

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Sistemas Operativos I

Práctica I: Introducción a los comandos Linux



Pedraza Celón Ian Yael

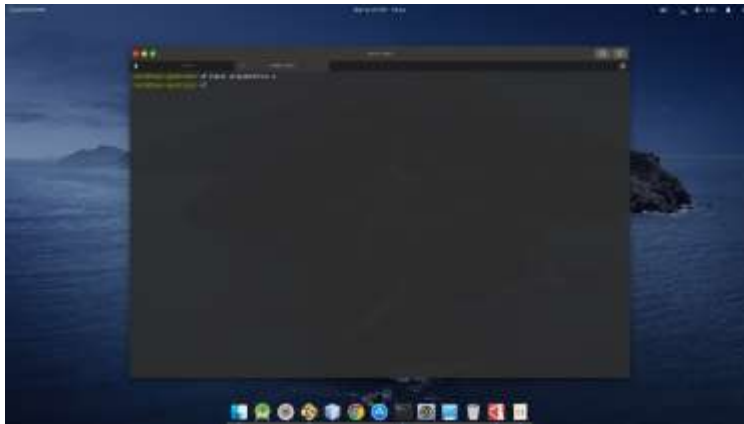
Linux es un sistema operativo que, a diferencia de sus competidores Windows y Mac, es de código abierto. Lo que quiere decir que cualquiera podría desarrollar características y módulos para este sistema, y como consecuencia lo convierte en un sistema operativo reprogramable.

Gracias a estas características es que Linux tiene una gran comunidad de desarrollo apoyándolo por detrás, haciéndolo uno de los preferidos para los grandes desarrollos de software.

El **objetivo** de esta práctica es conocer los comandos básicos del sistema operativos Linux y los conceptos básicos para desenvolverse en este sistema.

Desarrollo.

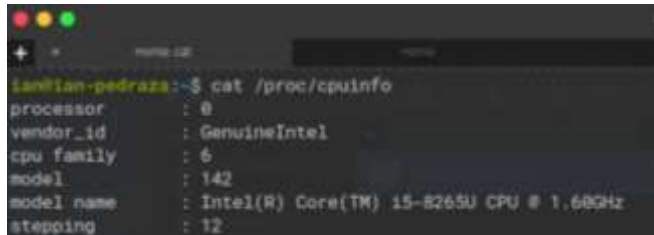
- ## 1. Ouvrir la terminal Linux.



2. Para acceder a los manuales de ayuda de los comandos se usa el comando `man`. Digite el siguiente comando **man cat** y para salir de la ayuda presione la tecla **q**.



3. Digite el siguiente comando **cat /proc/cpuinfo** y anote el nombre del CPU y la velocidad.

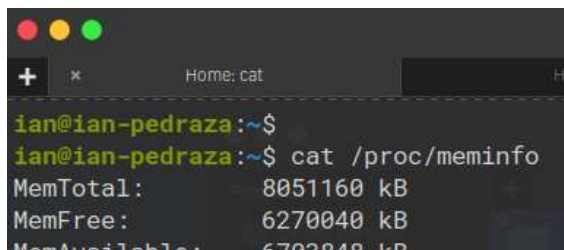


```
ian@ian-pedraza:~$ cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 142
model name     : Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
stepping       : 12
```

Nombre del CPU: Intel Core i5-8265U

Velocidad: 1.60 GHz

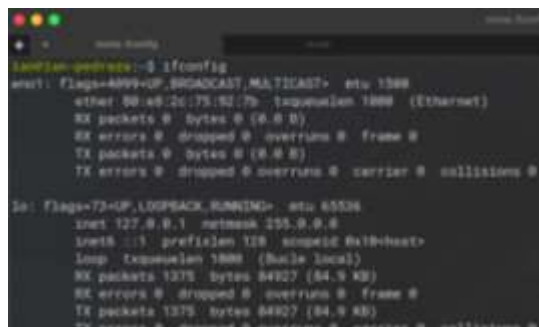
4. Digite el siguiente comando **cat /proc/meminfo** y anote la memoria total del sistema.



```
ian@ian-pedraza:~$ cat /proc/meminfo
MemTotal:      8051160 kB
MemFree:       6270040 kB
MemAvailable:  6702848 kB
```

