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 [1]

Ver la documentación oficial

El comando `man` nos permite ver el **man**ual 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
`pwd`	Imprimir la ruta absoluta del directorio actual (del inglés: ***p**rint **w**orking **d**irectory*).	-
`cd`	**C**ambiar el **d**irectorio actual.	La ruta (relativa o absoluta)[2] a un directorio.
`ls`	**Ls**tar el contenido de un directorio.	La ruta a un directorio. Si no es especificado, toma el valor del directorio actual.
`tree`	Imprimir el *árbol* 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*[3], el usuario y grupo *owner*, el tamaño y la fecha de última modificación de cada archivo.

Crear, mover y borrar archivos

Comando	Descripción	Parámetros
`mkdir`	Crear un directorio vacío (del inglés: ***mk*e *dir***ectory).	El nombre del directorio.
`touch`	Crear un archivo.	El nombre del archivo.
`cp`	**Cop**iar un archivo.	La ruta origen y destino.
`mv`	**M**over/renombrar un archivo.	El nombre anterior del archivo y el nuevo.
`rm`	**Rem**over/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:

```bash
$ 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: *con**cat**enate*).	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 **hexadecimal**.	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`:

```bash
# 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:

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

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: ***ch****ange* ***mod****e*). Ejemplo:

```bash
# Para dar permisos de ejecución
chmod +x mi-script.sh
# Para configurar nuevos permisos usando el formato Unix
chmod 664 kernel.config
```

chown

Permite cambiar el usuario dueño de un archivo (del inglés: ***ch****ange* ***own****er*). Ejemplo:

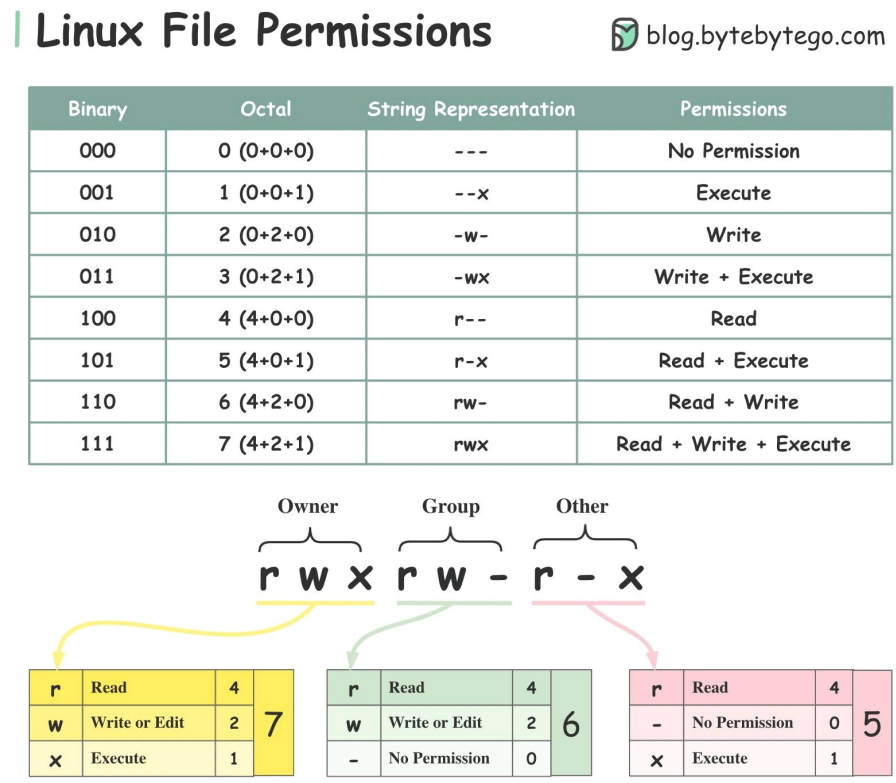
```bash
# Para cambiar el ownership de un archivo a mi usuario
chown $USER ejemplo.txt
# Para cambiar el ownership de una carpeta y todo su contenido
chown -R $USER /home/utnso/swap
```

TIP 1

Recuerden que pueden validar los permisos del archivo ejecutando `ls -l` sobre el directorio en donde se encuentre como les mostramos **acá**.

