

a) Estudie y explique para qué sirven los comandos **ls**, **cat**, **chmod**, **echo**, **grep**, **cp**, **mv**, **rm** y **wc**. Dé ejemplos de uso. Para el comando **ls** averigüe para qué sirven las opciones **-l**, **-t** y **-a**. De ejemplo de uso para cada uno de esas opciones y la combinación de ellas. Ayuda: use el comando **man**, ej. **man ls**.

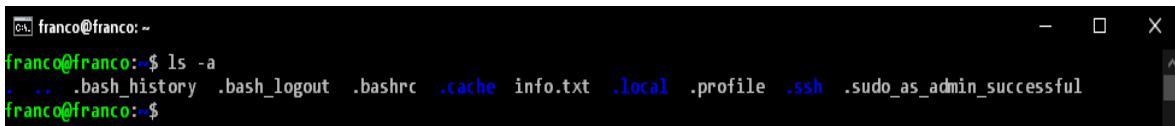
Comando **ls**:

Sirve para visualizar todos los archivos dentro del directorio en el que uno se encuentra. Tiene diferentes opciones, tales como:

- l: Muestra toda la información.
- t: Ordena los archivos por fecha de modificación.
- a: Muestra los archivos ocultos dentro del directorio.
- X: Ordena los archivos por extensión.
- lh: Realiza lo mismo que -l pero agrega el tamaño de los archivos.
- R: Muestra el contenido de todos los subdirectorios de forma recursiva.
- S: Ordena por tamaño de archivo.

Ejemplo de uso:

- `ls -a`



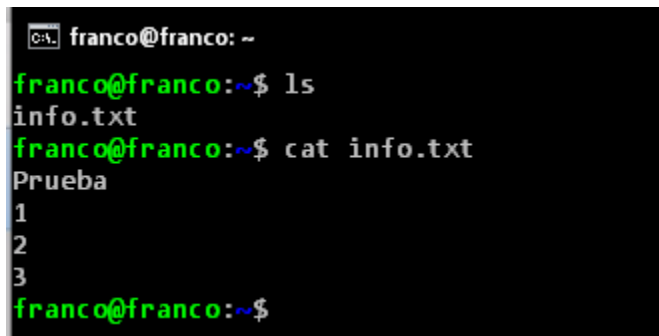
```
franco@franco: ~  
franco@franco:~$ ls -a  
.  ..  .bash_history  .bash_logout  .bashrc  .cache  info.txt  .local  .profile  .ssh  .sudo_as_admin_successful  
franco@franco:~$
```

Comando **cat**:

Permite crear, fusionar, imprimir archivos y algunas otras funciones para el manejo de archivos.

Ejemplo de uso:

- `cat "nombre_archivo.txt"`



```
franco@franco: ~  
franco@franco:~$ ls  
info.txt  
franco@franco:~$ cat info.txt  
Prueba  
1  
2  
3  
franco@franco:~$
```

Comando **chmod**:

Se utiliza para cambiar los permisos de archivos o directorios.

Ejemplo de uso:

- `chmod ug+x "nombre_archivo.txt"`

```
franco@franco: ~  
franco@franco:~$ chmod ug+x info.txt  
franco@franco:~$ ls -l  
total 4  
-rwxrwxr-- 1 franco franco 13 Sep 28 19:34 info.txt  
franco@franco:~$
```

Comando **echo**:

Permite la impresión de texto en pantalla.

Ejemplo de uso:

- `echo "texto"`

```
franco@franco: ~  
franco@franco:~$ echo Prueba  
Prueba  
franco@franco:~$
```

Comando **grep**:

Comando que se utiliza para la búsqueda de una cadena de caracteres (string) en un archivo específico. Una vez encontrado un resultado este se imprime en pantalla.

Ejemplo de uso:

- `grep 'texto' "nombre_archivo.txt"`

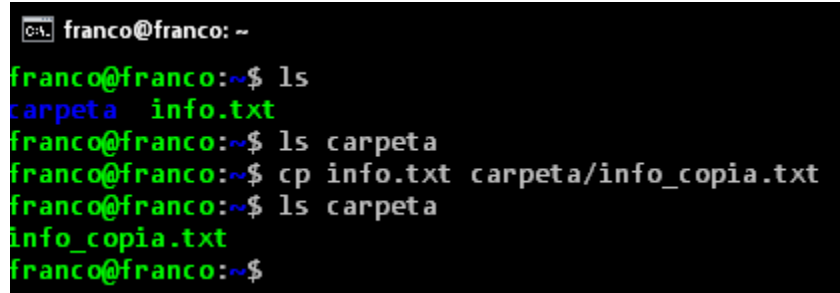
```
franco@franco: ~  
franco@franco:~$ grep 'Prueba' info.txt  
Prueba  
franco@franco:~$ cat info.txt  
Prueba  
1  
2  
3  
franco@franco:~$
```

Comando **cp**:

Sirve para copiar archivos o directorios.