Memoria total 8051160 kb -> 8GB

5. Digite el siguiente comando **ifconfig** y anote la dirección IP del sistema.



```
ian@ian-pedraza:~$ ifconfig
enp1s0: flags=4099<UP,BROADCAST,MULTICAST>  etu 1588
    ether 50:ad:2c:75:82:7b  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  etu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 010<host>
    loop  txqueuelen 1000  (local loop)
    RX packets 1375  bytes 84927 (84.9 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1375  bytes 84927 (84.9 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

IP: 127.0.0.1

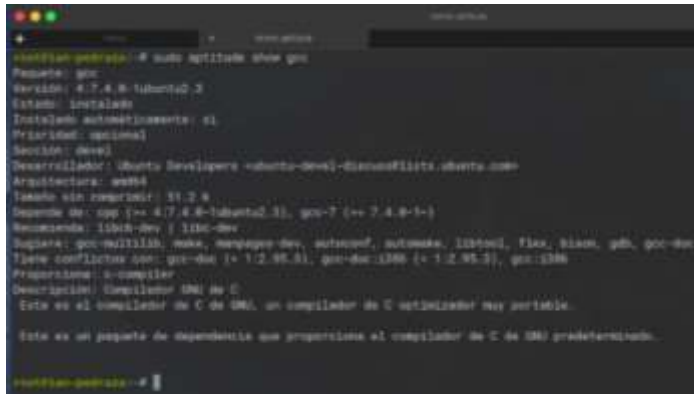
6. Digite el siguiente comando **uname -a** y tome nota de



```
ian@ian-pedraza:~$ uname -a
Linux ian-pedraza 4.15.0-74-generic #84-Ubuntu SMP Thu Dec 19 08:06:28 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

- a. Nombre del ordenador: ian-pedraza
- b. La versión de kernel que está usando: 4.15.0–74-generic #84 Ubuntu
- c. Fecha de compilación del kernel: 19/12/19

Digite el siguiente comando **udo aptitude show gcc**, deduzca



```
lanfrán-pedraza:~$ sudo aptitude show gcc
Paquete: gcc
Versión: 4:7.4.0-1ubuntu2.3
Estado: instalado
Instalado automáticamente: si
Prioridad: opcional
Sección: devel
Desarrollador: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Arquitectura: amd64
Tamaño del paquete: 33.2 kB
Depende de: gcc (= 4:7.4.0-1ubuntu2.3), gcc-7 (= 7.4.0-1~)
Recomienda: libc-dev, libc-dev
Sugiere: gcc-multilib, make, makepkg-dev, autoconf, automake, libtool, flex, bison, gdb, gcc-doc
Tipo conflictos con: gcc-doc (= 1:2.95.1), gcc-doc:i386 (= 1:2.95.1), gcc:i386
Proporciona: c-compiler
Descripción: Compilador GNU de C
Este es el compilador de C de GNU, un compilador de C utilizado muy portable.
Este es el paquete de dependencias que proporciona el compilador de C de GNU predefinido.

lanfrán-pedraza:~$
```

- a. El nombre del paquete: gcc
- b. La versión se encuentra instalada: 4:7.4.0-1ubuntu2.3
- c. La versión de actualización: 4:7.4.0-1ubuntu2.3

7. Digite el siguiente comando [usuario@localhost /home/usuario]\$ **ls**



```
lanfrán-pedraza:~$ ls
Android  Desktop  Idiomas  netbeans-8.0.1  Plantillas  snap  zshrc
Descargas  Documentos  Música  NetBeansProjects  Publico  Videos
```

- a. Que carpetas se ven con ese comando:
Las carpetas visibles de la ruta actual.
- b. Donde están ubicadas esas carpetas
Home
- c. Deduzca para que sirven las carpetas home
Es el directorio raíz del sistema operativo
- d. Proporciona un listado largo (permisos, tamaños, dueño, etc.), digita **ls -l**

```
ian@ian-pedraza:~$ ls -l
total 52
drwxr-xr-x 3 ian ian 4096 nov 10 14:31 Android
drwxr-xr-x 5 ian ian 4096 nov 15 21:22 Descargas
drwxr-xr-x 2 ian ian 4096 nov 10 13:56 Desktop
drwxr-xr-x 5 ian ian 4096 dic 3 18:47 Documentos
drwxr-xr-x 3 ian ian 4096 ene 14 09:07 Descargas
drwxr-xr-x 2 ian ian 4096 nov 9 13:25 Musica
drwxr-xr-x 14 ian ian 4096 nov 11 17:31 sistemas-8.8.1
drwxr-xr-x 3 ian ian 4096 nov 10 16:34 NetBeansProyecto
drwxr-xr-x 2 ian ian 4096 nov 9 13:25 Practicas
drwxr-xr-x 2 ian ian 4096 nov 9 13:25 Proyectos
drwxr-xr-x 3 ian ian 4096 nov 10 14:50 snap
drwxr-xr-x 2 ian ian 4096 nov 9 13:25 Videos
drwxr-xr-x 16 ian ian 4096 nov 22 10:10 Zojel
ian@ian-pedraza:~$
```

e. [usuario@localhost /home/usuario]\$ ls -l *.f ¿Qué realiza?
Muestra los archivos detalladamente

