

PR_01.1

Nota: Las capturas pueden variar en color, resolución y aspecto debido al uso de dos equipos diferentes durante la práctica.

1. Conceptos Básicos y Variables
2. Obtener Ayuda y Localizar Archivos
3. Navegación y Listado de archivos
4. Manipulación de Archivos y Directorios
5. Archivo y Compresión
6. Redirección, Tuberías y Filtros
7. Scripts Básicos

1. Conceptos Básicos y Variables

1. Muestra el contenido de tu variable de entorno `HOME` . Luego, usa `cd` junto con esa variable para navegar a dicho directorio y verifica con `pwd` que te encuentras en la ubicación correcta

```
akoarguel at servidor in ~ in
λ echo $HOME
/home/akoarguel
akoarguel at servidor in ~ in
λ cd $HOME
akoarguel at servidor in ~ in
λ pwd
/home/akoarguel
akoarguel at servidor in ~ in
λ
```

Utilizamos `echo` para mostrar la información por pantalla, usamos esa misma variable para ir hasta la ubicación con `cd` y comprobamos si estamos en la ubicación deseada con `pwd` .

2. Ejecuta el comando `whoami` . Ahora, crea una variable local llamada `USUARIO_ACTUAL` que contenga el resultado del comando anterior y muéstrala en el terminal.

```

akoarguel at servidor in ~ in
λ whoami
akoarguel
akoarguel at servidor in ~ in
λ USUARIO_ACTUAL=$(whoami)
akoarguel at servidor in ~ in
λ echo $USUARIO_ACTUAL
akoarguel
akoarguel at servidor in ~ in
λ _

```

El comando `whoami` muestra el usuario. De esta forma crearemos la variable `$USUARIO_ACTUAL` y le daremos este valor para así que, cuando lo mostremos con `echo` nos muestre el nombre del usuario.

3. Intenta crear un archivo llamado `dos palabras.txt` sin usar comillas. Observa el resultado con `ls`. ¿Qué ha ocurrido y por qué? Ahora, bórralo(s) y créalo correctamente.

```

akoarguel at servidor in ~/PR_01.1 in
λ touch dos palabras.txt
akoarguel at servidor in ~/PR_01.1 in
λ tree
.
├── dos
└── palabras.txt

1 directory, 2 files
akoarguel at servidor in ~/PR_01.1 in
λ _

```

Al intentar crear un documento llamado *dos palabras.txt* sin usar las comillas, nos encontramos con que ha creado dos archivos diferentes.

Para solucionarlo primero borramos con el comando `rm` los dos archivos y lo creamos correctamente con el uso de comillas.

```

└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ rm dos
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ rm palabras.txt
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ touch "dos palabras.txt"
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ tree

└─┬─ dos palabras.txt

1 directory, 1 file
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ

```

4. Usa el comando `type` para averiguar si `ls` y `cd` son internos o externos al shell. ¿Qué diferencia prácticas crees que implica esto?

```

└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ type -a ls
ls is aliased to `ls --color=auto'
ls is /usr/bin/ls
ls is /bin/ls
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ type -a cd
cd is aliased to `_omb_directories_cd'
cd is a shell builtin
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ _

```

- `ls` es **externo**, un programa situado en la carpeta ./bin/ls
- `cd` es interno, forma parte del propio shell

Esto quiere decir que los comandos internos a demás de ser más rápidos pueden modificar directamente el entorno SHELL mientras que, los comandos externos son programas independientes que se ejecutan independientemente.

5. Muestra tu `PATH` actual. Crea un directorio `~/mi_bin` y añádelo temporalmente al principio de tu `PATH`. Verifica que el cambio se ha realizado correctamente.

```
akoarguel at servidor in ~ in
λ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s
akoarguel at servidor in ~ in
λ mkdir -p ~/mi_bin
mkdir: created directory '/home/akoarguel/mi_bin'
akoarguel at servidor in ~ in
λ export PATH=~/mi_bin:$PATH
akoarguel at servidor in ~ in
λ echo $PATH
/home/akoarguel/mi_bin:/usr/local/sbin:/usr/local/bi
games:/snap/bin
akoarguel at servidor in ~ in
λ _
```

La variable `$PATH` nos muestra las carpetas del equipo. Hemos creado un directorio con el nombre de `~/mi_bin` y lo hemos añadido temporalmente (hasta que la maquina/terminal se reinicie) al principio del `PATH` con el comando `export`.

