Cátedra de Sistemas Operativos

**Tutorial de Introducción a Linux**

Versión 1.13

# Trabajo Práctico Nro. 1

# Integrantes:

## Santiago Fernández Núñez Mat.: 945634

## Matias Hillmann Mat.: 1040145

## Lautaro Corradini Mat.: 1030428

## Juan Jose Lopez Alvarez Mat.: 1053368

## 1. Introducción

Este trabajo práctico pretende introducir al alumno al uso, configuración y breve administración del sistema operativo que será usado como plataforma para los trabajos prácticos posteriores.

## 2. Objetivos

∙ Adquirir los conocimientos básicos necesarios para poder usar un sistema operativo moderno de tipo UNIX, como lo es Linux.

∙ Conocer y comprender las distintas herramientas de administración y configuración de dicho sistema operativo.

∙ Generar una base de información necesaria para la elaboración de los trabajos prácticos posteriores.

## 3. Desarrollo

**Nota:** tenga presente que se le solicita que informe que marca y versión utiliza de Linux, y que este Trabajo Práctico puede ser base para un trabajo practico posterior. Asimismo, dependiendo de la versión, algunos comandos pueden tener una sintaxis diferente. La entrega deberá ser solo en formato digital. En caso de ser recursante deberá utilizar una versión de Linux diferente a la presentada anteriormente (se conservan los históricos por 4 cuatrimestres).

**Marca:** Ubuntu

**Version:**

**mati@mati-VirtualBox:~$ uname -a**

**Linux mati-VirtualBox 5.4.0-28-generic #32-Ubuntu SMP Wed Apr 22 17:40:10 UTC 2020 x86\_64 x86\_64 x86\_64 GNU/Linux**

### 3.1. Ayuda

**3.1.1. man: es un programa que formatea y muestra las páginas del manual de referencia del sistema. El formato de uso básico es “man tema” donde tema es el nombre de la página del manual que se quiere ver.**

DESCRIPTION

man is the system's manual pager. Each

page argument given to man is normally

the name of a program, utility or func‐

tion. The manual page associated with

each of these arguments is then found and

displayed. A section, if provided, will

direct man to look only in that section

of the manual. The default action is to

search in all of the available sections

following a pre-defined order (see DE‐

FAULTS), and to show only the first page

found, even if page exists in several

sections.

**3.1.1.1. ¿Qué tipo de información provee el man, como la organiza internamente y como busca dentro de la misma? Para saberlo, tipee “man man” (sin comillas), use los cursores UP y DOWN para recorrer la pagina.**

**3.1.1.2. Salga de la página anterior tipeando “q”.**

**3.1.1.3. Investigue que hace el comando ls tipeando “man ls”.**

LS(1) User Commands LS(1)

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor

--sort is specified.

Mandatory arguments to long options are mandatory for short options too.

**3.1.1.4. Liste de manera ordenada (por tamaño del archivo) mediante el comando “ls” la mayor cantidad de información posible sobre todos los archivos que se encuentren en su directorio home (ej: /home/guest) incluyendo aquellos que empiezan con un punto.**

mati@mati-VirtualBox:~$ ls -a

. .cache .gnupg Pictures Templates Videos

.. .config .local .profile .vboxclient-clipboard.pid

.bash\_history Desktop .mozilla Public .vboxclient-display-svga-x11.pid

.bash\_logout Documents Music .ssh .vboxclient-draganddrop.pid

.bashrc Downloads .pam\_environment .sudo\_as\_admin\_successful .vboxclient-seamless.pid

**3.1.1.4.1. Tip: cuando se está viendo una página del manual, puede buscar cadenas tipeando “/cadenaABuscar” sin comillas.**

**3.1.1.4.2. Tip: para posicionarse dentro de un directorio debe hacer “cd directorio”. Ejemplo: cd /usr/local**

**3.1.1.5. Supongamos que se desea conocer el prototipo de la función de ANSI C printf(). Tipee “man printf” y vea que sucede. ¿Es la página que estábamos buscando?**

PRINTF(1) User Commands PRINTF(1)

NAME

printf - format and print data

SYNOPSIS

printf FORMAT [ARGUMENT]...

printf OPTION

DESCRIPTION

Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

--help display this help and exit

--version

output version information and exit

FORMAT controls the output as in C printf. Interpreted sequences are:

\" double quote

\\ backslash

\a alert (BEL)

\b backspace

\c produce no further output

\e escape

\f form feed

\n new line

\r carriage return

\t horizontal tab

\v vertical tab

\NNN byte with octal value NNN (1 to 3 digits)

\xHH byte with hexadecimal value HH (1 to 2 digits)

\uHHHH Unicode (ISO/IEC 10646) character with hex value HHHH (4 digits)

\UHHHHHHHH

Unicode character with hex value HHHHHHHH (8 digits)

%% a single %

%b ARGUMENT as a string with '\' escapes interpreted, except that octal escapes are of the form \0 or \0NNN

%q ARGUMENT is printed in a format that can be reused as shell input, escaping non-printable characters with the pro‐

posed POSIX $'' syntax.

and all C format specifications ending with one of diouxXfeEgGcs, with ARGUMENTs converted to proper type first. Variable

widths are handled.

NOTE: your shell may have its own version of printf, which usually supersedes the version described here. Please refer to

your shell's documentation for details about the options it supports.

*RTA: no, porque es el printf del shell y no del ansi C*

**3.1.2. whatis**

**3.1.2.1. Investigue que hace el comando whatis.**

WHATIS(1) Manual pager utils WHATIS(1)

NAME

whatis - display one-line manual page descriptions

SYNOPSIS

whatis [-dlv?V] [-r|-w] [-s list] [-m system[,...]] [-M path] [-L locale] [-C file] name ...

DESCRIPTION

Each manual page has a short description available within it. whatis searches the manual page names and displays the man‐

ual page descriptions of any name matched.

name may contain wildcards (-w) or be a regular expression (-r). Using these options, it may be necessary to quote the

name or escape (\) the special characters to stop the shell from interpreting them.

index databases are used during the search, and are updated by the mandb program. Depending on your installation, this

may be run by a periodic cron job, or may need to be run manually after new manual pages have been installed. To produce

an old style text whatis database from the relative index database, issue the command:

whatis -M manpath -w '\*' | sort > manpath/whatis

where manpath is a manual page hierarchy such as /usr/man.

**3.1.2.2. Tipee whatis printf. Como podemos ver, existen resultados en más de una sección.**

**Punto 4.1.1.1 si es necesario y busque como indicar en qué sección buscar).**

mati@mati-VirtualBox:~$ whatis printf

printf (1) - format and print data

printf (3) - formatted output conversion

**3.1.2.4. Al escribir por ejemplo “cd /bin” nos desplazamos hacia el directorio /bin. ¿Pero que hace exactamente “cd”? Tipee “whatis cd”.**

mati@mati-VirtualBox:~$ whatis cd

cd: nothing appropriate.

**3.1.3. whereis**

**3.1.3.1. Investigue que hace el comando whereis.**

**3.1.3.2. Tipee “whereis ls”, “whereis socket” y “whereis printf” .**

mati@mati-VirtualBox:~$ whereis ls

ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz

mati@mati-VirtualBox:~$ whereis ls

ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz

**3.1.3.3. Del punto 3.1.2.4 seguimos sin tener una descripción formal de “cd”. Tipee “whereis cd” y vea que sucede.**

mati@mati-VirtualBox:~$ whereis cd

cd:

**3.1.4. help**

**3.1.4.1. Investigue que hace el comando help tipeando “help help”.**

mati@mati-VirtualBox:~$ help help

help: help [-dms] [pattern ...]

Display information about builtin commands.

Displays brief summaries of builtin commands. If PATTERN is

specified, gives detailed help on all commands matching PATTERN,

otherwise the list of help topics is printed.

**3.1.4.2. Tipee “help cd”.**

mati@mati-VirtualBox:~$ help cd

cd: cd [-L|[-P [-e]] [-@]] [dir]

Change the shell working directory.

Change the current directory to DIR. The default DIR is the value of the

HOME shell variable.

The variable CDPATH defines the search path for the directory containing

DIR. Alternative directory names in CDPATH are separated by a colon (:).

A null directory name is the same as the current directory. If DIR begins

with a slash (/), then CDPATH is not used.

If the directory is not found, and the shell option `cdable\_vars' is set,

the word is assumed to be a variable name. If that variable has a value,

its value is used for DIR.

**3.1.4.3. ¡Finalmente logramos el objetivo! Dados los resultados del punto 3.1.2.4, 3.1.3.2, 3.1.3.3 y este punto, ¿Qué diferencia existe (no funcionalmente hablando) entre “cd” y por ejemplo “ls”? ¿Cuál de estos dos comandos es un “built-in command”?**

**3.1.5. apropos**

**3.1.5.1. Investigue que hace el comando apropos.**

mati@mati-VirtualBox:~$ apropos

apropos what?

**3.1.5.2. Supongamos que estamos buscando una función de C que se encarga de suspender la ejecución del proceso que la llama por un tiempo determinado. Tipee “apropos suspend” y vea si encuentra una función que cumpla con tales características.**

mati@mati-VirtualBox:~$ apropos suspend

aio\_suspend (3) - wait for asynchronous I/O operation or timeout

fsfreeze (8) - suspend access to a filesystem (Ext3/4, ReiserFS, JFS, XFS)

gai\_suspend (3) - asynchronous network address and service translation

pasuspender (1) - Temporarily suspend PulseAudio

rt\_sigsuspend (2) - wait for a signal

sigsuspend (2) - wait for a signal

sleep.conf.d (5) - Suspend and hibernation configuration file

systemd-sleep.conf (5) - Suspend and hibernation configuration file

systemd-suspend-then-hibernate.service (8) - System sleep state logic

systemd-suspend.service (8) - System sleep state logic

usleep (3) - suspend execution for microsecond intervals

**3.1.6. info: es un programa para leer documentación, entre la cual se incluyen tutoriales para efectuar distintas tareas en Linux. Este se compone de una estructura del tipo árbol, dividido en nodos de información. Cada nodo describe un tópico específico con un determinado nivel de detalle, el mismo se encuentra señalado con un \* (asterisco) y se puede acceder a él posicionando el cursor encima y teclando <enter>.**

**3.1.6.1. Investigue un poco más el comando info tipeando “man info”.**

INFO(1) User Commands INFO(1)

NAME

info - read Info documents

SYNOPSIS

info [OPTION]... [MENU-ITEM...]

DESCRIPTION

Read documentation in Info format.

Frequently-used options:

-a, --all

use all matching manuals

-k, --apropos=STRING

look up STRING in all indices of all manuals

-d, --directory=DIR

add DIR to INFOPATH

-f, --file=MANUAL

specify Info manual to visit

-h, --help

display this help and exit

--index-search=STRING

go to node pointed by index entry STRING

-n, --node=NODENAME

specify nodes in first visited Info file

-o, --output=FILE

output selected nodes to FILE

-O, --show-options, --usage

go to command-line options node

--subnodes

recursively output menu items

-v, --variable VAR=VALUE

assign VALUE to Info variable VAR

--version

display version information and exit

-w, --where, --location

print physical location of Info file

The first non-option argument, if present, is the menu entry to start from; it is searched for in all 'dir' files along

INFOPATH. If it is not present, info merges all 'dir' files and shows the result. Any remaining arguments are treated as

the names of menu items relative to the initial node visited.

For a summary of key bindings, type H within Info.

**3.1.6.2. Ingrese al programa info tipeando “info”.**

File: dir, Node: Top, This is the top of the INFO tree.

This is the Info main menu (aka directory node).

A few useful Info commands:

'q' quits;

'H' lists all Info commands;

'h' starts the Info tutorial;

'mTexinfo RET' visits the Texinfo manual, etc.

\* Menu:

Basics

\* Common options: (coreutils)Common options.

\* Coreutils: (coreutils). Core GNU (file, text, shell) utilities.

\* Date input formats: (coreutils)Date input formats.

\* Ed: (ed). The GNU line editor

\* File permissions: (coreutils)File permissions.

Access modes.

\* Finding files: (find). Operating on files matching certain criteria.

Compression

\* Gzip: (gzip). General (de)compression of files (lzw).

Development

\* SSIP: (ssip). Speech Synthesis Interface Protocol.

\* Speech Dispatcher: (speech-dispatcher).

Speech Dispatcher.

DOS

-----Info: (dir)Top, 212 lines --Top---------------------------------------------------------------------------

**3.1.6.3. Para navegar entre los nodos de información, algunas opciones son:**

**3.1.6.3.1. u: desplaza al nodo superior.**

**3.1.6.3.2. n: desplaza al nodo siguiente.**

**3.1.6.3.3. p: desplaza al nodo previo.**

### 3.2. Teclado / Terminales

#### 3.2.1. ¿Qué sucede si tecleo cat /e <tab> p <tab>? (donde tab es la tecla tabulación). Presione <tab> nuevamente ¿Qué pasó ahora?