8. Crea la siguiente estructura de directorios y subdirectorios usando los comandos en terminal:

a) **mkdir [opcion] nombredir**- Crear un directorio

```
SOS >Practicas
    >Tareas
    >Proyectos
```

```
root@ian-pedraza:~# mkdir SOS
root@ian-pedraza:~# cd SOS
root@ian-pedraza:~/SOS# mkdir Practicas
root@ian-pedraza:~/SOS# mkdir Tareas
root@ian-pedraza:~/SOS# mkdir Proyectos
root@ian-pedraza:~/SOS# ls
Practicas  Proyectos  Tareas
root@ian-pedraza:~/SOS#
```

b) Nano es un editor de texto fácil de utilizar. abrelo[usuario@localhost /home/usuario/]\$ nano y crea un archivo con tu nombre.txt en Practicas y cópialo en Proyectos usa **cp [opcion] fuente destino**-Copia de archivos



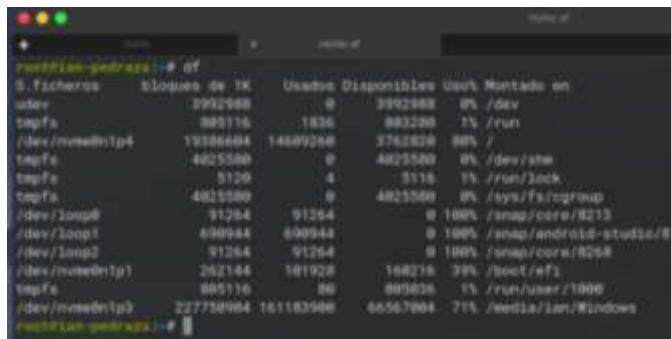
```
root@ian-pedraza:~/SOS/Practicas# nano nombre.txt
root@ian-pedraza:~/SOS/Practicas# cp nombre.txt ../Proyectos/
```

- c) **rm [opcion] nombrearch**-Borrar archivos, ahora bórralo de practicas



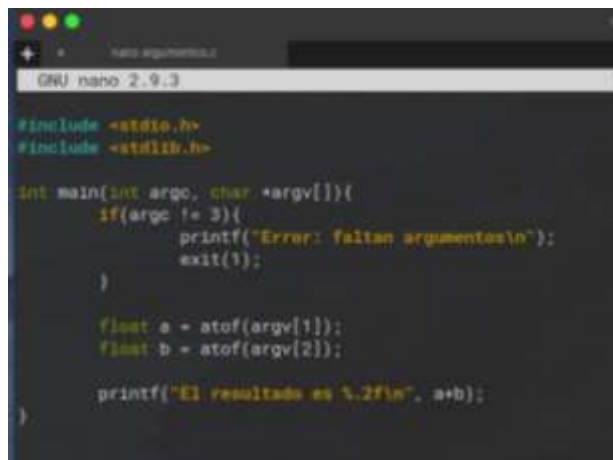
```
root@ian-pedraza:~/505/Practicas# nano nombre.txt
root@ian-pedraza:~/505/Practicas# cp nombre.txt ../Proyectos/
root@ian-pedraza:~/505/Practicas# rm nombre.txt
```

- d) Escribe el cp, ando [usuario@localhost /home/usuario/]\$ **df** ¿Qué hace este comando?



```
root@ian-pedraza:~# df
Filesystem            bloques de 1K  Usados Disponibles  Use% Montado en
udev                  3992968        0  3992968         0% /dev
tmpfs                 885116      1836    883280         1% /run
/dev/mmcblk0p4       19386684 14889288  3762828    88% /
tmpfs                4825580        0   4825580         0% /dev/shm
tmpfs                 5120         4      5116         1% /run/lock
tmpfs                4825580        0   4825580         0% /sys/fs/cgroup
/dev/loop4            91264       91264         0  100% /snap/core/8213
/dev/loop1            608544     608544         0  100% /snap/android-studio/81
/dev/loop2            91264       91264         0  100% /snap/core/8268
/dev/mmcblk0p1       262144    181928    188216    39% /boot/efi
tmpfs                 885116        0    885116         1% /run/user/1000
/dev/mmcblk0p3       227758984 161183968  66575084    71% /media/ian/Windows
```