2. Obtener Ayuda y Localizar Archivos

1. Abre la página del manual para el comando `chmod`. ¿En qué sección del manual se encuentra? ¿Qué indica ese número de sección sobre el tipo de comando?

```
CHMOD(1) User Commands CHMOD(1)
NAME
  chmod - change file mode bits
SYNOPSIS
  chmod [OPTION]... MODE[,MODE]... FILE...
  chmod [OPTION]... OCTAL-MODE FILE...
  chmod [OPTION]... --reference=RFILE FILE...
DESCRIPTION
  This manual page documents the GNU version of chmod.  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.

  The format of a symbolic mode is [ugoa...][[+=]perms...], where perms is either zero or more letters from the set rwXst, or a single letter from the set ugo.  Multiple symbolic modes can be given, separated by commas.

  A combination of the letters ugo controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users (a).  If none of these are given, the effect is as if (a) were given, but bits that are set in the umask are not affected.
```

El manual de `chmod` se obtiene con el comando `man`. El manual se encuentra en la *sección 1* que parece indicar que es la parte con comandos de usuario ejecutables.

2. Usando la función de búsqueda dentro de la pagina del manual `ls`, encuentra la opción que ordena los archivos por tamaño.

```
15 servidor in ~
◆ → man ls_
```

```
-r, --reverse
    reverse order while sorting

-R, --recursive
    list subdirectories recursively

-s, --size
    print the allocated size of each file, in blocks

-S      sort by file size, largest first

--sort=WORD
    sort by WORD instead of name: none (-U), size (-S), time (-t),

--time=WORD
    select which timestamp used to display or sort; access time (-a),
    change time (-c): ctime, status; modified time (default): mtime,
    creation;
```

Encontramos dos opciones para organizar por tamaño. `ls -s` organiza por tamaño cada archivo en bloques y `ls -S` organiza los archivos por tamaño, primero los archivos con mayor tamaño.

```
15 servidor in /
◆ → ls -s
total 2097236
 0 bin
 4 bin.usr-is-merged
 4 boot
 4 cdrom
 0 dev
 4 etc
 4 home
 0 lib
 0 lib64
 4 lib.usr-is-merged
16 lost+found
 4 media
 4 mnt
 4 opt
 0 proc
 4 root
 0 run
 0 sbin
 4 sbin.usr-is-merged
 4 snap
 4 srv
 2097156 swap.img
 0 sys
 4 tmp
 4 usr
 4 var

15 servidor in /
◆ → ls -S
swap.img  bin.usr-is-merged  etc  media  root  sbin.usr-is-merged  srv  var  sbin  proc
lost+found  boot  home  mnt  sbin.usr-is-merged  tmp  run  bin  sys
dev  cdrom  lib.usr-is-merged  opt  snap  usr  lib64  lib

15 servidor in /
◆ →
```

3. Imagina que has olvidado dónde se guarda el archivo de configuración de usuario. Sabiendo que se llama `passwd`, usa `find` para buscarlo desde el directorio raíz. Anota la ruta completa que has encontrado.

```
15 servidor in /  
◆ → sudo find / -name passwd  
/etc/pam.d/passwd  
/etc/passwd  
/usr/share/doc/passwd  
/usr/share/lintian/overrides/passwd  
/usr/share/bash-completion/completions/passwd  
/usr/bin/passwd
```

```
15 servidor in /  
◆ → _
```

El comando `find` requiere de usar `sudo` por tema de derechos de administrador. Además le añadimos la etiqueta `-name` para buscar en concreto `passwd`.