El presionar *<tab>* se autocompleta la carpeta */etc*. Al apretar <tab> la segunda vez no sucede nada. Queda el siguiente path:

cat /etc/p

Al apretar *<tab>* una segunda vez aparecen estas opciones:

pam.conf pm/ prosody/

pam.d/ pnm2ppa.conf protocols

papersize polkit-1/ pulse/

passwd popularity-contest.conf purple/

passwd- ppp/ python/

pcmcia/ printcap python2.7/

perl/ profile python3/

pki/ profile.d/ python3.8/

#### 3.2.2. ¿Qué sucede si tecleo cat /e <tab> pas <tab>?

Se autocompleta el siguiente path:

cat /etc/passwd

#### 3.2.3. En este punto analizaremos las distintas terminales que hay en un sistema GNU/Linux. Ejecute los siguientes comandos e indique cuál fue el resultado:

**3.2.3.1. who**

sfn@PC:~$ who

sfn :0 2020-05-09 14:28 (:0)

**3.2.3.2. Presione la tecla <alt>, y sin soltarla presione cualquiera de las teclas de función. En la pantalla debería aparecer el login del sistema, de lo contrario, ejecute el paso nuevamente presionando otra tecla de función. Si ya tiene el login del sistema vuelva a loguearse.**

Con el SO Ubuntu fue necesario presionar las teclas <ctrl> + <alt> + tecla de función para que apareciese el login del sistema.

**3.2.3.3. Ejecute nuevamente el comando who. ¿Qué diferencias encuentra con la primera vez que lo ejecutó?**

Apareció una nueva línea, además de la anterior:

sfn@PC:~$ who

sfn :0 2020-05-09 14:28 (:0)

sfn tty5 2020-05-09 16:28

**3.2.3.4. Ejecute el comando whoami ¿qué muestra?, ¿Qué diferencias tiene con el comando ejecutado en el punto anterior?**

En vez de mostrar tres columnas de datos, sólo muestra el primero (el nombre de usuario):

sfn

**3.2.3.5. Repita el paso 3.2.3.2 y el 3.2.3.3 hasta que no encuentre ninguna sesión para abrir.**

Sólo pudo loguearse con terminales abiertas con las teclas de función f3, f4, f5 y f6.

Las terminales f7, f8 y f9 abrieron pero solo mostraron un cursor, nunca dieron la posibilidad de loguearse.

**3.2.3.6. Una vez terminado el punto anterior, Ud. se encontrará sesionado en el sistema como mínimo seis veces. Lo que acaba de hacer es abrir seis terminales virtuales (que podrían ser usadas por distintos usuarios, con diferentes perfiles), en la misma máquina. Así como existen terminales virtuales dentro del mismo equipo, si Ud. cuenta con una red, o con terminales tipo serie, podría abrir tantas sesiones de trabajo como Ud. quiera o necesite.**

**3.2.3.7. “Todo en Linux es un archivo”, y las terminales no son la excepción. Cada Terminal está representada por un archivo llamado ttyx donde x es un número de Terminal, y dichos archivos se encuentran dentro del directorio /dev. Investigue el comando “tty”.**

Según man:

tty - print the file name of the terminal connected to standard input.

Al ejecutar este comando en una terminal abierta desde la GUI del sistema recibimos:

sfn@PC:~$ tty

/dev/pts/0

Al ejecutarlo en una de las terminales abiertas antes:

sfn@PC:~$ tty

/dev/tty3

Haciendo una mínima investigación, la diferencia en los “directorios” (*/pts* y */tty*) parece darse por la diferencia una “pseudoterminal” y una “terminal virtual”.

En el primer caso, [Wikipedia nos dice](https://en.wikipedia.org/wiki/Pseudoterminal) que fueron introducidas con la computadora DEC PDP-6 en 1967. Una pseudo terminal sería en realidad “dos pseudo-dispositivos: uno, el *esclavo*, emula un dispositivo terminal físico (hardware) de ingreso de texto, y otro, el *maestro*, provee los medios por los que el proceso que emula la terminal controla al esclavo”. Ejecutando *man pts* se puede descubrir que “when a process opens /dev/ptmx, it gets a file descriptor for a pseudoterminal master (PTM), and a pseudoterminal slave (PTS) device is created in the /dev/pts directory”.

En cambio, el path */dev/tty#* corresponde, [según la doc del kernel de Linux](https://www.kernel.org/doc/Documentation/admin-guide/devices.txt), a las consolas virtuales.

**3.2.3.8. Tipee “whatis echo” para saber rápidamente qué hace el comando echo. Luego asegúrese de estar logueado en la 1er y 2da Terminal, y desde la 1er terminal tipee “echo hola! > /dev/tty2”. ¿Qué pasó? (más adelante aprenderá en detalle el uso del ‘>’).**

sfn@PC:~$ whatis echo

echo (1) - display a line of text

O sea, muestra una línea de texto.

Al intentar ejecutar el comando en la consola virtual *tty4* “apuntando” a *tty3* recibimos este mensaje de error:

-bash: /dev/tty3: Permission denied

Probamos con las restantes consolas virtuales y ocurrió lo mismo. Intentamos entonces usar las terminales de Ubuntu/Gnome. Abrimos dos terminales, lo que generó dos paths: */dev/pts/0* y */dev/pts/1*. Desde la primera terminal (*/0*) ejecutamos:

sfn@PC:~$ echo hola! > /dev/pts/1

y en la segunda terminal obtuvimos:

sfn@PC:~$ hola!

### 3.3. Sistema

**3.3.1. Investigue los comandos:**

**3.3.1.1. poweroff**

Permite apagar el sistema. En el manual se especifica diferentes opciones para el comando: -f

Fuerza el apagado del equipo

**3.3.1.2. reboot**

Reinicia el equipo, mismos comandos que poweroff

### 3.4. Usuarios

**3.4.1. ¿Qué es la cuenta de superusuario (root) y para qué se utiliza? (probablemente tenga que buscarlo en internet).**

*La cuenta superusuario en Linux posee todos los privilegios y permisos para realizar acciones sobre el sistema.*

**3.4.2. Investigue que sucede con la cuenta de root en Ubuntu (el sistema operativo de uso en la cátedra). Investigue como realizar comandos a nombre del usuario root.**

*Por lo general, los sistemas operativos contemplan el uso de sólo un usuario, el cual tiene permisos de administrador. En Linux las cosas se manejan de una forma particular, se separa la cuenta de usuario común de la de superusuario y es eso lo que conocemos como root.*

*Para poder realizar comandos como si fuéramos root debemos usar sudo. La función de sudo es permitir que un usuario normal pueda ejecutar comandos como si fuera administrador.*

**3.4.3. Investigue los comandos “su” y “login”**

*El comando “su” funciona para ingresar al sistema utilizando cualquier otro usuario definido.*

*El comando “login” inicia una nueva sesión con el usuario definido. Para utilizar el comando “login” es necesario tener permisos de superusuario, este obstáculo se sortea utilizando: sudo login <user>.*

**3.4.4. Loguearse como superusuario (root), y realizar los siguientes pasos (éste punto no puede ser realizado en el laboratorio):**

jjlopezal@Juan-Caece:~$ sudo login root

[sudo] contraseña para jjlopezal:

Contraseña:

Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-29-generic x86\_64)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

1 actualización se puede instalar inmediatamente.

0 de estas actualizaciones son una actualización de seguridad.

Para ver estas actualizaciones adicionales ejecute: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

Último inicio de sesión: mié may 6 22:33:32 -03 2020 en pts/0

root@Juan-Caece:~#

**3.4.5. adduser/addgroup**

**3.4.5.1. Investigue que hace el comando adduser/addgroup.**

*Los comandos “adduser” y “addgroup” añaden usuarios y grupos al sistema de acuerdo a las opciones de la*

*línea de órdenes y a la configuración en “/etc/adduser.conf”.*

**3.4.5.2. Cree un nuevo usuario, cree un nuevo grupo, y agregue el usuario a ese grupo.**

# Creo al usuario matias

root@Juan-Caece:/home/jjlopezal# useradd matias

# Verifico que se creo correctamente el usuario

root@Juan-Caece:/home/jjlopezal# getent passwd | grep matias

matias:x:1001:1001::/home/matias:/bin/sh

#Creo el grupo caece

root@Juan-Caece:/home/jjlopezal# groupadd caece

#Agrego al usuario matias al grupo caece

root@Juan-Caece:/home/jjlopezal# adduser matias caece

Añadiendo al usuario `matias' al grupo `caece' ...

Añadiendo al usuario matias al grupo caece

Hecho.

# Verifico que matias se agrego al grupo caece

root@Juan-Caece:/home/jjlopezal# groups matias

matias : matias caece

**3.4.6. deluser/delgroup**

**3.4.6.1. Investigue que hace el comando deluser/delgroup.**

*Los comandos “deluser” y “delgroup” eliminan usuarios y grupos del sistema de acuerdo a las opciones en*

*línea de órdenes y a la configuración en /etc/deluser.conf y /etc/adduser.conf.*

**3.4.6.2. Borre el usuario creado anteriormente (incluyendo el borrado de su directorio en home y todos sus archivos).**

root@Juan-Caece:/# deluser --remove-all-files matias

Eliminando al usuario `matias' ...

Aviso: el grupo `matias' no tiene más miembros.

Hecho.

**3.4.7. Investigue cómo hacer para saber todos los grupos a los que pertenece un usuario.**

*Con el comando groups se pueden listar todos los grupos a los que pertenece el usuario.*

*Por ejemplo:*

root@Juan-Caece:/home/jjlopezal# groups matias

### 3.5. Archivos

**3.5.1. ¿Qué hacen los siguientes comandos?**

**3.5.1.1. cp**

Copy path. Copia la dirección/directorio de un archivo. Todo en linux es un archivo.

mati@mati-VirtualBox:~$ cp

cp: missing file operand

Try 'cp --help' for more information.

mati@mati-VirtualBox:~$ cp --help

Usage: cp [OPTION]... [-T] SOURCE DEST

or: cp [OPTION]... SOURCE... DIRECTORY

or: cp [OPTION]... -t DIRECTORY SOURCE...

*Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.*

**3.5.1.2. mv**

Move. Renombra hacia otro directorio o mueve hacia otro directorio.Todo en linux es un archivo.

mati@mati-VirtualBox:~$ mv --help

Usage: mv [OPTION]... [-T] SOURCE DEST

or: mv [OPTION]... SOURCE... DIRECTORY

or: mv [OPTION]... -t DIRECTORY SOURCE...

*Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.*

**3.5.1.3. rm**

Remove. Borra/desvincula archivos/carpetas. Todo en linux es un archivo.

mati@mati-VirtualBox:~$ rm --help

Usage: rm [OPTION]... [FILE]...

Remove (unlink) the FILE(s).

**3.5.1.4. scp**

Es la copia de archivos segura de SSH. Copia archivos entre anfitriones(hosts) en una red(network).

mati@mati-VirtualBox:~$ scp --help

unknown option -- -

usage: scp [-346BCpqrTv] [-c cipher] [-F ssh\_config] [-i identity\_file]

[-J destination] [-l limit] [-o ssh\_option] [-P port]

[-S program] source ... target

mati@mati-VirtualBox:~$ scp -help

unknown option -- h

usage: scp [-346BCpqrTv] [-c cipher] [-F ssh\_config] [-i identity\_file]

[-J destination] [-l limit] [-o ssh\_option] [-P port]

[-S program] source ... target

mati@mati-VirtualBox:~$ scp

usage: scp [-346BCpqrTv] [-c cipher] [-F ssh\_config] [-i identity\_file]

[-J destination] [-l limit] [-o ssh\_option] [-P port]

[-S program] source ... target

**3.5.1.5. telnet**

Telnet es un comando de linux para comunicarse con otro anfitrión usando el protocolo telnet. Similar a SSH, su seguridad se basa en un cifrado por clave privada compartida.

mati@mati-VirtualBox:~/Documents$ man telnet

NAME

telnet — user interface to the TELNET protocol

SYNOPSIS

telnet [-468ELadr] [-S tos] [-b address] [-e escapechar]

[-l user] [-n tracefile] [host [port]]

DESCRIPTION

The telnet command is used for interactive communication with an‐

other host using the TELNET protocol. It begins in command mode,

where it prints a telnet prompt ("telnet> "). If telnet is in‐

voked with a host argument, it performs an open command implic‐

itly; see the description below.

**3.5.1.6. ssh**

Secure Shell.

Línea de comandos remota hacia otras máquinas, reales o virtuales.

Posee encripción simétrica, antisimétrica y utiliza hashing para seguridad.

mati@mati-VirtualBox:~$ ssh

usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind\_interface]

[-b bind\_address] [-c cipher\_spec] [-D [bind\_address:]port]

[-E log\_file] [-e escape\_char] [-F configfile] [-I pkcs11]

[-i identity\_file] [-J [user@]host[:port]] [-L address]

[-l login\_name] [-m mac\_spec] [-O ctl\_cmd] [-o option] [-p port]

[-Q query\_option] [-R address] [-S ctl\_path] [-W host:port]

[-w local\_tun[:remote\_tun]] destination [command]

**3.5.1.7. touch**

Actualiza el tiempo de acceso y modificación de cada archivo de entrada al tiempo actual. También da la posibilidad de crear los nuevos archivos.

<https://www.geeksforgeeks.org/touch-command-in-linux-with-examples/>

mati@mati-VirtualBox:~$ touch --help

Usage: touch [OPTION]... FILE...