9. Realiza el programa en C de recibir argumentos en el main y guárdalo en Tareas.



```
GNU nano 2.9.3
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
    if(argc != 3){
        printf("Error: faltan argumentos\n");
        exit(1);
    }

    float a = atof(argv[1]);
    float b = atof(argv[2]);

    printf("El resultado es %.2f\n", a+b);
}
```

10. Realiza el programa en C de copiar un archivo a otro mediante argumentos en línea de comando y guárdalo en prácticas.

```
GNU nano 2.9.3 practicas.c

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

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Error: Faltan argumentos!\n");
        exit(1);
    }

    char comando[1000];
    strcpy(comando, "cp ");
    strcat(comando, argv[1]);
    strcat(comando, " ");
    strcat(comando, argv[2]);

    system(comando);
    return 0;
}
```

```
GNU nano 2.9.3 practicas.c
Hello World From Practicas
```

```
practicas-practicas:~/Practicas$ nano archive.txt
practicas-practicas:~/Practicas$ ls
archive.txt  args  argumentos_archivos.s  argumentos.c
practicas-practicas:~/Practicas$ cd ..
practicas-practicas:~$ ls
Practicas  Proyectos  Tarwas
practicas-practicas:~$ cd Practicas/
practicas-practicas:~/Practicas$ gcc argumentos_archivos.s -o args
practicas-practicas:~/Practicas$ ./args archive.txt ..
practicas-practicas:~/Practicas$ cd ..
practicas-practicas:~$ ls
archive.txt  Practicas  Proyectos  Tarwas
practicas-practicas:~$ nano archive.txt
practicas-practicas:~$
```

```
GNU nano 2.9.3 practicas.c
Hello World From Practicas
```

Parte II

1. Ejecución Primer Plano Por ejemplo, imagine que se desea hacer ping a una maquina por un tiempo indefinido:

`usuario@localhost /home/usuario/]$ ping localhost`

Al ejecutarse el programa anterior, observara al prompt del sistema ocupado, lo que significa que se está ejecutándose en primer plano. Presione CTRL+C para cancelar la tarea.



```
root@ian-pedraza:~/500# ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data:
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.051 ms
```

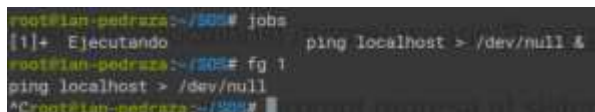
2. Ejecución en segundo plano. Ahora escriba lo siguiente:
`[usuario@localhost/home/usuario]$ ping localhost >/dev/null &`



```
root@ian-pedraza:~/500# ping localhost >/dev/null &
[1] 3731
root@ian-pedraza:~/500#
```

Observara que el prompt regresa al sistema operativo, sin embargo si usted ejecuta el comando **Jobs**, Linux le muestra las tareas que se están ejecutando.

Para mover una tarea que se esta ejecutando en background a foreground, solo es necesario digital **Jobs** y luego **fg no_tarea**



```
root@ian-pedraza:~/500# jobs
[1]+  Ejecutando                  ping localhost > /dev/null &
root@ian-pedraza:~/500# fg 1
ping localhost > /dev/null
^Croot@ian-pedraza:~/500#
```

3. Visualizar todos los procesos que se están ejecutando en Linux, escriba **ps [-all]**



```
root@ian-pedraza:~/500# ps -all
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
# S 0 2329 2310 0 00 0 - 12347 wait pts/r 00:00.01 bash
# S 0 3476 2329 0 00 0 - 13052 - pts/r 00:00.00 ps
root@ian-pedraza:~/500#
```


4. **who [opciones]:** informa quien esta actualmente en el sistema:

whoami muestra quienes estamos en la terminal en la que estamos trabajando

who muestra quienes están conectados al sistema en ese momento

```
root@lan-pedraza:~/# who
lan      tty7      2020-01-15 21:48 (:0)
root@lan-pedraza:~/# whoami
root
root@lan-pedraza:~/#
```

5. **pwd**, este comando imprime el directorio actual en pantalla, es la abreviación de **P**rint **W**orking **D**irectory

```
[usuario@localhost /home/usuario]$ pwd
```

```
root@lan-pedraza:~/SOS# pwd
/home/lan/SOS
root@lan-pedraza:~/SOS#
```

6. **ls -a: listar archivos invisibles:** Los archivos cuyos nombres comienzan por un punto (.) se llaman archivos invisibles, porque normalmente no aparecen con la instrucción **ls**

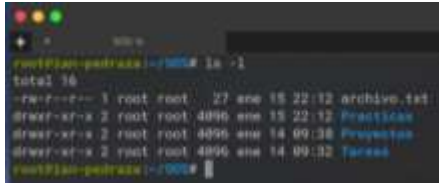
[illegible]