4. Crea un archivo vacío llamado `text_locate.txt` en tu directorio home. Inmediatamente después, búscalo con `locate`. ¿Aparece en los resultados?

Para usar `locate` primero debemos instalarlo.

```
sudo apt install plocate
```

Tras realizar el ejercicio varias veces el resultado es correcto. Solo que necesitamos permisos `sudo` para poder crear un archivo en la ruta `/home`.

```
15 servidor in /home  
◆ → sudo touch text_locate.txt  
  
15 servidor in /home  
◆ → locate text_locate.txt  
/home/text_locate.txt
```

5. Basado en el ejercicio anterior, ¿qué comando (probablemente con sudo) necesitas ejecutar para que locate sí encuentre tu archivo? Ejecútalo y verifica que ahora sí lo encuentras.

Como se puede ver en el ejercicio anterior, si es necesario usar `sudo` para crear el archivo pero no para localizarlo.

3. Navegación y Listado de archivos

1. Navega al directorio `/etc` . Desde ahí, sin usar `cd` , lista el contenido de tu directorio `home` usando una ruta con el atajo `~` .

```
15 servidor in /home
◆ → cd /etc/

15 servidor in /etc
◆ → ls /home
akoarguel text_locate.txt

15 servidor in /etc
◆ → _
```

2. Desde tu directorio `home` , navega a `/` y luego a `var` y finalmente a `log` usando una sola línea de comando y rutas relativas

```
15 servidor in /home
◆ → pwd
/home

15 servidor in /home
◆ → cd ../../var/log

15 servidor in /var/log
◆ → pwd
/var/log
```

3. Lista el contenido de `/etc` en formato largo. En la salida, identifica el propietario, el grupo y los permisos del archivo `passwd`.

Para ello nos apoyaremos del comando para consultar el manual de `ls`.

```
man ls
```

```
-I, --ignore=PATTERN
    do not list implied entries matching shell PATTERN

-k, --kibibytes
    default to 1024-byte blocks for file system usage;

-l
    use a long listing format

-L, --dereference
    when showing file information for a symbolic link,
    follow the link rather than for the link itself

-m
    fill width with a comma separated list of entries
```

Al usar solo `ls -l /etc` muestra una lista muy grande de archivos, para simplificarlo vamos a hacer uso de la tubería `|` y del filtro `grep passwd`. (Esta información la saque de *stackoverflow*).

```
16 servidor in /etc
♦ → ls -l /etc | grep passwd
-rw-r--r-- 1 root root      1746 oct 16 15:10 passwd
-rw-r--r-- 1 root root      1688 oct 16 15:08 passwd-

16 servidor in /etc
♦ →
```

4. Compara la salida de `ls -l /etc` y `ls -lh /etc`. ¿Qué hace la opción `h` y por qué es útil para personas?

```
ls -l /etc
```



```
-rw-r--r-- 1 root root      681 abr  8  2024 xattr.conf
drwxr-xr-x 4 root root    4096 ago  5 17:02 xdg
drwxr-xr-x 2 root root    4096 ago  5 17:02 xml
-rw-r--r-- 1 root root      460 ago  5 17:14 zsh_command_not_found
```

```
ls -lh /etc
```

```
-rw-r--r-- 1 root root      681 abr  8  2024 xattr.conf
drwxr-xr-x 4 root root    4,0K ago  5 17:02 xdg
drwxr-xr-x 2 root root    4,0K ago  5 17:02 xml
-rw-r--r-- 1 root root      460 ago  5 17:14 zsh_command_not_found
```

La única diferencia es la columna de los tamaños de cada archivo.