Update the access and modification times of each FILE to the current time.

A FILE argument that does not exist is created empty, unless -c or -h

is supplied.

A FILE argument string of - is handled specially and causes touch to

change the times of the file associated with standard output.

mati@mati-VirtualBox:~$ touch

touch: missing file operand

Try 'touch --help' for more information.

**3.5.2. A la hora de referirse a archivos, se puede usar tanto su dirección relativa (al directorio en el que se encuentra situado) o absoluta. Sitúese como root dentro del directorio /root. Luego copie el archivo .bashrc a la ruta absoluta /var/.bashrc. Ahora, mueva ese archivo desde esa dirección hasta /home/.bashrc sin desplazarse del directorio inicial (/root).**

mati@mati-VirtualBox:~$ sudo bash

root@mati-VirtualBox:/home/mati#

root@mati-VirtualBox:/home/mati# cd

root@mati-VirtualBox:~#

root@mati-VirtualBox:~# cp .bashrc /var

para fijarse si está:

root@mati-VirtualBox:~# cd /var

root@mati-VirtualBox:/var# ls -a

### 3.6. Permisos

#### 3.6.1. Cree un archivo tipeando “ls > archivo”.

Creado (no hubo output).

#### 3.6.2. Tipee ls –l en dicho directorio: los primeros 10 caracteres corresponden a los permisos. Investigue cómo se estructuran los permisos de un archivo (puede tipear info y luego ir a la sección de “permisos de archivo” o “file permissions”).

Al tipear *ls -l* se imprime en pantalla el listado de archivos del directorio (~/, la homa del usuario actual, *sfn*) con el siguiente formato:

-rw-r--r-- 1 sfn sfn 0 jun 13 2015 a.out

-rw-rw-r-- 1 sfn sfn 1004 may 11 01:29 archivo

lrwxrwxrwx 1 sfn sfn 15 mar 5 2018 'Backup (disco)'

…

Los caracteres *-rw-rw-r--* representarían los permisos de *archivo*.

Indagando en *info* y en internet, los permisos de acceso a archivos/carpetas forman parte de los “bits de modo” de un archivo/directorio. Estos bits incluyen “bits de permisos de archivos” y “bits de modo especial” (que afectan solo a algunos archivos).

En los bits de permisos es posible configurar tres tipos:

1. **lectura** (en directorios esto significa lista el contenido)
2. **escritura** (cambio) (en directorios significa crear y borrar archivos que estén adentro)
3. **ejecución** como programa (en directorios signifca acceder a archivos).

Hay tres categorías de usuarios (clases) para los que se definen estos permisos:

1. el **propietario** (owner)
2. otros usuarios que estén en el **grupo** del archivo
3. todo el **resto** de los usuarios

El propietario y el grupo son dados a un archivo cuando este es creado (en general son el usuario actual y el grupo es el del directorio donde está el archivo; pero esto puede variar según el SO).

Ahora bien, estos bits pueden representarse usando “la notación simbólica, que permite representar permisos en una serie de 10 caracteres” ([Wikipedia](https://es.wikipedia.org/wiki/Permisos_de_acceso_a_archivos)). En esta notación,

* El **primer carácter** indica el tipo de archivo ('-' denota un archivo regular, 'd' un directorio, 'b' un archivo especial de bloques, 'l' un enlace simbólico, etc.).
* Luego, cada clase de permisos se representa por conjuntos de tres caracteres.
  + El primer conjunto (caracteres 2 a 4) representa la clase de usuario.
  + El segundo conjunto (caracteres 5 a 7) representa la clase de grupo.
  + El tercer y último conjunto (caracteres 8 a 10) de tres caracteres representa la clase del resto.

Cada uno de los tres caracteres representa los permisos de lectura, escritura y ejecución respectivamente:

* 'r' si el bit de lectura está asignado, '-' en caso contrario.
* 'w' si el bit de escritura está asignado, '-' en caso contrario.
* 'x' si el bit de ejecución está asignado, '-' en caso contrario.

En el caso de nuestro archivo *archivos*, tenemos entonces:

* - archivo regular
* rw- lecutra y escritura permitidas al propietario/usuario; no ejecución
* rw- ídem para el grupo
* r-- sólo se permite lectura para el resto de los usuarios

#### 3.6.3. chmod/chown

##### 3.6.3.1. Investiga qué hacen esos comandos.

**chmod**

Modifica los bits de modo de un archivos. Según las descripción de su man: “*chmod* changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits”.

La “representación simbólica” de los nuevos permisos tienen que tener el patrón *[ugoa...][[-+=][perms...]...]*, donde

* [ugoa] controla a qué usuarios se les cambiará el acceso al archivo:
  + u: el propietario
  + g: otros usuarios dle grupo
  + o: otros usuarios que no son del grupo
  + a: todos los usuarios
* [-+=] son los operadores
  + + hace que los nuevos permisos se agreguen al archivo.
  + - hace que sean removidos de los del archivo.
  + = hace que se agreguen los seleccionados y que los permisos no mencionados sean removidos.
* [perms..] puede ser varias o ninguna letra del set *rwxXst*, o una sola letra del set [ugo]. Las letras rwxXst cambian los permisos de los usuarios seleccionados:
  + r: read
  + w: write
  + x: execute (or search for directories)
  + X: execute/search only if the file is a directory or already has execute permission for some user
  + s: set user or group ID on execution
  + t: restricted deletion flag or sticky bit

**chown**

Permite cambiar el owner y el grupo de un archivo. Según su man: “chown changes the user and/or group ownership of each given file. If only an owner (a user name or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed).

##### 3.6.3.2. Haga que el archivo “archivo” creado anteriormente pueda ser modificado por cualquier usuario.

Ejecutando el siguiente comando:

sfn@PC:~$ chmod a+w archivo

Verificamos el efecto:

sfn@PC:~$ ls -l archivo

-rw-rw-**rw-** 1 sfn sfn 1004 may 09 16:29 archivo

##### 3.6.3.3. Compruebe que logró el punto anterior logueandose en otra Terminal con otro usuario y modificando dicho archivo (tipeando nuevamente “ls > archivo”).

Si bien ya lo habíamos verificado con ls, ejecutamos los siguientes comandos en otra terminal:

sfn@PC:~$ echo "hola" > archivo

sfn@PC:~$ cat archivo

hola

##### 3.6.3.4. Loguéese con el usuario original y quite los todos los permisos del archivo (lectura, escritura y ejecución) a todos los usuarios distintos del dueño y de los que pertenecen al mismo grupo. Luego, haga que el nuevo dueño del archivo sea el otro usuario.

Ejecutamos en el usuario original (sfn):

sfn@PC:~$ chmod g-rw,o-rw archivo

sfn@PC:~$ ls -l archivo

-rw------- 1 sfn sfn 5 may 09 17:18 archivo

Luego cambiamos el propietario:

sfn@PC:~$ sudo chown **visitante** archivo

[sudo] password for sfn:

sfn@PC:~$ ls -l archivo

-rw------- 1 **visitante** sfn 5 may 09 17:18 archivo

##### 3.6.3.5. ¿Cómo haría para volver a poseer dicho archivo sin loguearse con el nuevo dueño del archivo?

Usando sudo con el usuario original se puede recobrar la posesión:

sfn@PC:~$ sudo chown **sfn** archivo

sfn@PC:~$ ls -l archivo

-rw------- 1 **sfn** sfn 5 may 09 17:18 archivo

##### 3.6.3.6. Investigue qué es el “SUID bit” (busque en “man chmod”).

SUID: Set user ID (setear la identificación del usuario)

Es un permiso de accesos correspondiente a archivos y/o directorios en un sistema Unix. Es útil para permitir ejecutar binarios que requieren un rango de privilegios a usuarios comunes.

##### 3.6.3.7. Investigue cómo aplica la estructura de los permisos a los directorios.

En ubuntu (UNIX en general) se divide en tres la estructura de permisos de directorios:

read (lectura), write (escritura) y execute (ejecución).

Lectura: se pueden ver los archivos contenidos en el directorio

Escritura: se permite el agregado y remoción de archivos al directorio.

Ejecución: permite la ejecución de un programa dentro del directorio en el que se contiene. Es importante porque otros usuarios pueden entonces leerlo y escribirlo para colaborar, pero evita la ejecución indeseada (accidental o maliciosa) y los efectos que pueda tener la ejecución del mismo archivo.

Los tipos de permisos tienen todos asociados un valor:

Lectura: 4

Escritura: 2

Ejecución: 1

Las distintas combinaciones de permisos (incluyendo sin permisos, 0) se señalizan mediante las sumas de estos números. Por ejemplo, si un archivo para cierto usuario tiene todos los permisos, tenemos 4+2+1=7. Si solamente puede leer y escribir, tenemos 4+2=6.

Como UNIX es un sistema multiusuario, tiene sentido que el mismo distinga también entre jerarquías de usuarios. Las tres categorías son:

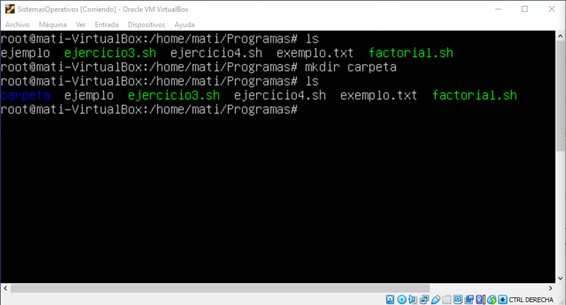
Owner(propietario)

Group(grupo)

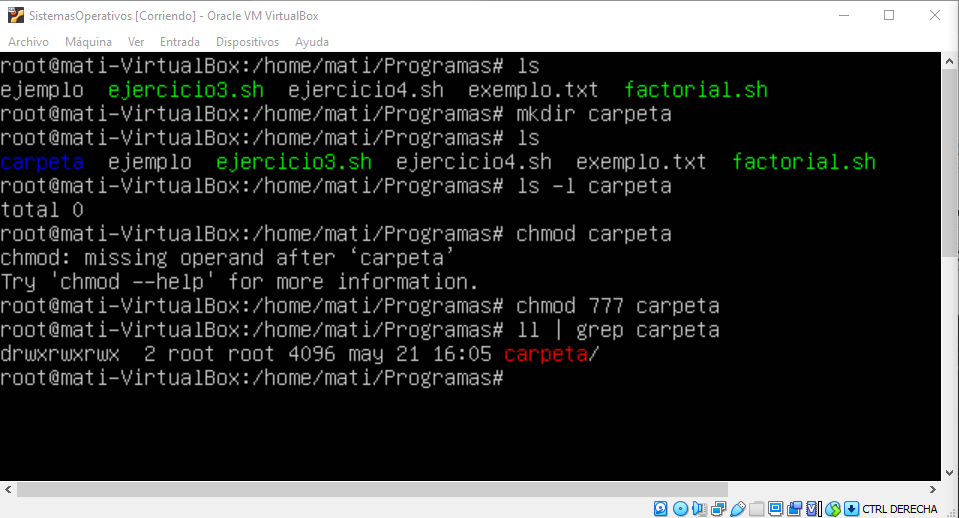
Other(otros)

Si un archivo por ejemplo tiene asociado la combinación de permisos 777, significa que la lectura, escritura y ejecución está permitido para cualquier usuario del sistema. Un usuario no necesita ser una persona física, por ejemplo puede referirse a un proceso como usuario.

