

# Guía de uso de Bash

La intención de esta guía es contar con un "machete" de comandos de la consola de Linux que suelen ser útiles a lo largo de la cursada y en las entregas.

Para familiarizarse con el uso de los más básicos de una forma más interactiva, los invitamos a jugar a **Mario Bash** 🍄

## Comandos Básicos <sup>[1]</sup>

### Ver la documentación oficial

El comando `man` nos permite ver el **manual** de cualquier comando de Linux, o también cualquier función estándar de C.

Por ejemplo, `man fopen` nos explicará cómo funciona la función `fopen()`, y `man ls` nos explicará en detalle el comando `ls`.

TIP

También pueden encontrar los mismos manuales en línea:

- [fopen\(3\) — Linux manual page](#)
- [ls\(1\) — Linux manual page](#)

### Moverse entre carpetas

Comando	Descripción	Parámetros
<code>pwd</code>	Imprimir la ruta absoluta del directorio actual (del inglés: <i>print working directory</i> ).	-
<code>cd</code>	Cambiar el directorio actual.	La ruta (relativa o absoluta) <sup>[2]</sup> a un directorio.
<code>ls</code>	Listar el contenido de un directorio.	La ruta a un directorio. Si no es especificado, toma el valor del directorio actual.
<code>tree</code>	Imprimir el <i>árbol</i> de directorios con sus archivos partiendo desde el directorio indicado.	La ruta a un directorio. Si no es especificado, toma el valor del directorio actual.

#### TIP 1

- `.` es el directorio actual.
- `..` es el directorio padre (relativo al actual).
- `~` es el directorio home.

Ejemplos:

- `/home/utnso/./Desktop` es lo mismo que `/home/utnso/Desktop`
- `/home/utnso/./username` es lo mismo que `/home/username`
- `~/Documents` es lo mismo que `/home/utnso/Documents`

#### TIP 2

Si a `ls` le incluimos los flags `-l` para listar la información del archivo y `-a` para incluir los archivos ocultos (que empiezan con `.`), veremos algo parecido a esto:

```
$ ls -la
drwxr-xr-x 30 utnso utnso 4096 jul  7 00:18 .
drwxr-xr-x  3 root  root  4096 feb 19  2018 ..
-rwxrwxr-x 13 utnso utnso 4096 jul  7 00:18 mi-script.sh
```

De esta forma podemos ver los permisos, el contador de *hard links*<sup>[3]</sup>, el usuario y grupo *owner*, el tamaño y la fecha de última modificación de cada archivo.

## Crear, mover y borrar archivos

Comando	Descripción	Parámetros
<code>mkdir</code>	Crear un directorio vacío (del inglés: <i>make directory</i> ).	El nombre del directorio.
<code>touch</code>	Crear un archivo.	El nombre del archivo.
<code>cp</code>	Copiar un archivo.	La ruta origen y destino.
<code>mv</code>	Mover/renombrar un archivo.	El nombre anterior del archivo y el nuevo.
<code>rm</code>	Remover/eliminar un archivo.	El nombre del archivo.

#### TIP 1

Para eliminar una carpeta con todos sus archivos dentro, se debe agregar el flag `-r` (de "recursive"). Ejemplo: `rm -r /usr/bin/eclipse`

### TIP 2

Todos estos comandos (excepto **touch** ) permiten visualizar que la operación se realizó correctamente utilizando el flag **-v** . Ejemplo:

```
$ touch README.md
$ mkdir -v docs
mkdir: created directory 'docs'

$ cp -v README.md HELP.md
'README.md' -> 'HELP.md'

$ mv -v HELP.md docs/HELP.md
renamed 'HELP.md' -> 'docs/HELP.md'

$ rm -v docs/HELP.md
removed 'docs/HELP.md'
```

## Visualizar archivos

Comando	Descripción	Parámetros
less	Ver el contenido de un archivo.	El nombre del archivo.
cat	Imprimir el contenido de uno o varios archivos (del inglés: <i>concatenate</i> ).	El nombre de uno o más archivos.
head	Imprimir las primeras 10 líneas de un archivo.	El nombre del archivo.
tail	Imprimir las últimas 10 líneas de un archivo.	El nombre del archivo.
hexdump	Imprimir el contenido de un archivo en <b>hexadecimal</b> .	El nombre del archivo.
grep	Filtrar e imprimir el contenido de un archivo.	El filtro y el nombre del archivo.