- `ls -l /etc` → muestra la información del tamaño en bytes. (2048, 4096, etc.).
- `ls -lh /etc` → añade una opción (`-h`) que significa *human-readable*, que muestra los tamaños de forma legible para humanos, como:

```
2.0K 1.3M 512K
```

5. Ejecuta `ls -R ~`. ¿Qué hace la opción `R`? ¿Por qué podría ser peligroso usarla en el directorio raíz (`/`)?

```
16 servidor in /
♦ → ls -R ~
/home/akoarguel:
carpeta01 carpeta02 carpeta03

/home/akoarguel/carpeta01:
fichero01.txt fichero02.txt

/home/akoarguel/carpeta02:
fichero01.txt fichero02.txt fichero03.txt fichero04.txt fichero05.txt

/home/akoarguel/carpeta03:

16 servidor in /
♦ → _
```

Lo peligroso del comando `R` es que muestra el directorio indicando todos sus subdirectorios, uno dentro de otro.

!! Es peligroso ejecutar `ls -R /` porque el sistema intentará listar recursivamente todo el contenido del sistema, incluyendo archivos visuales del kernel. Esto puede provocar una ralentización en todo el sistema o en la terminal.

4. Manipulación de Archivos y Directorios

1. Crea la estructura de directorios `proyecto/src` , `proyecto/doc` y `proyecto/bin` usando un único comando `mkdir` .

```
akoarguel at servidor in ~ in
λ mkdir proyecto/src proyecto/doc proyecto/bin
mkdir: created directory 'proyecto'
mkdir: created directory 'proyecto/src'
mkdir: created directory 'proyecto/doc'
mkdir: created directory 'proyecto/bin'
akoarguel at servidor in ~ in
λ _
```

2. Crea un archivo `~/notas.txt` . Muévelo a `~/proyecto/doc` y, en el mismo comando, renómbralo a `README.md`

```
akoarguel at servidor in ~ in
λ touch ~/notas.txt
akoarguel at servidor in ~ in
λ mv ~/notas.txt ~/proyecto/doc/README.md
renamed '/home/akoarguel/notas.txt' -> '/home/akoarguel/proyecto/doc/README.md'
akoarguel at servidor in ~ in
λ _
```

`touch` para crear el archivo y usamos `mv` para mover el archivo hasta el directorio deseado y, en la misma ruta renombramos el archivo a `"README.md"`.



3. Copia el archivo `README.md` de `proyecto/doc` a `proyecto/bin`. Luego, borra el archivo original de la carpeta `doc`.

```
akoarguel at servidor in ~/proyecto in
λ cp doc/README.md ~/proyecto/bin/README.md
'doc/README.md' -> '/home/akoarguel/proyecto/bin/README.md'
akoarguel at servidor in ~/proyecto in
λ tree
.
├── bin
│   └── README.md
├── doc
│   └── README.md
└── src

4 directories, 2 files
akoarguel at servidor in ~/proyecto in
λ _
```

```
akoarguel at servidor in ~/proyecto in
λ rm ~/proyecto/doc/README.md
akoarguel at servidor in ~/proyecto in
λ tree
.
├── bin
│   └── README.md
├── doc
└── src
```

4. Intente borrar el directorio `proyecto` con `rmdir`. ¿Qué error obtienes? Ahora, usa `rm` con la opción correcta para borrar el directorio y todo lo que contiene.

```

└─◆ akoarguel at ◆ servidor in ◆ ~ in
└─λ rmdir proyecto/
rmdir: failed to remove 'proyecto/': Directory not empty
└─◆ akoarguel at ◆ servidor in ◆ ~ in
└─λ

```

Resulta que si intentamos borrar un directorio con contenido (carpetas, ficheros, etc.) no podemos utilizar rmdir ya que nos saldrá el siguiente error

rmdir: failed to remove 'proyecto/': Directory not empty

Para borrar la carpeta exitosamente debemos usar `rm -r`.

```

└─◆ akoarguel at ◆ servidor in ◆ ~ in
└─λ rm -r proyecto/
└─◆ akoarguel at ◆ servidor in ◆ ~ in
└─λ tree
.
├── mi_bin
├── PR_01.1
│   ├── carpeta01
│   └── carpeta02

```

Es recomendable usar antes 'sudo'