TIP 2

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



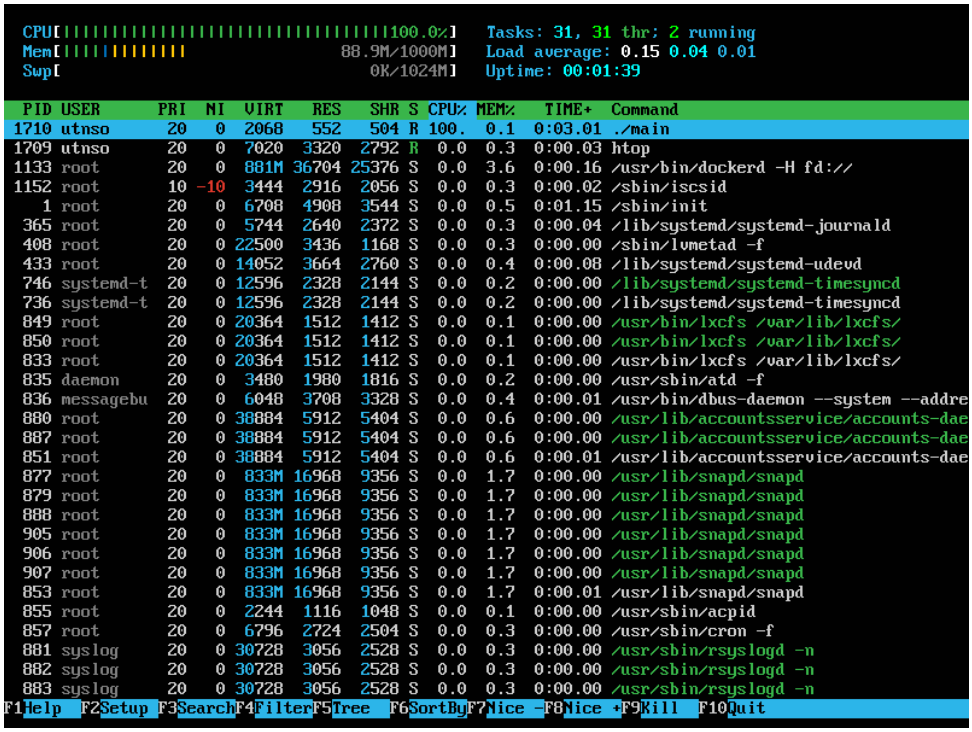
Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute

Comandos útiles para la entrega

`htop`

Un administrador de procesos de Linux (del inglés: ***h**uman-readable **t**able **o**f **p**rocesses*). Las features más destacadas son:

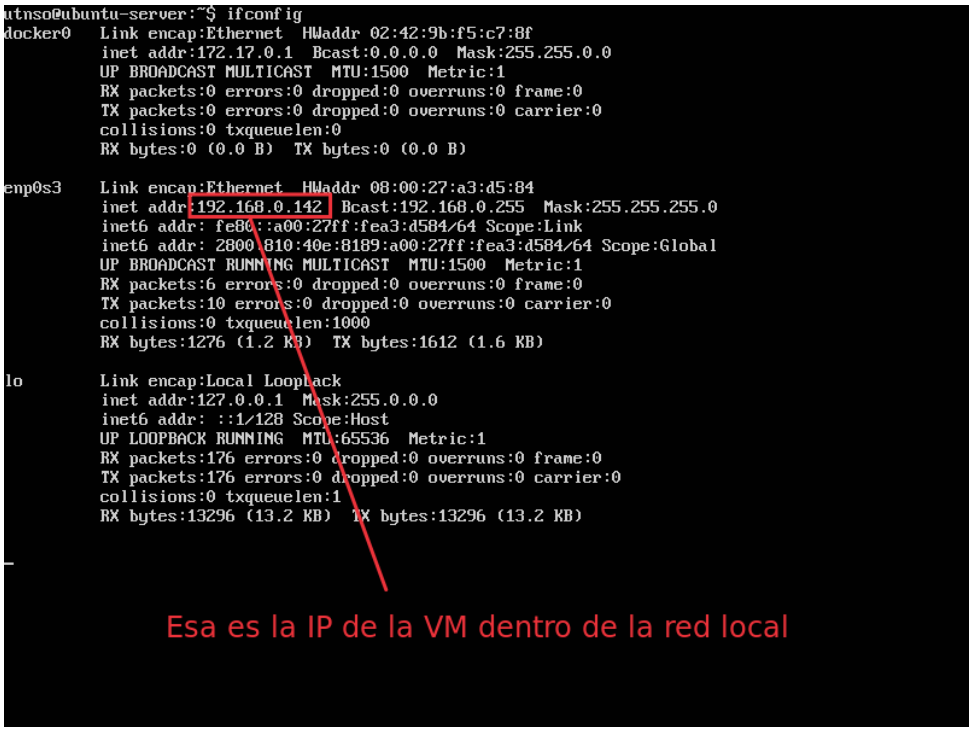
- Visualizar el uso de CPU y RAM (para detectar esperas activas y memory leaks).
- Ordenar los procesos por PID, nombre, uso de CPU/RAM, etc. con el mouse.
- Filtrar los procesos (e hilos KLT) por nombre con `F4` .
- Enviar señales a uno o varios procesos de forma intuitiva con `F9` .