##### 3.6.3.8. Loguéese como root en otra Terminal y cree un directorio tipeando “mkdir /undir”.

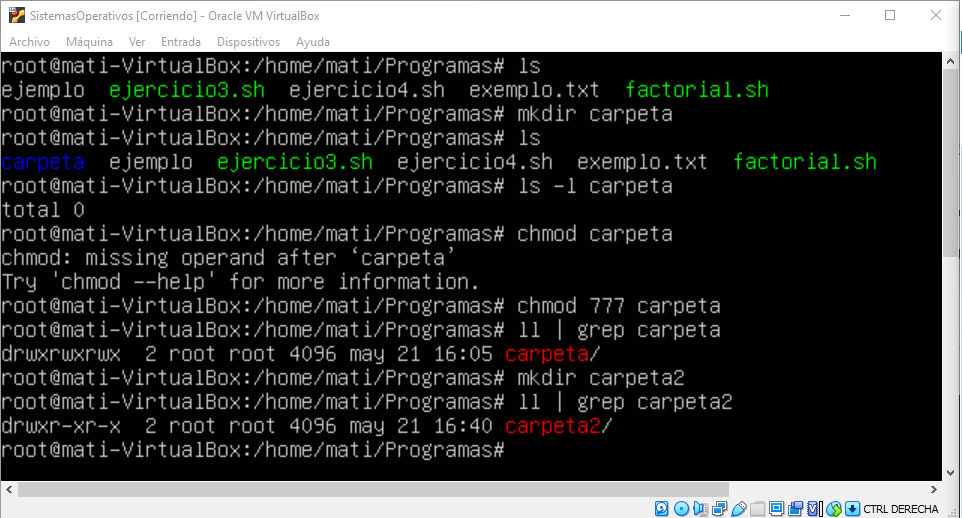


##### 3.6.3.9. Haga que cualquier usuario tenga todo tipo de permisos sobre ese directorio.

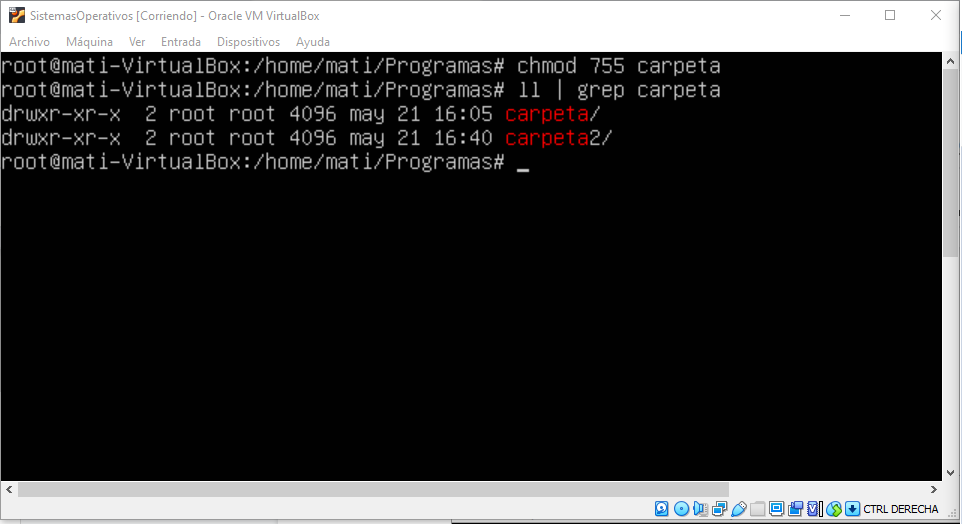
****

##### 3.6.3.10. Deshaga lo que acaba de hacer, y cree el subdirectorio “subdir” dentro de “/undir”.

vemos primero con que permisos se creó una carpeta (ver carpeta2)

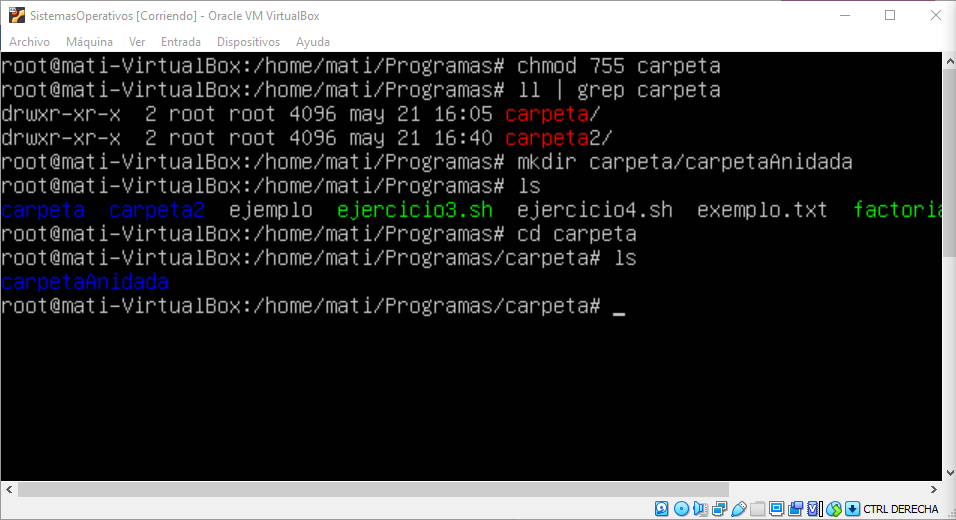


volvemos a configurar carpeta/ con los permisos iniciales



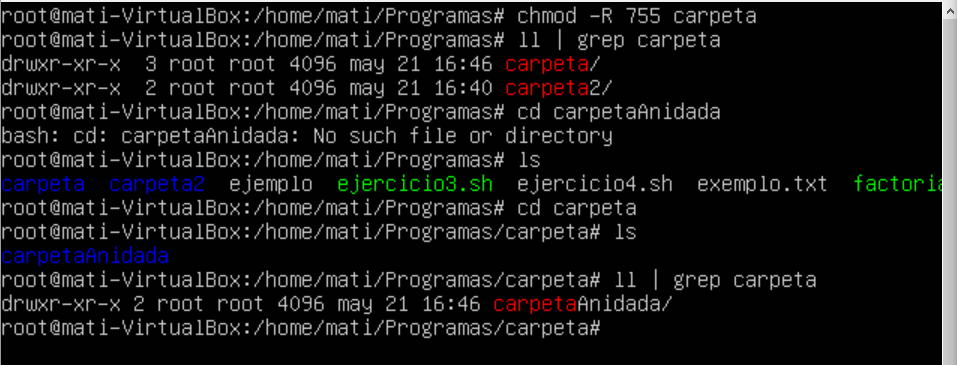
drwxr-xr-x: directory read write execute, read-write, read-write

carpeta anidada:

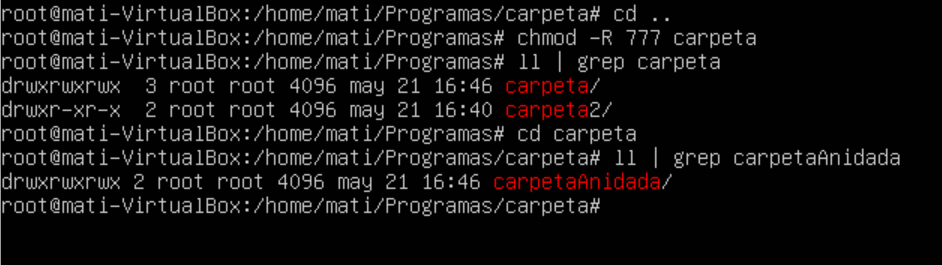


##### 3.6.3.11. Investigue cómo cambiar los permisos de manera recursiva sobre /undir para que todos sus archivos, subdirectorios y archivos dentro de los subdirectorios se vean afectados.

simplemente agregando el modificador -R a chmod



ahora volvemos a poner en carpeta y carpetaAnidada los permisos 777

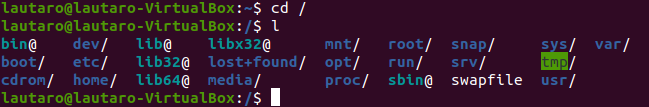


### 3.7. Directorios

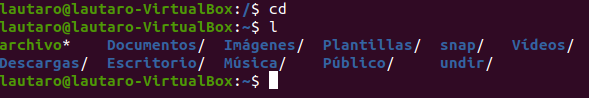
**3.7.1. ¿Para qué se usa el comando cd? Ejecute las siguientes variantes de cd y observe cuál fue el resultado obtenido:**

El comando cd (change directory) es una orden para mostrar el directorio actual y también para cambiarlo.

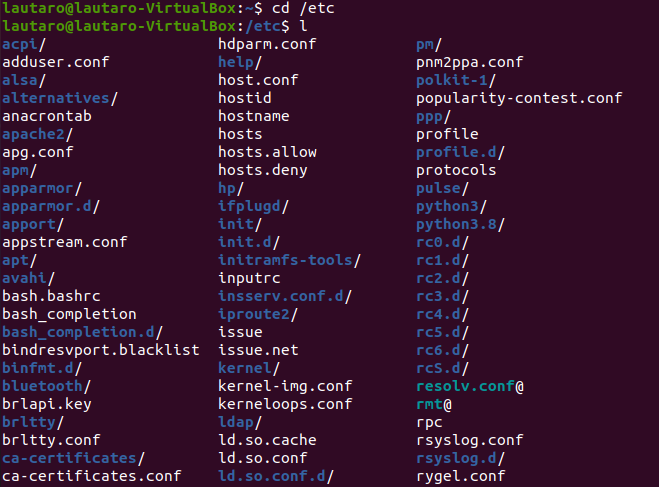
**3.7.1.1. cd /**

****

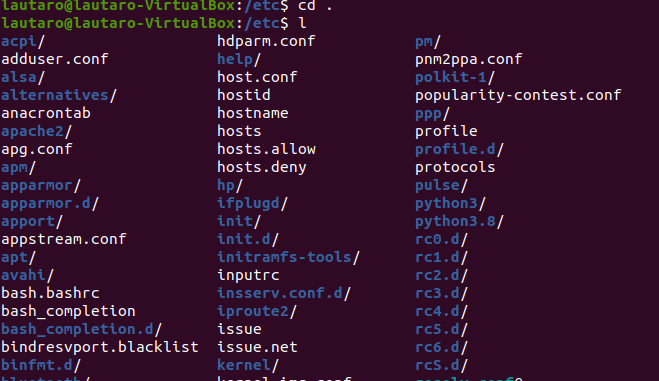
**3.7.1.2. cd**

****

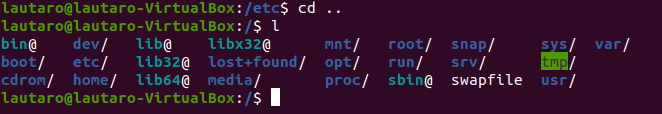
**3.7.1.3. cd /etc**

****

**3.7.1.4. cd .**

****

**3.7.1.5. cd ..**

****

**3.7.2. mkdir, rmdir, rm (nuevamente)**

**3.7.2.1. Investigue dichos comandos.**

mkdir: es un comando para crear un nuevo subdirectorio o carpeta del sistema de archivos.

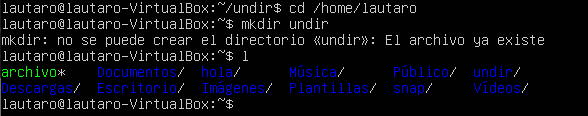
rmdir: comando que elimina directorios vacíos.

rm: es un comando usado para eliminar archivos y directorios del sistema de archivos.

**3.7.2.2. Borre un directorio que no se encuentra vacío.**

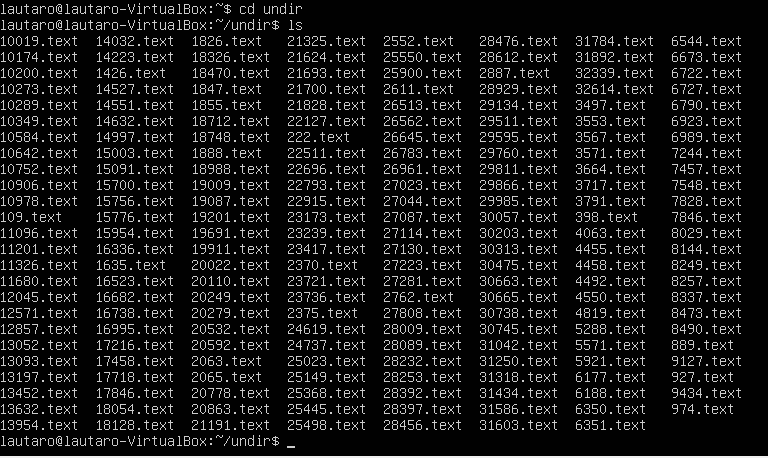
****

**3.7.2.3. Dentro de /home/<usuario> cree el directorio undir.**

****

**(**Ya lo habia creado pero no hice un recorte, por eso aparece “El archivo ya existe”)

**3.7.2.4. Ingrese a dicho directorio, y tipee lo siguiente para crear muchos archivos “while (true) do ps > $RANDOM.text; done;”. Tipee ctrl.+c luego de 5 seg para finalizar el comando. Luego tipee ls para corroborar la creación de los archivos.**

****

**3.7.2.5. Tipee “rm \*” e investigue que pasó.**



### 3.8. Filtros

**3.8.1. ¿Cuál es la diferencia de los comandos more, less y cat? Cree un archivo de texto tipeando “ps –fea > texto” y visualícelo con los distintos comandos.**

El comando ‘cat’ imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. Al igual que ‘cat’, ‘more’ permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que ‘more’ pagina los resultados. El comando ‘less’ es el más completo de los tres, pues puede hacer todo lo que hace ‘more’ añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder).

**3.8.1.1. Investigue como buscar cadenas de texto cuando se visualiza un archivo con less. ¿Y cómo se hace para repetir la búsqueda? ¿Y para repetir la búsqueda hacia atrás? Esto le servirá cuando lea páginas del man! (ya que se leen mediante el less).**

Para buscar una cadena de texto, se utilizan los mismos comandos que en el editor de texto vi, ya que less esta basado en este editor. Para buscar la cadena “jjl” debemos escribir “/” el comando seria : /jjl . Para seguir buscando las próximas ocurrencias se debe presionar “n”, para ir a la ocurrencia anterior lo hacemos con “N”. Si nos encontramos en el final del archivo podemos utilizar “?” este comando buscará de abajo para arriba, de la misma manera buscamos las próximas ocurrencias con “n” o en sentido inverso “N”.

**3.8.2. ¿Cuál es la diferencia entre tail y head?. ¿Qué hace la opción –f del comando tail?**

Los comandos head y tail permiten mostrar de forma parcial el contenido de un fichero. Como su nombre indica, head muestra las primeras líneas del fichero (la cabecera) y tail muestra las últimas líneas (la cola). La opción -f causará que la cola se repita para siempre, buscando nuevos datos al final de los archivos. Cuando aparecen nuevos datos, se imprimirán.