5. Navega a /etc. Usando un solo comando ls con globbing, lista todos los archivos que empiecen por la letra `s` y terminen con `.conf`.

```

└─◆ akoarguel at ◆ servidor in ◆ /etc in
└─λ ls s*.conf
sensors3.conf  sudo.conf  sudo_logsrvd.conf  sysctl.conf
└─◆ akoarguel at ◆ servidor in ◆ /etc in
└─λ _

```

5. Archivo y Compresión

1. Crea un directorio `tar` llamado `log_backup.tar` que contenga todos los archivos del directorio `/var/log`. ¿Qué advertencias de "permiso denegado" aparecen y por qué?

```

└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ cp -r /var/log /home/akoarguel/PR_01.1/log_backup.tar

```

```

└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ ls
carpeta01 carpeta02 log_backup.tar
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ cd log_backup.tar/
/home/akoarguel/PR_01.1/log_backup.tar
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1/log_backup.tar in
└─λ ls
alternatives.log cloud-init.log dmesg.1.gz faillog lastlog
appport.log cloud-init-output.log dmesg.2.gz installer private
apt dist-upgrade dmesg.3.gz journal README
auth.log dmesg dmesg.4.gz kern.log syslog
bootstrap.log dmesg.0 dpkg.log landscape sysstat
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1/log_backup.tar in
└─λ

```

al realizar una copia de esta carpeta nos salen una serie de advertencias como:

```
cp: cannot open './cloud-init-output.log' for reading: Permission denied
```

Parece que se debe a que estos archivos no pueden estar duplicados en el equipo.

2. **Comprime el archivo `log_backup.tar` con `gzip`. Compara el tamaño del archivo original y el comprimido usando `ls -lh`.**

```

16 servidor in ~/PR_01.1
◆ → tree
.
└─ log_backup.tar

1 directory, 1 file

16 servidor in ~/PR_01.1
◆ → ls -lh log_backup*
-rw-r--r-- 1 root root 99M oct 20 16:15 log_backup.tar

16 servidor in ~/PR_01.1
◆ → gzip log_backup.tar

16 servidor in ~/PR_01.1
◆ → ls -lh log_backup.tar.gz
-rw-r--r-- 1 akoarguel akoarguel 1,4M oct 20 16:15 log_backup.tar.gz

16 servidor in ~/PR_01.1
◆ → _

```

primero hemos mirado cuanto ocupa el archivo original (99 Megabytes).
Tras comprimir el archivo con gzip pesa 1,4 Megabytes.

3. Lista el contenido del archivo `log_backup.tar.gz` sin extraerlo para verificar que los archivos están dentro.

Tras investigar, el mejor comando para realizar esto es el siguiente:

```
tar -tzf log_backup.tar.gz
```

Esto es una parte del contenido como ejemplo:

```
var/log/installer/subiquity-server-info.log.1427
var/log/installer/curtin-install/
var/log/installer/curtin-install/subiquity-partitioning.conf
var/log/installer/curtin-install/subiquity-initial.conf
var/log/installer/curtin-install/subiquity-extract.conf
var/log/installer/curtin-install/subiquity-curtin-apt.conf
var/log/installer/curtin-install/subiquity-curthooks.conf
var/log/installer/subiquity-server-debug.log.1427
var/log/installer/curtin-install.log
var/log/installer/device-map.json
var/log/installer/cloud-init.log
var/log/installer/subiquity-client-info.log
var/log/installer/subiquity-client-info.log.1388
var/log/installer/subiquity-client-debug.log
var/log/installer/casper-md5check.json
var/log/installer/installer-journal.txt
var/log/dmesg.1.gz
var/log/dmesg
var/log/btmp
```

4. Extrae únicamente el archivo `syslog` (o `messages`) de `log_backup.tar.gz` a tu directorio `/tmp`

```
16 servidor in ~/PR_01.1
◆ → sudo tar -xzf log_backup.tar.gz -C /tmp var/log/syslog
[sudo] password for akoarguel:

16 servidor in ~/PR_01.1
◆ → ls -l /tmp/var/log/
total 480
-rw-r----- 1 syslog adm 489071 oct 20 16:15 syslog

16 servidor in ~/PR_01.1
◆ →
```