### TIP 1

Tanto para **head** como para **tail** se puede especificar el número de líneas a leer través del flag **-n** :

```
# Lee las primeras 5 líneas del archivo ~/.bashrc
head -n 5 ~/.bashrc
# Lee las últimas 7 líneas del archivo ~/.bashrc
tail -n 7 ~/.bashrc
```

### TIP 2

Una de las grandes utilidades de **tail** es que junto con el flag **-f** nos permite visualizar las líneas que se van agregando a un archivo en tiempo real.

Esto viene como anillo al dedo para monitorear archivos de logs durante la entrega, por ejemplo, ejecutando:

```
tail -f kernel.log
```

bash

### TIP 3

Una forma más bonita de usar el comando **hexdump** es junto con el flag **-C**, el cual agrega una columna a la derecha imprimiendo el contenido del archivo en ASCII.

```
$ hexdump -C main.c

00000000  23 69 6e 63 6c 75 64 65  20 3c 73 74 64 6c 69 62  |#include <stdlib|
00000010  2e 68 3e 0a 23 69 6e 63  6c 75 64 65 20 3c 73 74  |.h>.#include <st|
00000020  64 69 6f 2e 68 3e 0a 0a  69 6e 74 20 6d 61 69 6e  |dio.h>..int main|
00000030  28 29 20 7b 0a 09 70 72  69 6e 74 66 28 22 48 65  |() {..printf("He|
00000040  6c 6c 6f 20 57 6f 72 6c  64 21 21 21 22 29 3b 0a  |llo World!!!");.|
00000050  09 72 65 74 75 72 6e 20  30 3b 0a 7d 0a           |.return 0;|.}|
0000005d
```

Las funciones de [memory.h](#) de las commons imprimen un stream de un cierto tamaño utilizando este formato.

## Cambiar permisos y ownership

### chmod

Permite cambiar los permisos de un archivo (del inglés: **change mode**). Ejemplo:

```
# Para dar permisos de ejecución
chmod +x mi-script.sh

# Para configurar nuevos permisos usando el formato Unix
chmod 664 kernel.config
```

bash

### chown

Permite cambiar el usuario dueño de un archivo (del inglés: **change owner**). Ejemplo:

```
# Para cambiar el ownership de un archivo a mi usuario
chown $USER ejemplo.txt

# Para cambiar el ownership de una carpeta y todo su contenido
chown -R $USER /home/utnso/swap
```

bash

### TIP 1

Recuerden que pueden validar los permisos del archivo ejecutando **ls -l** sobre el directorio en donde se encuentre como les mostramos [aquí](#).

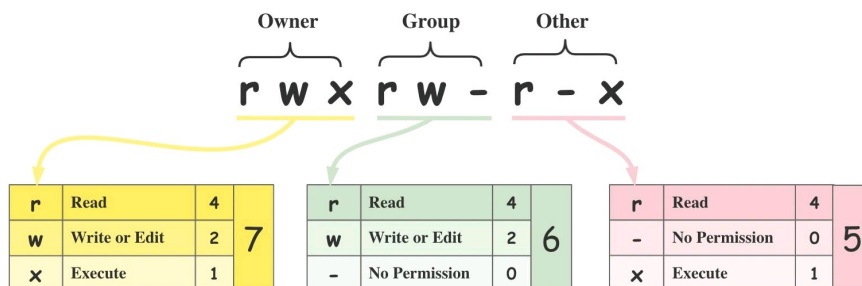
## TIP 2

Les dejamos un machete para escribir los permisos tipo Unix en octal:

# Linux File Permissions

 blog.bytbytego.com

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwX	Read + Write + Execute



## Comandos útiles para la entrega

### htop

Un administrador de procesos de Linux (del inglés: *human-readable table of processes*).

Las features más destacadas son:

- Visualizar el uso de CPU y RAM (para detectar esperas activas y memory leaks).
- Ordenar los procesos por PID, nombre, uso de CPU/RAM, etc. con el mouse.
- Filtrar los procesos (e hilos KLT) por nombre con **F4**.
- Enviar **señales** a uno o varios procesos de forma intuitiva con **F9**.