Ejemplo de uso:

- `cp "nombre_archivo.txt" "nombre_directorio"/"nombre_archivo_copia.txt"`



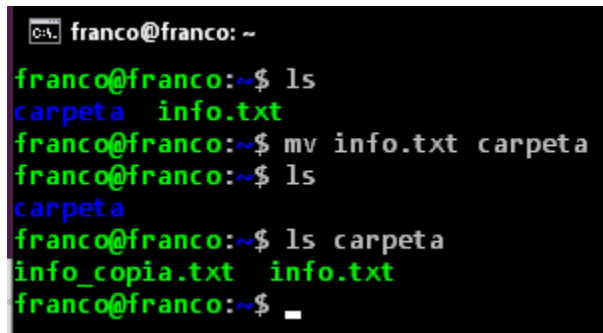
```
franco@franco: ~  
franco@franco:~$ ls  
carpeta  info.txt  
franco@franco:~$ ls carpeta  
franco@franco:~$ cp info.txt carpeta/info_copia.txt  
franco@franco:~$ ls carpeta  
info_copia.txt  
franco@franco:~$
```

Comando **mv**:

Es usado para mover o renombrar archivos o directorios.

Ejemplo de uso:

- `mv "nombre_archivo.txt" "nombre_carpeta"`



```
franco@franco: ~  
franco@franco:~$ ls  
carpeta  info.txt  
franco@franco:~$ mv info.txt carpeta  
franco@franco:~$ ls  
carpeta  
franco@franco:~$ ls carpeta  
info_copia.txt  info.txt  
franco@franco:~$
```

Comando **rm**:

Permite la eliminación de archivos o directorios.

Ejemplo de uso:

- `rm "nombre_archivo"`

```
franco@franco: ~/carpeta
franco@franco:~$ cd carpeta/
franco@franco:~/carpeta$ ls
info_copia.txt  info.txt
franco@franco:~/carpeta$ rm info_copia.txt
franco@franco:~/carpeta$ ls
info.txt
franco@franco:~/carpeta$ _
```

Comando **wc**:

wc es un comando utilizado para realizar conteos, tales como:

- palabras (wc -w)
- bytes (wc -c)
- lineas (wc -l)
- letras (wc -m)

Ejemplo de uso:

```
franco@franco: ~/carpeta
franco@franco:~/carpeta$ more info.txt
Prueba
1
2
3
franco@franco:~/carpeta$ wc -w info.txt
4 info.txt
franco@franco:~/carpeta$ wc -c info.txt
13 info.txt
franco@franco:~/carpeta$ wc -l info.txt
4 info.txt
franco@franco:~/carpeta$ wc -m info.txt
13 info.txt
franco@franco:~/carpeta$ _
```

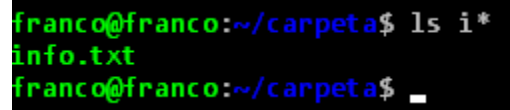
b) Explique qué son los metacaracteres y dé ejemplos de uso de ellos.

R: Los metacaracteres son un conjunto de caracteres especiales que son utilizados para realizar búsquedas u otras operaciones sobre archivos o directorios.

Ejemplos de uso:

- Buscar todos los archivos que comiencen con la letra “i”

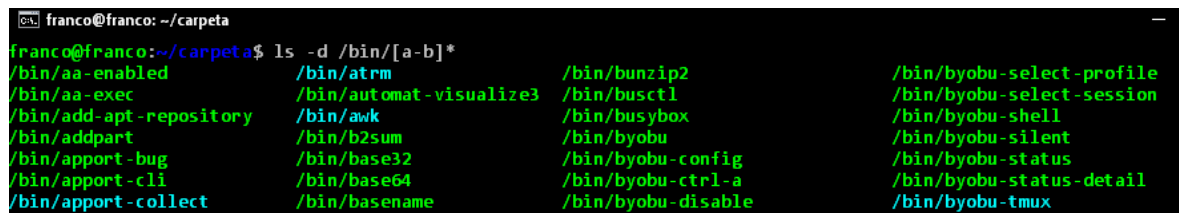
`ls i*`



```
franco@franco:~/carpeta$ ls i*
info.txt
franco@franco:~/carpeta$ _
```

- Buscar los archivos del directorio /bin que comiencen por las letras a y b con cualquier extensión de letras después.