**3.8.2.1. Loguéese en una Terminal y tipee “echo > a.txt” para crear el archivo “a.txt”. Luego tipee “tail –f a.txt”.**

jjlopezal@Juan-Caece:~/caece$ echo > a.txt

jjlopezal@Juan-Caece:~/caece$ tail -f a.txt

**3.8.2.2. Desde otra Terminal, tipee “while (true) do date >> a.txt; sleep 2; done;”.**

jjlopezal@Juan-Caece:~/caece$ while (true) do date >> a.txt; sleep 2; done;

**3.8.2.3. Vuelva a la terminal anterior y vea lo que sucede.**

En la terminal que se ejecutó el comando tail, se puede ver en tiempo real las líneas que agrega el while en el archivo a.txt

**3.8.2.4. No se olvide de finalizar el comando de la 2da Terminal! (con ctrl.+c)**

**3.8.3. ¿Qué es lo que realiza el comando sort?**

El comando sort ordena líneas de los archivos de entrada.

**3.8.4. ¿Qué es lo que realiza el comando uniq?**

El comando uniq quita las líneas que están repetidas, dejando la salida con líneas que no están repetidas. Para que funcione correctamente los ficheros deben estar ordenados, ya que trabaja con líneas adyacentes.

**3.8.5. grep**

**3.8.5.1. ¿Para qué sirve?**

El comando grep se utiliza para hacer coincidir e imprimir un patrón de búsqueda o una expresión regular de un solo archivo o varios archivos de texto.

**3.8.5.2. Busque en el archivo “texto” todas las líneas que contengan la palabra “root”.**

jjlopezal@Juan-Caece:~/caece$ grep root texto

root 1 0 0 may08 ? 00:00:04 /sbin/init splash

root 2 0 0 may08 ? 00:00:00 [kthreadd]

root 3 2 0 may08 ? 00:00:00 [rcu\_gp]

.

.

.

**3.8.6 find**

**3.8.6.1. ¿Para qué sirve?**

El comando find se encarga de realizar búsquedas de ficheros y carpetas, desde la línea de comandos, en un árbol de directorios.

**3.8.5.1. Busque un patrón determinado en un directorio utilizando find y grep.**

En mi directorio cree los archivos “a1”, “a2”, “a3”, “a4”, “a5”, “a6” con el siguiente contenido:

==> a1 <==

Quiero ver si encuentra la palabra perro.

==> a2 <==

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

==> a3 <==

Quiero ver si encuentra la palabra perro.

==> a4 <==

Aca solo hay gatos

==> a5 <==

Aca no hay mascotas

==> a6 <==

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Quiero ver si encuentra la palabra perro.

Ahora en este directorio quiero ver todos los archivos que contienen la palabra perro utilizando los comandos find y grep.

jjlopezal@Juan-Caece:~/caece$ find -type f -exec grep -l "perro" {} \;

./a1

./a2

./a3

./a6

### 3.9. Redirecionamiento de E/S

**3.9.1. Antes de que un comando sea ejecutado, su entrada/salida estándar pueden ser redireccionados usando una notación especial del shell. Investíguelo tipeando “man bash” y llegando luego a la sección “REDIRECTIONS”.**

**DESCRIPTION**

Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file.

Bash also incorporates useful features from the Korn and C shells (ksh and csh).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification

(IEEE Standard 1003.1). Bash can be configured to be POSIX-conformant by default.

**Pipelines**

A pipeline is a sequence of one or more commands separated by one of the control operators | or |&. The format for a

pipeline is:

[time [-p]] [ ! ] command [ [|⎪|&] command2 ... ]

The standard output of command is connected via a pipe to the standard input of command2. This connection is performed

before any redirections specified by the command (see REDIRECTION below). If |& is used, command's standard error, in ad‐

dition to its standard output, is connected to command2's standard input through the pipe; it is shorthand for 2>&1 |.

This implicit redirection of the standard error to the standard output is performed after any redirections specified by

the command.

The return status of a pipeline is the exit status of the last command, unless the pipefail option is enabled. If

pipefail is enabled, the pipeline's return status is the value of the last (rightmost) command to exit with a non-zero

status, or zero if all commands exit successfully. If the reserved word ! precedes a pipeline, the exit status of that

pipeline is the logical negation of the exit status as described above. The shell waits for all commands in the pipeline

to terminate before returning a value.

If the time reserved word precedes a pipeline, the elapsed as well as user and system time consumed by its execution are

reported when the pipeline terminates. The -p option changes the output format to that specified by POSIX. When the

shell is in posix mode, it does not recognize time as a reserved word if the next token begins with a `-'. The TIMEFORMAT

variable may be set to a format string that specifies how the timing information should be displayed; see the description

of TIMEFORMAT under Shell Variables below.

When the shell is in posix mode, time may be followed by a newline. In this case, the shell displays the total user and

system time consumed by the shell and its children. The TIMEFORMAT variable may be used to specify the format of the time

information.

Each command in a pipeline is executed as a separate process (i.e., in a subshell). See COMMAND EXECUTION ENVIRONMENT for

a description of a subshell environment. If the lastpipe option is enabled using the shopt builtin (see the description

of shopt below), the last element of a pipeline may be run by the shell process.

When the shell is in posix mode, time may be followed by a newline. In this case, the shell displays the total user and

system time consumed by the shell and its children. The TIMEFORMAT variable may be used to specify the format of the time

information.

Each command in a pipeline is executed as a separate process (i.e., in a subshell). See COMMAND EXECUTION ENVIRONMENT for

a description of a subshell environment. If the lastpipe option is enabled using the shopt builtin (see the description

of shopt below), the last element of a pipeline may be run by the shell process.

**REDIRECTION**

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell.

Redirection allows commands' file handles to be duplicated, opened, closed, made to refer to different files, and can

change the files the command reads from and writes to. Redirection may also be used to modify file handles in the current

shell execution environment. The following redirection operators may precede or appear anywhere within a simple command

or may follow a command. Redirections are processed in the order they appear, from left to right.

**3.9.1.1. Redireccionando la salida estándar.**

**3.9.1.1.1. Ejecute el comando “ps -fea” y redirija su salida a un archivo llamado “salida.txt”.**

man ps

NAME

ps - report a snapshot of the current processes.

SYNOPSIS

ps [options]

DESCRIPTION

ps displays information about a selection of the

active processes. If you want a repetitive update of

the selection and the displayed information, use

top(1) instead.

mati@mati-VirtualBox:~/Documents$ ps -fea > salida.txt

mati@mati-VirtualBox:~/Documents$ ls

archivo salida.txt

mati@mati-VirtualBox:~/Documents$

mati@mati-VirtualBox:~/Documents$ ps -fea

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 10:30 ? 00:00:01 /sbin/init

root 2 0 0 10:30 ? 00:00:00 [kthreadd]

root 3 2 0 10:30 ? 00:00:00 [rcu\_gp]

root 4 2 0 10:30 ? 00:00:00 [rcu\_par\_g

root 5 2 0 10:30 ? 00:00:00 [kworker/0

root 6 2 0 10:30 ? 00:00:00 [kworker/0

…[muchas líneas]

mati 1812 1308 0 10:31 ? 00:00:00 /usr/libex

mati 1820 1308 0 10:31 ? 00:00:00 /usr/libex

mati 2099 1308 0 10:32 ? 00:00:00 /usr/libex

mati 2102 1643 0 10:32 ? 00:00:00 update-not

root 2394 2 0 10:37 ? 00:00:00 [kworker/0

root 2502 2 0 10:45 ? 00:00:00 [kworker/0

root 2504 2 0 10:47 ? 00:00:00 [kworker/u

mati 2508 2022 0 10:48 pts/0 00:00:00 ps -fea

**3.9.1.1.2. Ídem punto anterior, pero que se agregue la salida del comando al final del archivo.**

<https://www.cyberciti.biz/faq/linux-append-text-to-end-of-file/>

<https://www.cyberciti.biz/faq/linux-unix-appleosx-bsd-cat-command-examples/>

ps -fea >> salida.txt

**3.9.1.2. Redireccionando la entrada estándar.**

**3.9.1.2.1. El comando “grep cadena archivo” imprime las líneas de archivo que contengan “cadena”. Investigue cómo hacer para lograr el mismo resultado sin especificarle a grep un archivo (investigue si es necesario qué hace grep cuando no se le especifica un archivo).**

mati@mati-VirtualBox:~$ grep --help

Usage: grep [OPTION]... PATTERNS [FILE]...

Search for PATTERNS in each FILE.

Example: grep -i 'hello world' menu.h main.c

PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:

-E, --extended-regexp PATTERNS are extended regular expressions

-F, --fixed-strings PATTERNS are strings

-G, --basic-regexp PATTERNS are basic regular expressions

-P, --perl-regexp PATTERNS are Perl regular expressions

-e, --regexp=PATTERNS use PATTERNS for matching

-f, --file=FILE take PATTERNS from FILE

-i, --ignore-case ignore case distinctions in patterns and data

--no-ignore-case do not ignore case distinctions (default)

-w, --word-regexp match only whole words

-x, --line-regexp match only whole lines

-z, --null-data a data line ends in 0 byte, not newline

ENVIRONMENT

The behavior of grep is affected by the following

environment variables.

…

pruebita:

mati@mati-VirtualBox:~/Documents$ grep 508 salida.txt

mati@mati-VirtualBox:~/Documents$ grep 942 salida.txt

root 942 1 0 10:30 ? 00:00:00 /usr/sbin/gdm3

root 1300 942 0 10:31 ? 00:00:00 gdm-session-worker [pam/gdm-password]

<https://stackoverflow.com/questions/14654775/what-does-grep-do-when-a-directory-or-file-isnt-added>

### 3.10. Pipelines (Tuberías)

**3.10.1. El carácter | (pipe) se usa para conectar la salida estándar de un comando con la entrada estándar de otro. Investíguelo tipeando “man bash” y llegando luego a la sección “Pipelines” (o “tuberías” en castellano).**

Según el man, las pipelines son “una secuencia de uno o màs comandos separados por los operadores | o |&. El formato de una pipeline es: *[time [-p]] [ ! ] command [ [|⎪|&] command2 ... ]*. Básicamente, el output del primer comando es pasado, gracias a la tubería, como input del segundo comando. Cada comando es ejecutado como un proceso separado.

**3.10.2. Haciendo uso de ps y grep, liste todos los procesos del usuario root.**

Ejecutamos

sfn@PC:~$ ps -ef | grep root

Y obtuvimos

root 1 0 0 may10 ? 00:00:26 /sbin/init splash

root 2 0 0 may10 ? 00:00:00 [kthreadd]

root 3 2 0 may10 ? 00:00:00 [rcu\_gp]

root 4 2 0 may10 ? 00:00:00 [rcu\_par\_gp]

root 9 2 0 may10 ? 00:00:00 [mm\_percpu\_wq]

root 10 2 0 may10 ? 00:00:00 [ksoftirqd/0]

root 11 2 0 may10 ? 00:00:26 [rcu\_sched]

root 12 2 0 may10 ? 00:00:00 [migration/0]

root 13 2 0 may10 ? 00:00:00 [idle\_inject/0]

root 14 2 0 may10 ? 00:00:00 [cpuhp/0]

root 15 2 0 may10 ? 00:00:00 [cpuhp/1]

…

**3.10.3. Usando pgrep, liste todos los PIDs (Process Ids) de procesos que tengan “bash” en su comando de ejecución, redirija la salida a un archivo de texto, y repita esto último 2 veces más (agregando al final del archivo). Luego, liste el contenido del archivo de manera ordenada, eliminando las líneas repetidas y almacene dicho listado en un archivo (todo esto en un mismo comando!).**

Para lograr ese resultado necesitamos:

* ejecutar *pgrep bash* 3 veces y guardar eso en un archivo  
  for i in {1..3}; do pgrep bash >> archivooriginal
* recuperar el contenido de ese archivo  
  cat archivooriginal
* ordenarlo  
  sort
* tomar sólo las líneas únicas y guardar ese resultado en un archivo nuevo  
  uniq > archivonuevo

Juntando todo con pipelines:

for i in {1..3}; do pgrep bash >> archivooriginal | cat archivooriginal | sort | uniq > archivonuevo