CPU:     100.0%										Tasks: 31, 31 thr: 2 running									
Mem:     88.9M/1000M										Load average: 0.15 0.04 0.01									
Swap:     0K/1024M										Uptime: 00:01:39									
PID	USER	PRI	NI	UIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command								
1710	utnso	20	0	2068	552	504	R	100.	0.1	0:03.01	./main								
1709	utnso	20	0	7020	3320	2792	R	0.0	0.3	0:00.03	htop								
1133	root	20	0	881M	36704	25376	S	0.0	3.6	0:00.16	/usr/bin/dockerd -H fd://								
1152	root	10	-10	3444	2916	2056	S	0.0	0.3	0:00.02	/sbin/iscsid								
1	root	20	0	6708	4908	3544	S	0.0	0.5	0:01.15	/sbin/init								
365	root	20	0	5744	2640	2372	S	0.0	0.3	0:00.04	/lib/systemd/systemd-journald								
408	root	20	0	22500	3436	1168	S	0.0	0.3	0:00.00	/sbin/lvmtd -f								
433	root	20	0	14052	3664	2760	S	0.0	0.4	0:00.08	/lib/systemd/systemd-udev								
746	systemd-t	20	0	12596	2328	2144	S	0.0	0.2	0:00.00	/lib/systemd/systemd-timesyncd								
736	systemd-t	20	0	12596	2328	2144	S	0.0	0.2	0:00.00	/lib/systemd/systemd-timesyncd								
849	root	20	0	20364	1512	1412	S	0.0	0.1	0:00.00	/usr/bin/xcfs /var/lib/xcfs/								
850	root	20	0	20364	1512	1412	S	0.0	0.1	0:00.00	/usr/bin/xcfs /var/lib/xcfs/								
833	root	20	0	20364	1512	1412	S	0.0	0.1	0:00.00	/usr/bin/xcfs /var/lib/xcfs/								
835	daemon	20	0	3480	1980	1816	S	0.0	0.2	0:00.00	/usr/sbin/atd -f								
836	messagebu	20	0	6048	3708	3328	S	0.0	0.4	0:00.01	/usr/bin/dbus-daemon --system --adre								
880	root	20	0	38884	5912	5404	S	0.0	0.6	0:00.00	/usr/lib/accountsservice/accounts-dae								
887	root	20	0	38884	5912	5404	S	0.0	0.6	0:00.00	/usr/lib/accountsservice/accounts-dae								
851	root	20	0	38884	5912	5404	S	0.0	0.6	0:00.01	/usr/lib/accountsservice/accounts-dae								
877	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
879	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
888	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
905	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
906	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
907	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.00	/usr/lib/snapd/snapd								
853	root	20	0	833M	16968	9356	S	0.0	1.7	0:00.01	/usr/lib/snapd/snapd								
855	root	20	0	2244	1116	1048	S	0.0	0.1	0:00.00	/usr/sbin/acpid								
857	root	20	0	6796	2724	2504	S	0.0	0.3	0:00.00	/usr/sbin/cron -f								
881	syslog	20	0	30728	3056	2528	S	0.0	0.3	0:00.00	/usr/sbin/rsyslogd -n								
882	syslog	20	0	30728	3056	2528	S	0.0	0.3	0:00.00	/usr/sbin/rsyslogd -n								
883	syslog	20	0	30728	3056	2528	S	0.0	0.3	0:00.00	/usr/sbin/rsyslogd -n								

#### TIP

En la última línea pueden encontrar las distintas opciones que se pueden usar.

## ifconfig

Permite consultar la IP de la VM actual para luego agregarla a los archivos de configuración (ya sea a mano o a través de un script).

```
utnso@ubuntu-server:~$ ifconfig
docker0: Link encap:Ethernet HWaddr 02:42:9b:f5:c7:8f
          inet addr:172.17.0.1 Bcast:0.0.0.0 Mask:255.255.0.0
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

enp0s3: Link encap:Ethernet HWaddr 08:00:27:a3:d5:84
          inet addr:192.168.0.142 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea3:d584/64 Scope:Link
          inet6 addr: 2800::810:40e:8189:a00:27ff:fea3:d584/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1276 (1.2 KB) TX bytes:1612 (1.6 KB)

lo: Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:176 errors:0 dropped:0 overruns:0 frame:0
          TX packets:176 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:13296 (13.2 KB) TX bytes:13296 (13.2 KB)
```

Esa es la IP de la VM dentro de la red local

## nano

Un editor de texto liviano simple que funciona desde la consola. Es la alternativa recomendada si llegan a necesitar editar algún archivo de configuración.

```
GNU nano 2.5.3      File: main.c      Modified

#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("Hello World!!!");
    return 0;
}
```

Get Help Write Out Where Is Read 8 lines Cut Text Justify Cur Pos Prev Page  
Exit Read File Replace Uncut Text To Spell Go To Line Next Page

### TIP

En la última línea pueden ver las distintas opciones que se pueden usar.