`ls -d /bin/[a-b]*`



```
franco@franco: ~/carpeta
franco@franco:~/carpeta$ ls -d /bin/[a-b]*
/bin/aa-enabled      /bin/atrm           /bin/bunzip2        /bin/byobu-select-profile
/bin/aa-exec         /bin/automat-visualize3 /bin/busctl         /bin/byobu-select-session
/bin/add-apt-repository /bin/awk            /bin/busybox        /bin/byobu-shell
/bin/addpart         /bin/b2sum          /bin/byobu          /bin/byobu-silent
/bin/apport-bug       /bin/base32         /bin/byobu-config   /bin/byobu-status
/bin/apport-cli       /bin/base64         /bin/byobu-ctrl-a   /bin/byobu-status-detail
/bin/apport-collect   /bin/basename       /bin/byobu-disable  /bin/byobu-tmux
```

c) Explique en que consiste la expansión por paréntesis de conjunto (Brace Expansion). Con esta herramienta, resuelve el siguiente problema:

En un directorio, se quieren crear subdirectorios para que almacenen respaldos diarios de todo un año. Debe tener la siguiente estructura :

```
directorio_actual/  
+ 2021-01-01/  
+ 2021-01-02/  
+ 2021-01-03/  
.  
.  
.  
+ 2021-09-18/  
.  
.  
.  
+ 2021-10-18/  
.  
.  
.  
+ 2021-12-31/
```

Suponga que todos los meses tienen 31 días. Debe ejecutar UN sólo comando para crear la estructura de directorios solicitada.

R: La expansión por paréntesis es utilizada para crear o generar cadenas arbitrarias, dicho de otra forma, es utilizada para generar cadenas con la mayor combinación posible con los prefijos y sufijos dados.

Comando a utilizar para crear los subdirectorios: `mkdir {2021}-{01..12}-{01..31}`

d) La interconexión de comandos a través de *pipes* permite construir, de forma muy simple, nuevas herramientas. Como ejemplo considere los comandos `ls` y `wc`, que interconectados permiten contar archivos del directorio actual: `ls | wc -l`. Mediante el uso de pipes resuelva:

1. Mediante `grep`, encontrar archivos cuyo nombre contenga el carácter `i` en el directorio `/bin`.

R: `ls -l /bin/ | grep i`

2. Contar los archivos con una secuencia de permisos `r-x` en los directorios `/bin` y `/usr/bin`.

R: `ls -l /bin/ | grep r-x | wc -l`

e) Las variables de ambiente definen aspectos del entorno de programación, y los comandos `set` y `echo` (mediante el metacaracter `$`) permiten ver su contenido.

1. Investigue el uso de las variables `HOME`, `SHELL`, `PATH`, y `PWD`. ¿Cómo se puede visualizar su contenido?

R: El contenido puede ser visualizado con el comando `echo`, de la forma:

- `echo $nombre_variable`

2. Investigue cómo se puede modificar el valor de una variable de ambiente.

R: Las variables pueden ser editadas con el operador "=", en el caso de PATH esta puede ser editada de la forma:

- PATH=\$PATH:/algo/algo/

Con esto lo que se hace es agregar una ruta /algo/algo, como puede ser "/usr/local/src" para que quede registrado en el PATH.

3. Ejecutando el comando *bash* dentro de la línea de comandos se crea un sub-shell, ¿Cómo afecta esto las variables de ambiente? ¿Cuál es el efecto de export? Explique y dé ejemplos.

R: No afecta a las variables de entorno dado que los cambios que se provocan en un "sub-shell" no son guardados, son válidos mientras el "sub-shell" esté en ejecución. Para este caso las variables de entorno se transforman en variables regulares de bash, es por esto por lo que no se ven afectadas.

En el caso del "export", si una variable es importada dentro de un "sub-shell" esta se transforma en una variable de entorno, por lo tanto, en este caso particular si se ven afectadas las variables de entorno ya que se estaría agregando una al conjunto de variables.

f) El intérprete de comandos bash es también un lenguaje de programación, con estamentos de control de flujo como for, while, etc. El código fuente a menudo se llama *script*. Si el archivo que contiene el script se llama **ejemplo.sh**, el comando **chmod +x ejemplo.sh**, le da permisos de ejecución al archivo **ejemplo.sh**.

Implementar un script BASH que liste cada argumento de entrada por separado, incluyendo el nombre del script. Además, debe mostrar su PID y mostrar las 10 primeras líneas del archivo /proc/PID/status.

R: Código realizado

```
franco@franco: ~  
franco@franco:~$ more code.bash  
#!/bin/bash  
  
echo "Nombre del archivo: $0"  
for i in "$@"  
do  
    let num=$num+1  
    echo "Argumento número $num: $i"  
done  
echo "Información del proceso (10 líneas):"  
more /proc/$$/status | head -10  
franco@franco:~$
```

Output del código

```
franco@franco: ~  
franco@franco:~$ bash code.bash arg1 arg2 arg3  
Nombre del archivo: code.bash  
Argumento número 1: arg1  
Argumento número 2: arg2  
Argumento número 3: arg3  
Información del proceso (10 líneas):  
Name: bash  
Umask: 0002  
State: S (sleeping)  
Tgid: 1111  
Ngid: 0  
Pid: 1111  
PPid: 1076  
TracerPid: 0  
Uid: 1000 1000 1000 1000  
Gid: 1000 1000 1000 1000  
franco@franco:~$
```