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

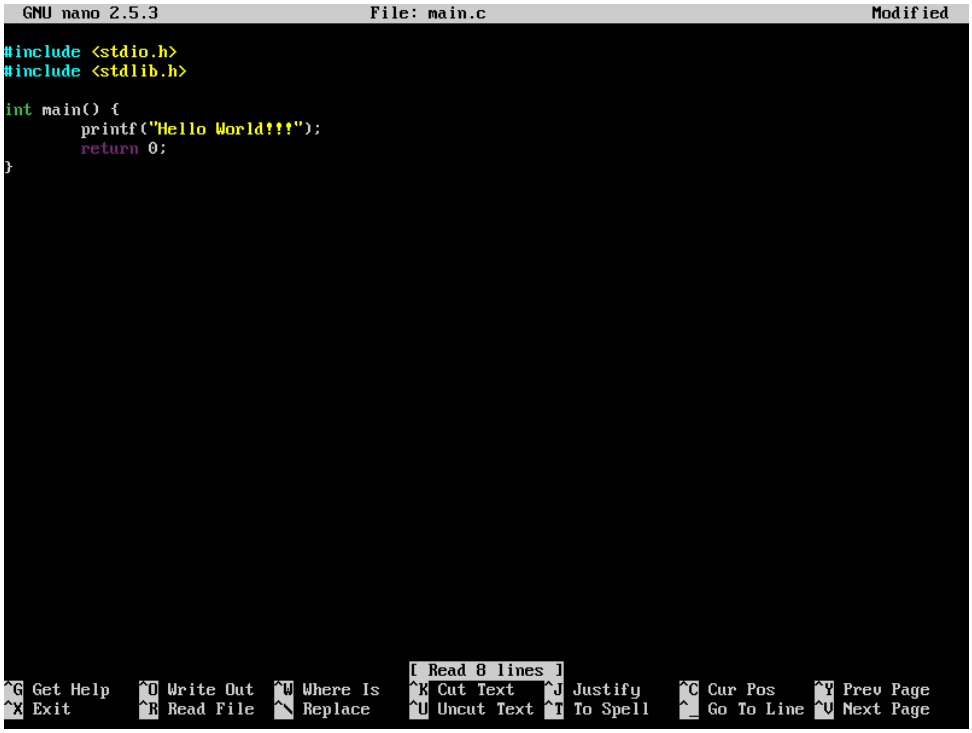
ifconfig

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).



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.



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:

```bash
lsof -i :8080
```

```
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[4]

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 `stdout` en un archivo, sobreescribiéndolo en caso de ya existir.	`echo "https://USER:TOKEN@github.com" > ~/.git-credentials`
`>>`	Concatenar `stdout` al final de un archivo existente (o crearlo en caso de no existir).	`echo "IP_CONSOLA=192.168.0.200" >> kernel.config`
`|`	"Pasarle" el `stdout` de un comando al `stdin` de otro	`cat *.c | grep sleep`

Variables de entorno[5]

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) ↩