Lo ejecutamos y verificamos después:

sfn@PC:~$ for i in {1..3}; do pgrep bash >> archivooriginal; done | cat archivooriginal | sort | uniq > archivonuevo

sfn@PC:~$ cat archivonuevo

187677

189575

189898

190028

195463

220820

227154

228005

228118

42033

sfn@PC:~$ cat archivooriginal

42033

187677

189575

189898

190028

195463

220820

229415

42033

187677

189575

189898

190028

195463

220820

229415

42033

187677

189575

189898

190028

195463

220820

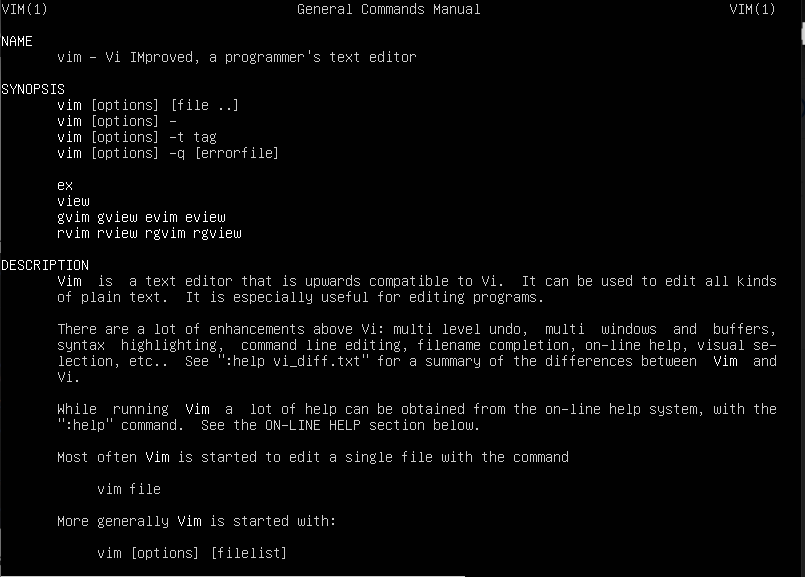
229415

### 

### 3.11. Vim

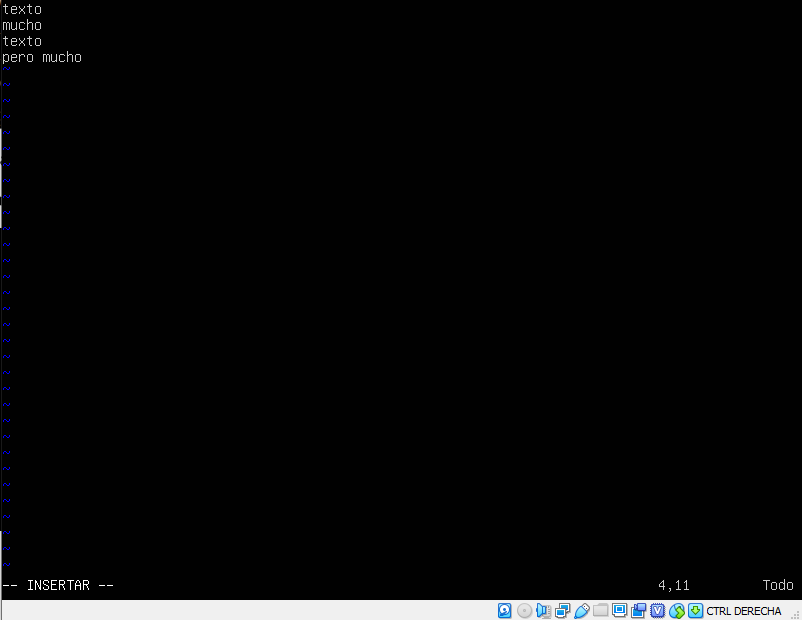
**3.11.1. Vim es uno de los editores de texto que vienen por defecto instalados en todo sistema Linux.**

**3.11.2. Tipee “man vim” para investigar un poco sus características.**

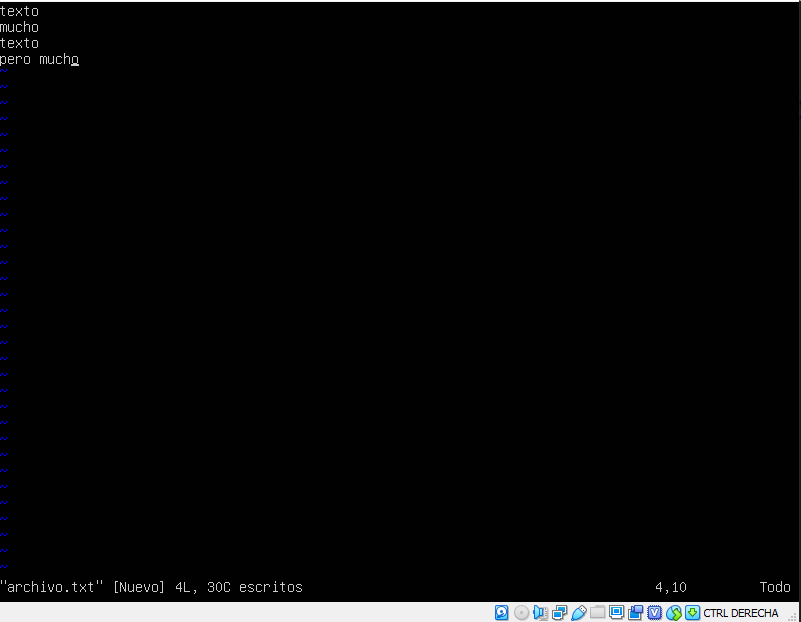
****

**3.11.3. Para crear un archivo y editarlo con el vim, tipee “vim archivo.txt”.**

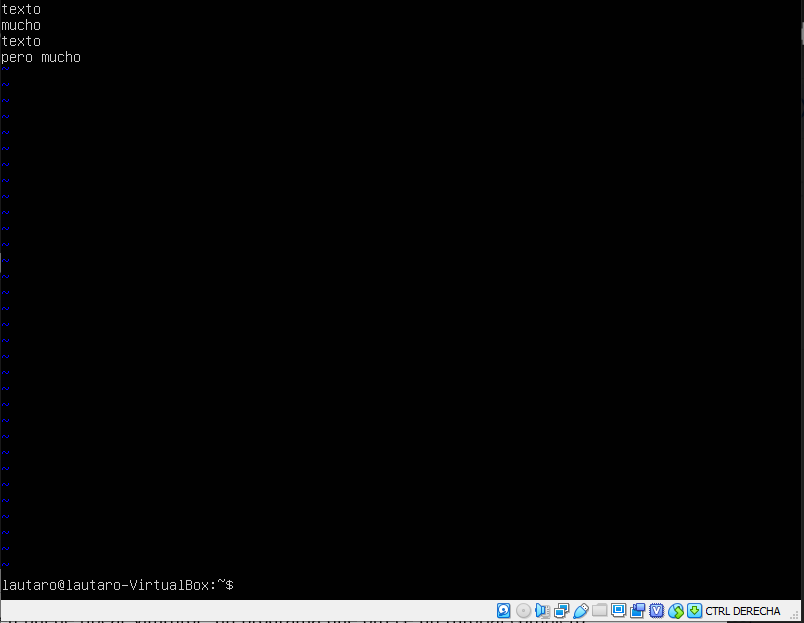
**3.11.4. Para comenzar a escribir debe ingresar al modo edición, presionando la tecla a. Escriba un poco de texto y luego salga del modo de edición presionando <ESC>.**

****

**3.11.5. Para grabar el archivo, presione: w (estando fuera del modo edición).**

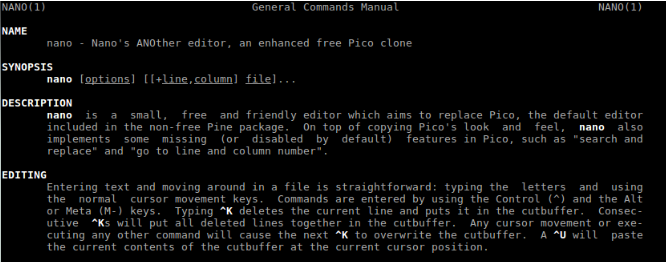
****

**3.11.6. Para salir del editor, presione: q (estando fuera del modo edición).**

****

**3.11.7. Para profundizar sobre el vim (más adelante, cuando sea necesario) puede tipear “vim”. Así entrará a una pantalla desde la cual podrá tipear “:help” y ingresar a la ayuda. También puede tipear vimtutor, un programa que ofrece un tutorial completo del vim.**

**3.11.8. En caso de no sentirse cómodo con el vim, investigue el programa “nano”**

****

## 4. Shell scripting y otras cuestiones

### 4.1.1. Variables de entorno

#### 4.1.1.1. ¿Qué es una variable de entorno? ¿Qué significa exportar una variable?¿Cómo se la exporta?

Las variables de entorno son objetos con nombre que almacenan valores para que puedan ser utilizados por procesos.El valor de una variable de entorno puede ser la ubicación de la carpeta donde están los archivos ejecutables de un sistema o su configuración de idioma, por ejemplo.

Existen dos tipos de variables de entorno:

* Variables locales: visible solo en la terminal donde definió. Pueden imprimirse ejecutando el comando env.
* Variables globales: visible para cualquier proceso en ejecución o que se ejecute desde el shell. Por conversión se escriben en mayúscula. Pueden imprimirse ejecutando el comando printenv.

(El comando set imprime ambos tipos de variables).

Para imprimir el valor de una variable se puede usar el comando echo seguido por $NombreDeVariable:

sfn@PC:~$ echo $LANG

en\_US.UTF-8

La declaración de variables es distinta según el tipo. En el caso de una variable local para declararla basta con escribir el nombre de la variable seguido de = y luego el valor:

sfn@PC:~$ variablePrueba=2

sfn@PC:~$ echo $variablePrueba

2

En el caso de las variables globales es necesario declarar una variable local y **exportarla** usando el comando export (o declare -x o typeset -x):

sfn@PC:~$ export VARIABLE1=1

sfn@PC:~$ declare -x VARIABLE2=2

sfn@PC:~$ typeset -x VARIABLE3=3

sfn@PC:~$ env | grep 'VARIABLE'

VARIABLE3=3

VARIABLE2=2

VARIABLE1=1

Ahora bien, esta generación de variables globales así hace que estén atadas al proceso de la terminal donde se declararon; cuando este termina, desaparecen de la memoria. Para hacerlas persistentes es necesario guardarlas en archivo. Por norma general las variables globales se suelen definir como variables específicas del usuario y se configuran en el archivo *~/.bashrc*.

#### 4.1.1.2. Cree un script llamado environ.sh que exporte una variable y luego imprima su valor en pantalla.

Para crear el archivo del script ejecutamos

nano environ.sh

Y luego en el editor

#!/bin/sh

export VARIABLE\_DEL\_TP='probando variables de entorno'

echo $VARIABLE\_DEL\_TP

#### 4.1.1.3. Después ejecute dicho script en la forma “./environ.sh”. Al finalizar el script verifique si dicha variable se encuentra en el entorno del shell actual. Investigue qué ocurrió.

El archivo no tiene permisos de ejecución. Lo cambiamos con

sfn@PC:~$ chmod +x environ.sh

Y ejecutamos luego:

sfn@PC:~$ ./environ.sh

probando variables de entorno

sfn@PC:~$

Intentamos verificar que la variable exista con el valor fuera del script y corroboramos que no existe:

sfn@PC:~$ echo $VARIABLE\_DEL\_TP

sfn@PC:~$

Investigando, esto parece deberse al alcance (scope) que tienen las variables definidas en un script. “When you call a bash script from your interactive shell, **a new shell is spawned to run the script** (…) Once the shell script exits, **its environment is destroyed**.”[[1]](#footnote-0) Es por eso que el valor de la variable no se persiste.

#### 4.1.1.4. Ejecute el mismo script en la forma “source ./environ.sh”

sfn@PC:~$ source ./environ.sh

probando variables de entorno

sfn@PC:~$ echo $VARIABLE\_DEL\_TP

probando variables de entorno

#### 4.1.1.5. ¿Qué ocurrió ahora? Investigue sobre el comando “source”

El valor de la variable declarada en el script se persistió y pudo accederse a ella desde la terminal.

Investigando descubrimos que el operador source o **.** (punto) “lee y ejecuta los comandos del archivo pasados como argumento en el contexto actual del shell”[[2]](#footnote-1). Es por ese motivo que el valor de la variable está disponible luego.

### 4.1.2. Procesos en primer y segundo plano

#### 4.1.2.1. ¿Cómo se ejecutan comandos en segundo plano (background) en Linux?

Para enviar un proceso directamente a segundo plano, agregamos al final del comando un ampersand. Ejemplo

~$ proceso.sh &

para enviarlo a segundo plano mientras está en ejecución en el shell:

ctrl+z (ctr+c lo corta por completo)

si lo interrumpimos y queremos seguir ejecutándolo:

bg %<nrodeproceso> (no van las <>, son indicativas nomás. bg = background)

#### 4.1.2.2. ¿Cómo hace para conocer qué procesos corren en segundo plano?

Ingresar el comando jobs en la shell.