7. **more.** Se utiliza para visualizar archivos en la pantalla. La ventaja con el comando **cat** es que hace una pausa al llenar la pantalla; con la barra espaciadora se avanza una página, con **enter>** se avanza una línea y con **q** o **del** se finaliza el comando.

```
[usuario@localhost/home/usuario]$more archivomuestra [Se muestra
contenido del archivo]
```

```
root@ian-pedraza:~/SOS# more archivo.txt
Hello World From Practices
root@ian-pedraza:~/SOS#
```

8. **Permisos.** Para mostrar en pantalla los permisos de los archivos o directorios se utiliza el comando **ls** seguido de la opción **-l**



```
root@ian-pedraza:~/SOS# ls -l
total 16
-rw-r--r-- 1 root root 27 ene 15 22:12 archivo.txt
drwxr-xr-x 2 root root 4096 ene 15 22:12 Practicas
drwxr-xr-x 2 root root 4096 ene 14 09:38 Proyectos
drwxr-xr-x 2 root root 4096 ene 14 09:32 Tareas
root@ian-pedraza:~/SOS#
```

9. **Cambio de permisos. chmod:** cambia los permisos. Existen dos formas de cambiar los permisos, el modo simbólico y el modo absoluto.

Modo simbólico: Cuando se utiliza este modo, se puede añadir o quitar permisos a los archivos y directorios: El formato del comando es:

\$chmod [who] codigo_operador permisos archivo

Who: tipo de usuario que puede ser


U: propietario del archivo

O: Usuarios clasificados como todos

A: Todos los usuarios del sistema (propietario, grupo y otros)

Codigo_operador: el símbolo + se utiliza para añadir permisos el símbolo – los remueve .

Permisos: tipo de permiso el cual puede ser de solo lectura (r), escritura (w), o ejecución (x).



```
root@ian-pedraza:~/SOS# chmod a-r archivo.txt
root@ian-pedraza:~/SOS#
```

Modo absoluto: El modo absoluto se especifica con 3 dígitos numéricos.

Cada umero representa el permiso de cada tipo de usuario. Estos dígitos, se obtienen para cada clase de usuario a parir de los siguientes valores:

1. Ejecución
2. Escritura
4. Lectura

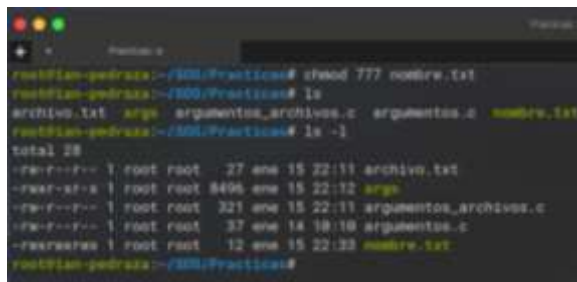
Sintaxis del comando:
chmod modo archivo

De donde:

- 0: ningún permiso
- 1: permiso de ejecución
- 2: permiso de escritura
- 3: permiso de ejecución + escritura (1+2)
- 4: permiso de lectura
- 5: permiso de lectura + ejecución (4+1)
- 6: permiso de lectura + escritura (4+2)
- 7: permiso de lectura, escritura y ejecución (4+2+1).

Realizar el cambio de los permisos del archivo creado Nombre.txt

```
[usuario@localhost /home/usuario/]$ chmod 777 Nombre.txt  
[usuario@localhost /home/usuario/]$ ls
```



```
root@kali:~/Practicas# chmod 777 nombre.txt  
root@kali:~/Practicas# ls  
archivo.txt  args  argumentos_archivos.c  argumentos.c  nombre.txt  
root@kali:~/Practicas# ls -l  
total 28  
-rwxrwxrwx 1 root root 27 ene 15 22:11 archivo.txt  
-rwxrwxrwx 1 root root 8496 ene 15 22:12 args  
-rwxrwxrwx 1 root root 321 ene 15 22:11 argumentos_archivos.c  
-rwxrwxrwx 1 root root 37 ene 14 18:18 argumentos.c  
-rwxrwxrwx 1 root root 12 ene 15 22:33 nombre.txt  
root@kali:~/Practicas#
```

Conclusiones:

Los comandos Linux nos proveen de herramientas muy específicas, que además de poder ir combinándolos, podemos hacer uso también de lenguajes de programación para poder generar programas mucho más completos e interesantes.

Pero conocer estos comandos es una parte fundamental para poder comenzar con el desarrollo de estos programas mucho más complejos.