Por ejemplo: **Ctrl + x** para salir.

► Probemos con un ejemplo...

## lsof

Permite listar todos los archivos abiertos. En las entregas, puede ser muy útil junto con el flag **-i** para corroborar que no haya ningún proceso escuchando en un puerto en particular.

Ejemplo:

```
lsof -i :8080
```

bash

```
utnso@utnso:~$ lsof -i :8080
COMMAND PID  USER  FD   TYPE DEVICE SIZE/OFF NODE NAME
server  1198 utnso   3u   IPv4  22586      0t0  TCP *:http-alt (LISTEN)
```

---

## Redirecciones y pipes<sup>[4]</sup>

Todos los procesos reciben información via la *entrada* estándar (o **stdin**) y devuelven un resultado por la *salida* estándar (o **stdout**). La entrada estándar de varios comandos como `cat` , `head` y `tail` suele ser un archivo, y la salida estándar, la consola.

Sin embargo, es posible *redirigir* esos flujos de información (o *streams*) para que los comandos "se pasen" el resultado el uno al otro, o lo guarden en un archivo. Para esto, utilizamos **operadores de redirección**.

Estos son los más básicos, junto con un ejemplo de uso:

Operadores	Caso de uso	Ejemplo
>	Escribir <code>stdout</code> en un archivo, sobrescribiéndolo en caso de ya existir.	<pre>echo "https://USER:TOKEN@github.com" &gt; ~/.git-credentials</pre>
>>	Concatenar <code>stdout</code> al final de un archivo existente (o crearlo en caso de no existir).	<pre>echo "IP_CONSOLA=192.168.0.200" &gt;&gt; kernel.config</pre>
	"Pasarle" el <code>stdout</code> de un comando al <code>stdin</code> de otro	<pre>cat *.c   grep sleep</pre>

---

## Variables de entorno<sup>[5]</sup>

Las variables de entorno son aquellas variables definidas para la consola que estamos utilizando actualmente, pero que podemos almacenar para que sean reconocidas cada vez que abrimos una nueva terminal.

Corriendo el comando `env` podemos visualizar todas las variables de Linux que tenemos configuradas. Aparecerán mostrando su nombre seguido de su valor, como por ejemplo `HOME=/home/utnso` .

Para poder ver el valor de una variable de entorno en particular, podemos hacerlo usando el comando `echo` y su nombre, por ej. `echo $USER` , nos mostrará el nombre de nuestro usuario.

**TIP**

Para acceder a las variables utilizamos el símbolo `$` .

### ¿Cómo configuro una variable de entorno?

Corriendo `export NOMBRE=valor` , podemos configurar una variable de entorno.



Por ejemplo, si hacemos `export MI_VARIABLE='aguante sistemas operativos'` y después listamos todas las variables disponibles con `env`, veremos que la que creamos recién es parte de las mismas. Haciendo `echo $MI_VARIABLE`, vemos que nos imprime por pantalla el valor de la misma.

El problema que esto tiene es que si probamos cerrando la terminal actual y abriendo otra sesión, veremos que al correr `echo $MI_VARIABLE` nuevamente, no nos devolverá nada.

¿Qué ocurrió? Nuestra variable de entorno había sido exportada únicamente para la sesión en la que estábamos trabajando y sus sesiones hijas, por lo tanto, no persistió.

Ahora, ¿cómo hacemos para definir una variable que valga para todas las sesiones? Podemos hacerlo agregando el `export` al final del archivo `~/.bashrc`.

#### TIP

`~/.bashrc` es un script que se va a ejecutar siempre que iniciemos una consola.

► Probemos con un ejemplo...

---

## Material recomendado

---

1. [34 comandos básicos de Linux que todo usuario debería conocer](#) (en inglés) ↩
2. [Rutas Relativas y Rutas Absolutas](#) ↩
3. [Tutorial sobre Hard Links y Soft Links](#) ↩
4. [5 formas de usar operadores de redirección en Bash](#) (en inglés) ↩
5. [Como leer y configurar variables de entorno en Linux](#) (en inglés) ↩