemos lo que tarda.

Disown:

(-a opcional) Sirve para independizar tareas de la terminal. Ejemplo: `disown firefox` (Si cierro la terminal no cierra el firefox)

Curl: Me saca el contenido de una página web: (`htmltotext` me lo convierte a legible)

Which: Me dice si un programa está instalado o no

Al hacer un cat a un archivo y modificarlo con tuberías no se puede redirigir al mismo archivo. Lo redirigimos a otro y redirigimos el otro al primero.

Instalación

Dependiendo de las versiones se usa una utilidad u otra:

En Ubuntu y Debian: `apt install paquete`

En SuSE o RedHat: `zypper install nombre_programa`

En Fedora o CentOS: `yum install paquete`

Bash Scripting

Los scripts de bash tienen de extensión `.sh`

El script siempre lo empezamos con `#!/bin/bash`. Acordarse de hacer ejecutable el archivo. Lo ejecutamos con `bash hola.sh` o con `./hola.sh`. Las variables van sin espacios.

Recordatorio sentencias: `if, then elif, else; while,do,done; until` (contrario `while`)

`for, in`: Va recorriendo hasta que salga. También usa `do` y `done`. Ejemplo: `for cups in {1..10};`

`$(cat españa.txt)`: Le indicas con el dollar que es un comando, lo podemos añadir a un `for`. (Con dos paréntesis es con función matemática)

Ejemplo: `for x in $(cat españa.txt)`

`read`: Pide una variable y la lee

`-p`: Pide enter para continuar un `while`

`Break` y `continue`: Se usan en loops, `break` sale del bucle y `continue` pasa a la siguiente vuelta y se olvida de esa.

`sleep 2`: Se usa para esperar x segundos (En este caso 2)

`$()`: Para mates dentro

```
case $class in
    1)
        type="Samurai"
        hp=10
        attack=20
        ;;
```

Variables de entorno:

`$RANDOM`: Elige un número aleatorio entre 0 y 32747 (para poner otro `% 2`)

`$USER`: Devuelve el usuario

`$PWD`: Devuelve el directorio de trabajo

\$HOSTNAME : Devuelve el nombre de la máquina

\$SHELL: Devuelve la shell que estás usando

Crear variable de entorno:

twitter="Elon Musk" , la exportamos con export twitter

Para hacerla permanente editamos .bashrc: poniendo export twitter="Elon Musk"

Matemáticas:

En bash, (no solo para scripts), para hacer operaciones matemáticas usamos dos paréntesis. Si queremos mostrarlo tenemos que hacerle un echo $$(2+2)$

* : Multiplica / : Divide % : Da el resto de una división

Input y Output:

0: Estándar input 1: Estándar Output 2: Estándar error

exit 0 o 1: Sale del script con código de éxito (2 para error)

Para pasar los errores a salida normal: 2>&1

Opciones en if:

- gt: Comprueba si un número es más grande que x (lt al revés)
- f: Comprueba si el argumento mandado es un archivo que existe y es un fichero.
- d: Lo mismo con un directorio.
- h: Lo mismo con un enlace simbólico.
- s: Comprueba si el archivo existe y no está vacío.
- r: Comprueba si el archivo es legible.
- w: Comprueba si el archivo tiene permisos de escritura.
- x: Comprueba si el archivo es ejecutable.
- nt: Comprueba que archivo es más nuevo, (-ot lo hace a la inversa).
- <: Comprueba que el primer argumento está alfabéticamente antes que el segundo (> al revés).
- !=: Comprueba que no sean iguales los dos argumentos.
- || : Es un OR en un if
- && : Es un AND en un if

Otras:

Opciones en while:

-le: Menor que algo. Ejemplo: while [[\$x -le 100]]

cat /etc/passwd | while read line; do


```
echo line
```

```
done
```

Me mostraría todas las líneas del fichero.

```
while read line; do
```

```
    echo line
```

```
done < /etc/passwd
```

Esto es lo mismo.

Let contador+=1: Suma 1 por cada paso a contador

Echo -e: Habilita la interpretación de caracteres tras \:

\a : Alerta

\e \E : Escape

\n : Nueva línea

\t : Tabula horizontalmente

\v : Tabula verticalmente

\$1: Hace referencia al primer argumento pasado al script

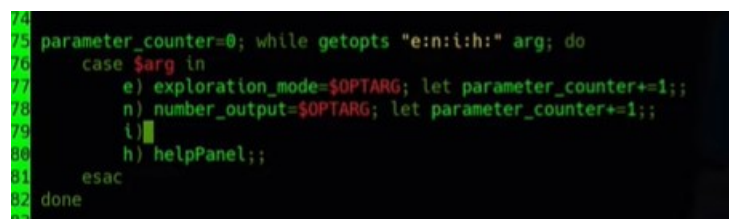
echo -n: No me salta de linea entre ese echo y el siguiente

una variable que va sumando en un bucle, si la lees fuera te vale lo que valía antes del bucle, para eso si queremos ver el valor final lo podemos meter a un fichero con >

```
function hola(){
```

```
    echo "Hola"
```

```
}
```



```
74 parameter_counter=0; while getopts "e:n:i:h:" arg; do
75     case $arg in
76         e) exploration_mode=$OPTARG; let parameter_counter+=1;;
77         n) number_output=$OPTARG; let parameter_counter+=1;;
78         i)
79         h) helpPanel;;
80     esac
81 done
82
83
```

OPTARG sirve para poder pasarle argumentos además de los parámetros. Ejemplo: ./miscript -a parametro1

(Los dos puntos al final, es que si no pones nada, te lleva a la última opción)

Con el while getopts se pueden elegir varios argumentos

```
while getopts "a:b:c:" arg; do
```

```
case $arg in
```

```
    a) funcion1=$OPTARG;
```

b) funcion2=\$OPTARG;

c) funcion3=\$OPTARG;

esac

done

tput cnorm: Para poner el cursor (tput civis para quitarlo)

; Es como si cambias de línea

set -f. Quita el globbing (*? y esos caracteres), nos vale para scripts que nos den fallo si ponemos asteriscos

Al llamar a funciones, podemos llamarlas con una variable al lado y esa será la variable \$1 de esa función¿?

Child Shells:

Una child shell se crea si pones bash en la terminal, no hereda ni alias ni variables. Se puede comprobar que la hemos creado haciendo un ps. Si hacemos Ctrl+E volvemos a la Shell padre.

Subshells:

Con echo \$BASH_SUBSHELL comprobamos si estamos dentro de una subshell o no.

Si lo hacemos entre paréntesis estamos creando una subshell. Las subshells sirven para poder juntar comandos.

Ejemplo: Si hacemos ls -l /tmp; date | wc -l en una shell normal, el wc -l nos va a contar las líneas del segundo comando.

Si hacemos el mismo comando pero los dos primeros en una subshell , el wc -l nos lo hace a los dos comandos (Parecido al paréntesis en mates)

Si yo agrego delante \$, es decir ejecuto \$(date -u), coje la salida de date -u y lo ejecuta como si fuera un comando. Sirve por si quiero hacer echo de varios comandos juntos.