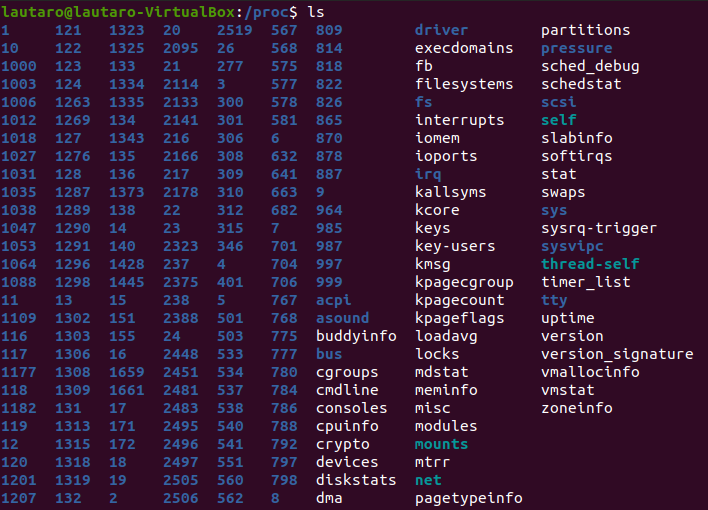
#### 4.1.2.3. ¿Cómo trae un proceso a primer plano nuevamente?

fg %<nrodeproceso>

### 4.1.3. /proc

**4.1.3.1. ¿Qué información contiene este directorio? Investigue exhaustivamente su contenido.**

El sistema de archivos /*proc* contiene un sistema de archivos imaginario o virtual. Este no existe físicamente en disco, sino que el núcleo lo crea en memoria. Se utiliza para ofrecer información relacionada con el sistema (originalmente acerca de procesos, de aquí su nombre). El sistema de archivos /proc se encuentra descrito con más detalle en la página de manual de proc.

****

/proc/1

Un directorio con información acerca del proceso número 1. Cada proceso tiene un directorio debajo de /proc cuyo nombre es el número de identificación del proceso(PID).

/proc/cpuinfo

Información acerca del procesador: su tipo, marca, modelo, rendimiento, etc.

/proc/devices

Lista de controladores de dispositivos configurados dentro del núcleo que está en ejecución

/proc/dma

Muestra los canales DMA que están siendo utilizados.

/proc/interrupts

Muestra la interrupciones que están siendo utilizadas, y cuantas de cada tipo ha habido.

/proc/filesystems

Lista los sistemas de archivos que están soportados por el kernel.

/proc/ioports

Información de los puertos de E/S que se estén utilizando en cada momento.

/proc/kcore

Es una imagen de la memoria física del sistema. Este archivo tiene exactamente el mismo tamaño que la memoria física, pero no existe en memoria como el resto de los archivos bajo /proc, sino que se genera en el momento en que un programa accede a este.

/proc/kmsg

Salida de los mensajes emitidos por el kernel. Estos también son redirigidos hacia syslog.

/proc/ksyms

Tabla de símbolos para el kernel.

/proc/loadavg

El nivel medio de carga del sistema; tres indicadores significativos sobre la carga de trabajo del sistema en cada momento.

/proc/meminfo

Información acerca de la utilización de la memoria física y del archivo de intercambio.

/proc/modules

Indica los módulos del núcleo que han sido cargados hasta el momento.

/proc/net

Información acerca del estado de los protocolos de red.

/proc/self

Un enlace simbólico al directorio de proceso del programa que esté observando a /proc. Cuando dos procesos observan a /proc, obtienen diferentes enlaces. Esto es principalmente una conveniencia para que sea fácil para los programas acceder a su directorio de procesos.

/proc/stat

Varias estadísticas acerca del sistema, tales como el número de fallos de página que han tenido lugar desde el arranque del sistema.

/proc/uptime

Indica el tiempo en segundos que el sistema lleva funcionando.

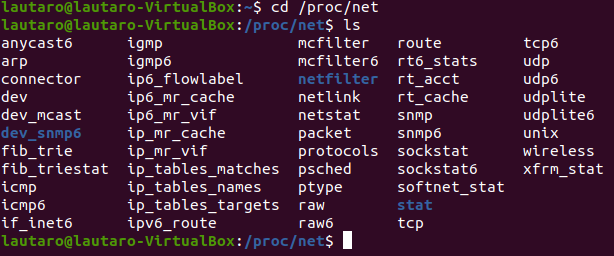
/proc/version

Indica la versión del núcleo

**4.1.3.2. ¿Qué representan los directorios numéricos?**

Los subdirectorios de /proc que tienen nombres numéricos se usan para proporcionar acceso a la información sobre los programas en ejecución.

**4.1.3.3. /proc/net: ¿Qué información contiene este directorio? A medida que envía datos a través de la red (por ejemplo al navegar con firefox), investigue qué archivos son modificados y qué representan esos valores cambiantes.**



arp — Lista la tabla del kernel ARP. Este archivo es particularmente útil para conectar la dirección del hardware a una dirección IP de un sistema.

Directorio atm/ — Los archivos dentro de este directorio contienen las configuraciones de Asynchronous Transfer Mode (ATM) y estadísticas. Este directorio es principalmente usado con redes ATM y tarjetas ADSL.

dev — Lista los diferentes dispositivos de red configurados en el sistema, complementado con estadísticas de transmisión y recepción. Este archivo le indica el número de paquetes que cada interfaz ha enviado y recibido, el número de paquetes entrantes y salientes, número de errores vistos, el número de paquetes abandonados y mucho más.

dev\_mcast — Lista los grupos multicast Layer2 en los que cada dispositivo esté escuchando.

igmp — Lista las direcciones IP con destinatarios múltiples (multicast) a las que el sistema se ha incorporado.

ip\_conntrack — Lista las conexiones de red para las máquinas que están reenviando conexiones IP.

ip\_tables\_names — Lista los tipos de iptables en uso. Este archivo sólo está presente si iptables esta activo en el sistema y contiene uno o más de los siguientes valores: filter, mangle o nat.

ip\_mr\_cache — Lista de la caché de routing de múltiple destinatario.

ip\_mr\_vif — Lista las interfaces virtuales de múltiple destinatario (multicast).

netstat — Contiene una amplia colección de estadísticas de red, incluyendo la temporización TCP, los cookies enviados y recibidos y mucho más.

psched — Lista de parámetros de planificación global del paquete.

raw — Lista las estadísticas de dispositivo brutos (raw).

route — Lista la tabla de enrutamiento del kernel.

rt\_cache — Contiene la caché de ruta actual.

snmp — Lista de los datos del protocolo Simple Network Management Protocol (SNMP) para varios protocolos de red en uso.

sockstat — Proporciona estadísticas de socket.

tcp — Contiene información detallada del socket TCP.

tr\_rif — Lista la tabla de enrutamiento de token ring RIF.

udp — Contiene información detallada del socket UDP.

unix — Lista sockets de dominio UNIX.

wireless — Lista datos de la interfaz de radio.

**4.1.4. ¿Qué información contiene el archivo /etc/passwd?**

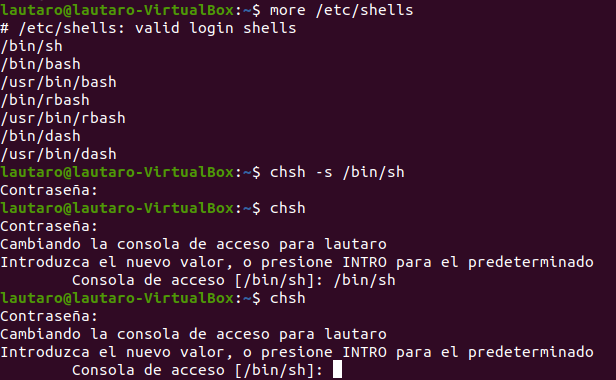
El contenido del fichero /etc/passwd determina quién puede acceder al sistema de manera legítima y que se puede hacer una vez dentro del sistema. Este fichero es la primera línea de defensa del sistema contra accesos no deseados

**4.1.5. ¿Para qué se utiliza el comando chsh?**

Su propósito es cambiar la shell de inicio de sesión. Al utilizarla, si no se proporciona una shell en la línea de comandos, chsh solicita una.

more /etc/shells

Para cambiar el shell por defecto desde la consola hacemos uso del comando chsh, éste comando tiene dos formas de ejecutarse, una es mediante parámetros y la otra de manera interactiva con el usuario, para hacer uso del mismo mediante parámetros hacemos uso del modificador -s seguido del shell que queremos utilizar.



**4.1.6. SSH: Herramienta de logueo y ejecución de comandos remoto. Utiliza diversos algoritmos de encripción para proveer seguridad a la comunicación.**

**4.1.6.1. Investigue su uso y como automatizar el logueo de un usuario en una máquina remota (es decir, sin requerir el ingreso de password), leer man ssh-keygen y analizar el uso del archivo authorized\_keys.**

SSH o Secure Shell, es un protocolo de administración remota que le permite a los usuarios controlar y modificar sus servidores remotos a través de Internet a través de un mecanismo de autenticación.

Otras características fundamentales de SSH son que nos va a permitir copiar datos de manera segura, tanto archivos como carpetas, a través del protocolo SFTP (SSH FTP), un protocolo hecho desde cero y que no tiene nada que ver con FTPS o FTPES (FTP sobre SSL/TLS).

*El protocolo SSH proporciona confidencialidad (los datos van cifrados punto a punto), autenticación (podremos autenticarnos frente al servidor SSH de múltiples maneras, con usuario/clave, criptografía de clave pública e incluso podremos configurar un segundo factor de autenticación), integridad (si los datos se modifican o los modifica un usuario malintencionado se podrá detectar, ya que usa HMAC para comprobar la integridad de todos y cada uno de los datos).*

*ssh-keygen genera, administra y convierte claves de autenticación para ssh y puede crear claves para el uso de las versiones de protocolo SSH 1 y 2.*

*Ssh-keygen también se puede utilizar para generar y actualizar Key Revocation Lists, y para probar si las claves dadas han sido revocadas por uno.*

*Con ssh-keygen se crean dos archivos de autenticación, uno con una clave pública (Public Key) y otra privada (Private Key), luego se debe enviar el archivo de la Public Key al servidor al que se desea conectar (indicando en donde se debe almacenar) y luego sólo se pide login una única vez mediante ssh. Luego, se debe crear en el servidor destino un archivo authorized\_keys que contendrá todas las Public Keys que acepta el mismo para logueo automático mediante SSH.*

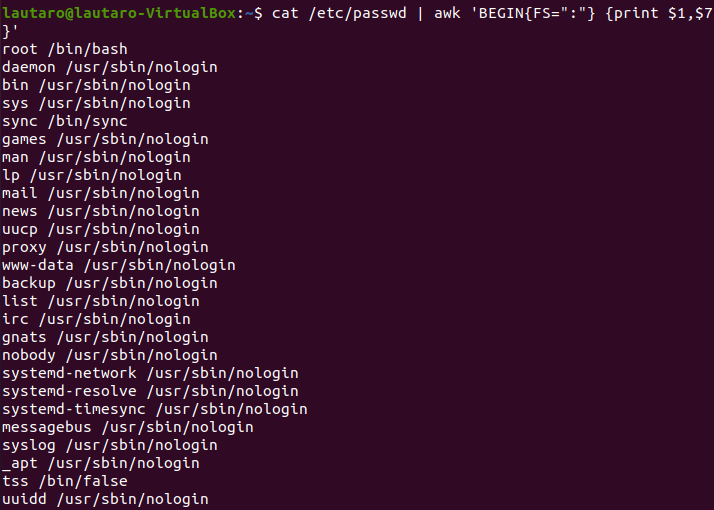
**4.1.7. Procesamiento de textos (awk): awk es un lenguaje de programación destinado al procesamiento de textos.**

**4.1.8. ¿Qué realizan las siguientes operaciones?:**

**4.1.8.1. ls -l | awk '{print $6}'**

Lista el mes de modificación de los archivos/directorios contenidos en el directorio donde se lo ejecuta.

**4.1.8.2. cat /etc/passwd | awk 'BEGIN {FS=”:”} {print $1,$7}'**

****

**4.1.9. Algunas lecturas de man recomendadas:**

**4.1.9.1. bash, chsh, ssh, ssh-keygen, netstat, lsof, proc, stat, ps, grep, make, chmod, cut, awk, pidof, renice, nice, ifconfig, tar, gzip, kill, cat, echo, install, bc, tty, who, w, wc.**

**4.1.10. Algunos comandos built-in a investigar:**

**4.1.10.1. read, “.” (punto) o su alias source, fg, bg, jobs, export, unset, kill**

read: Lee una línea de la entrada estándar y asigna las palabras de la línea a las variables indicadas, si el número de palabras es mayor que el número de variables se asigna a la última variable las palabras sobrantes. Si es a la inversa se restaura el valor de las variables no asignadas.

read ­-e: inserta toda la línea en la primer variable.

read -­p: frase muestra frase si se lee de un terminal.

read ­-p: "Escriba algo: " palabra1 palabra2 palabra3

echo $palabra1 $palabra2 $palabra3

“.” (punto) o su alias source

Utilizado para ejecutar un script . Podemos pasarle a source el nombre del script o utilizar el caracter ‘ . ‘ que tiene la misma función. De este modo el script se ejecuta en la consola actual y no se inicia una instancia nueva de la consola.

fg

Reanuda trabajos suspendidos poniéndolos en foreground, o trabajos en background los pasa a primer plano.

bg

El comando bg se utiliza para mover un trabajo a segundo plano.

jobs

Lista los trabajos que se están ejecutando actualmente, ya sea los que están suspendidos

(Control­Z) como los que están corriendo en segundo plano (background). En la salida de esta orden, el último trabajo se marca con un más + y el penúltimo con un menos – el resto de los trabajos no llevan marcas.

export

Exporta el valor de una variable para que pueda ser accesible en los subshell.

unset

Pone a cero el valor de las variables, si se consulta por ellas luego de setearlas a cero se

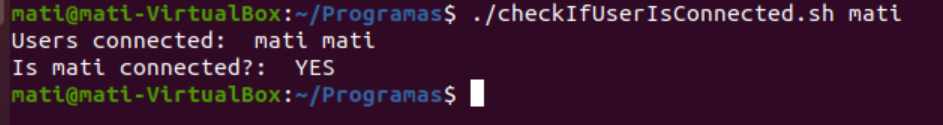
mostrará una cadena nula (una línea en blanco). Por lo tanto una alternativa es asignar explícitamente el valor nulo, dándole un valor vacío (variable=).

## 5. Ejercicios de Scripting

### 5.1. Realice un script llamado “userconnected” que retorne un YES si el primer parámetro coincide con algún usuario conectado o NO en caso contrario.

(Adjuntamos el archivo *userconnected.sh*)

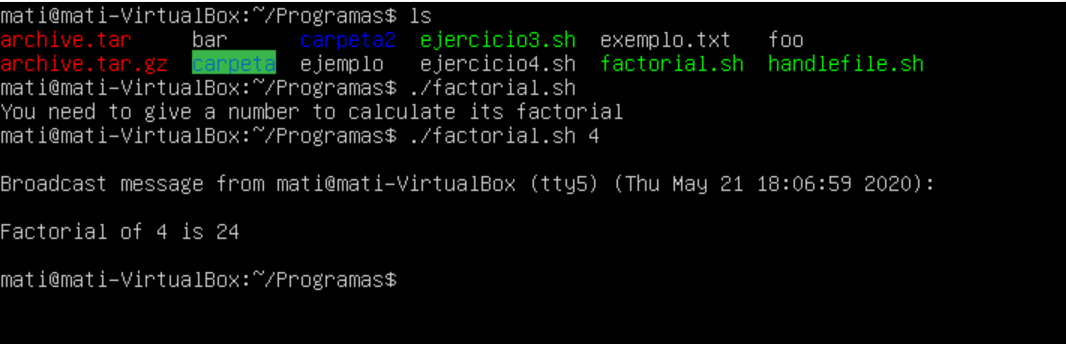
único usuario conectado mati:



### 5.2 Escriba un script que muestre el factorial de un número pasado como parámetro e indique el resultado por pantalla a los usuarios conectados.

Adjuntamos el archivo *printfactorial.sh*

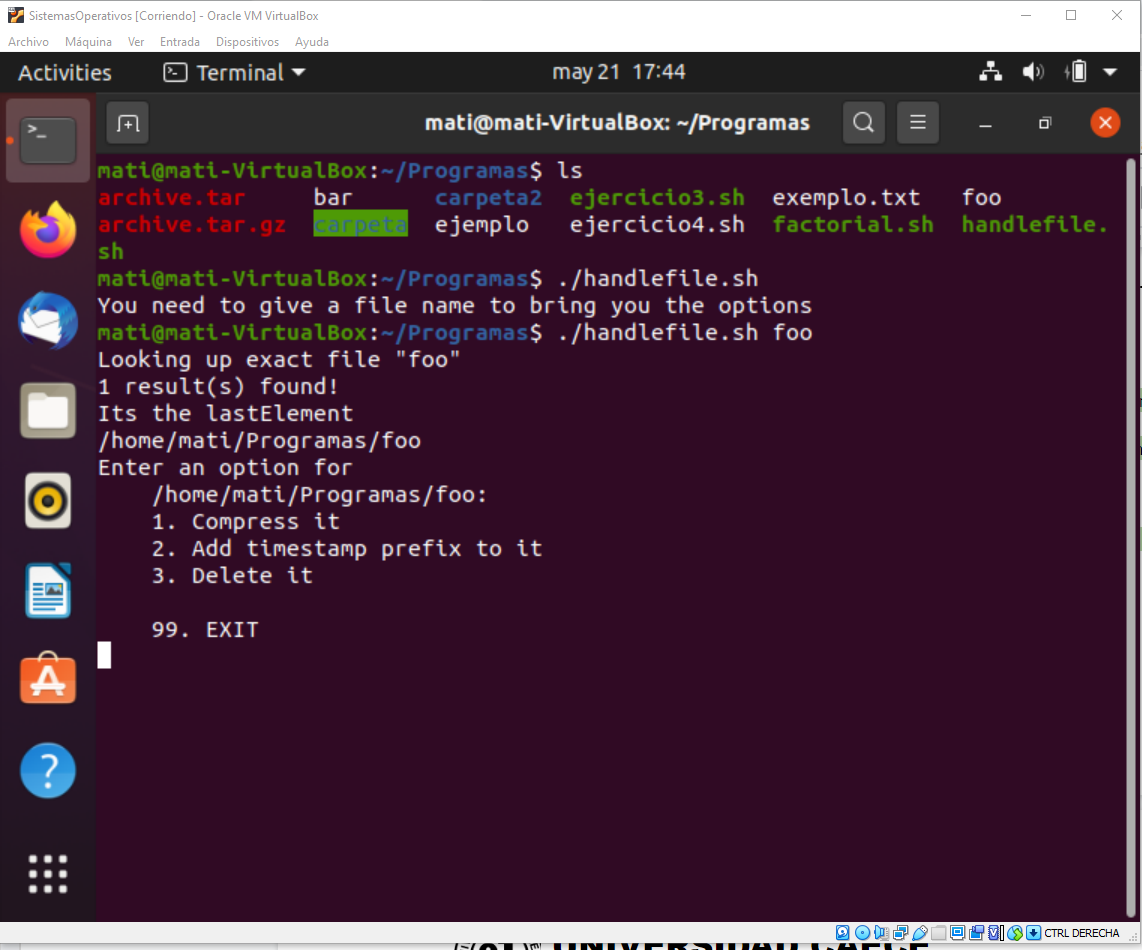
Captura de la ejecución:



### **5.3 Escriba un script que pregunte por un archivo de entrada, y que tenga las opciones dentro** de ese mismo script de comprimirlo, de asignarle una extensión con un timestamp correspondiente o bien de eliminarlo si está dentro del home del usuario logueado.

Adjuntamos el archivo *handleflie.sh*

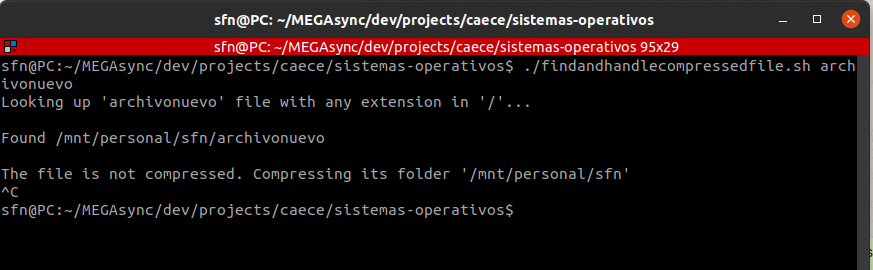
Captura de la ejecución:

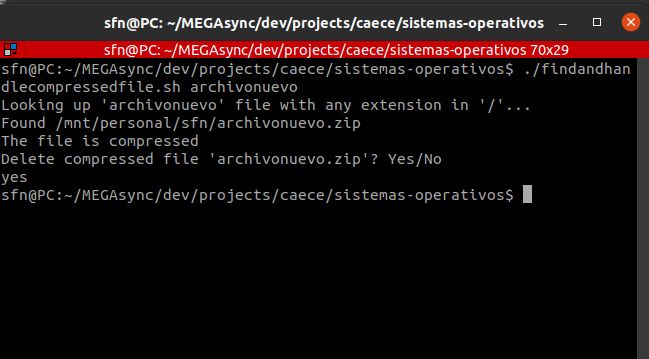


### 5.4 Realizar además un script que, si se ingresa por parámetro un archivo comprimido, lo busca por todo el disco, muestra la ruta e informa si efectivamente se lo quiere borrar, pero si no existe deberá comprimir la carpeta donde este el archivo en cuestión.

Adjuntamos el archivo *findandhandlecompressedfile.sh*

Capturas de su ejecución:

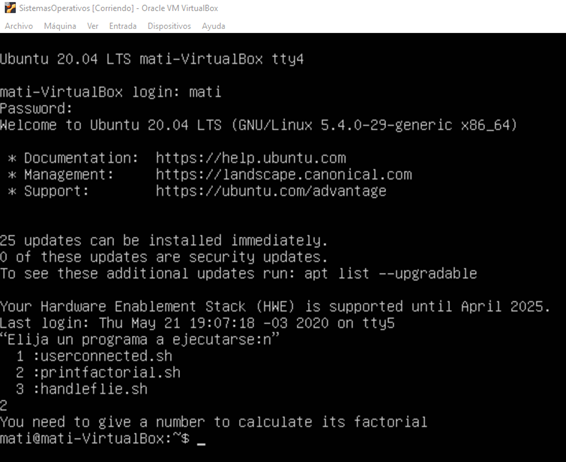




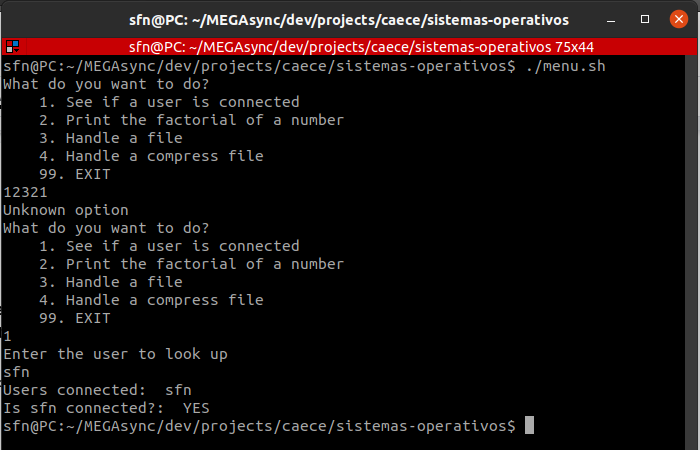
### 5.5 Genere un menú donde aparezca todas estas 4 opciones al acceder inmediatamente al login del usuario.

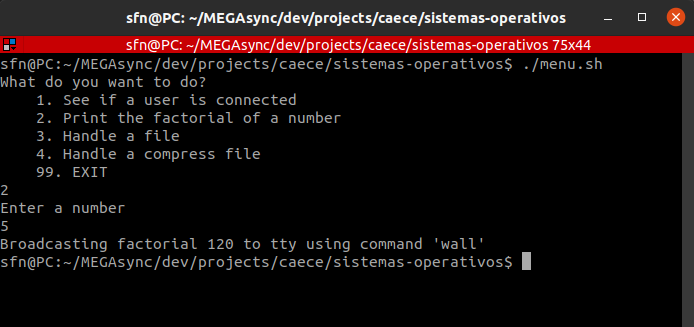
(Adjuntamos el archivo *menu.sh*)

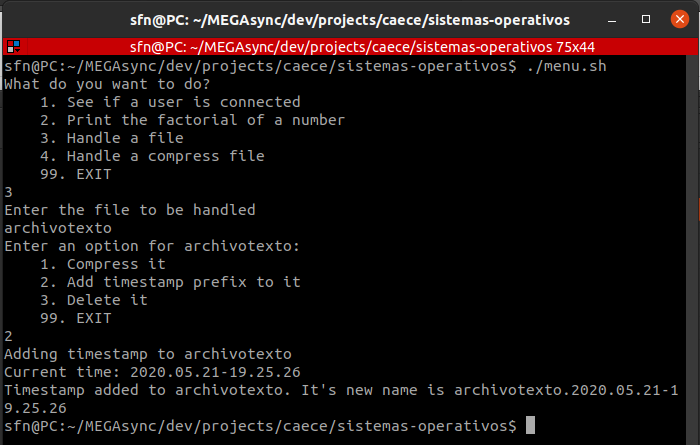
Para ejecutarse al loguearse, **el script debe ser agregado a la carpetra /etc/profile.d** (es necesario hacerlo como sudo). De esa manera se ejecutará cada vez que un usuario se loguee. Ej:

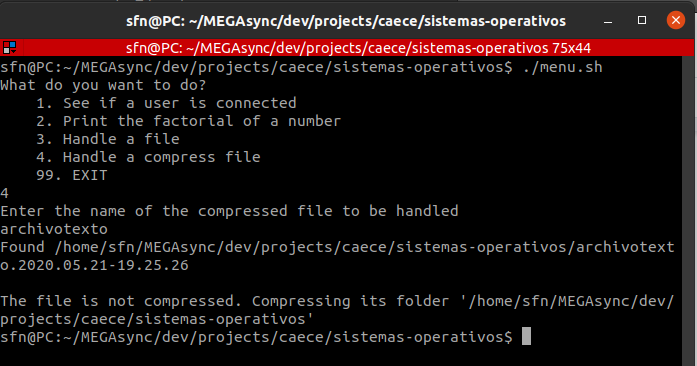


Capturas de la ejecución de *menu.sh*:









1. <https://www.shellscript.sh/variables1.html> [↑](#footnote-ref-0)
2. <https://ss64.com/bash/source.html> [↑](#footnote-ref-1)