Utilizaremos el siguiente comando para extraer el archivo que nos interesa (syslog).

```
sudo tar -xzf log_backup.tar.gz -C /tmp var/log/syslog
```

`-x` → extrae archivos del tar

`-z` → indica que está comprimido con gzip

`-f` (nombre del archivo) → archivo a extraer

`-C` → destino de extracción

5. **Crea tres archivos (a.txt, b.log, c.jpg) y luego crea un archivo zip que los contenga.**

```
16 servidor in ~/PR_01.1/ej5
◆ → touch a.txt b.log c.jpg

16 servidor in ~/PR_01.1/ej5
◆ → ls
a.txt  b.log  c.jpg
```

Deberemos instalar zip para poder continuar con la tarea.

```
sudo apt install zip
```

```
16 servidor in ~/PR_01.1
◆ → zip -r ej5.zip ej5
adding: ej5/ (stored 0%)
adding: ej5/b.log (stored 0%)
adding: ej5/c.jpg (stored 0%)
adding: ej5/a.txt (stored 0%)
```

6. **Elimina los tres archivos originales y luego recupéralos desde el archivo**

`.zip` .

```
16 servidor in ~/PR_01.1
♦ → rm -r ej5
```

```
16 servidor in ~/PR_01.1
♦ →
```

```
16 servidor in ~/PR_01.1
♦ → unzip ej5.zip
Archive: ej5.zip
  creating: ej5/
  extracting: ej5/b.log
  extracting: ej5/c.jpg
  extracting: ej5/a.txt
```

```
16 servidor in ~/PR_01.1
♦ → tree
```

```

.
├── ej5
│   ├── a.txt
│   ├── b.log
│   └── c.jpg
├── ej5.zip
└── log_backup.tar.gz
```

7. Usa `zcat` (o `gzcat`) para leer el contenido de un archivo de log comprimido sin crear un archivo descomprimido.

Nos dirigimos a la carpeta de `/var/log` y buscamos un archivo `.gz`.

```
16 servidor in /var/log
♦ → ls
alternatives.log  bootstrap.log      dist-upgrade      dmesg.2.gz      installer
appport.log       bttmp             dmesg             dmesg.3.gz      journal
apt              cloud-init.log    dmesg.0           dpkg.log         kern.log
auth.log          cloud-init-output.log dmesg.1.gz       faillog          landscape
```

```
16 servidor in /var/log
♦ → zcat dmesg.3.gz
```

Un ejemplo de un trozo de todo el contenido que nos saca:

```
[ 6.218655] kernel: audit: type=1400 audit(1760627432.813:7): apparmor="STATUS" operation="profile_load" profile
"unconfined" name="brave" pid=525 comm="apparmor_parser"
[ 6.221454] kernel: audit: type=1400 audit(1760627432.816:8): apparmor="STATUS" operation="profile_load" profile
"unconfined" name="buildah" pid=527 comm="apparmor_parser"
[ 6.224050] kernel: snd_intel8x0 0000:00:05.0: allow list rate for 1028:0177 is 48000
[ 6.236365] kernel: audit: type=1400 audit(1760627432.831:9): apparmor="STATUS" operation="profile_load" profile
"unconfined" name="busybox" pid=528 comm="apparmor_parser"
[ 6.243160] kernel: audit: type=1400 audit(1760627432.837:10): apparmor="STATUS" operation="profile_load" profil
"unconfined" name="cam" pid=529 comm="apparmor_parser"
[ 6.247147] kernel: audit: type=1400 audit(1760627432.841:11): apparmor="STATUS" operation="profile_load" profil
"unconfined" name="ch-checkns" pid=531 comm="apparmor_parser"
[ 8.482006] kernel: cfg80211: Loading compiled-in X.509 certificates for regulatory database
[ 8.482190] kernel: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 8.482286] kernel: Loaded X.509 cert 'wens: 61c038651aabdcf94bd0ac7ff06c7248db18c600'
[ 8.520361] kernel: e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 12.777427] kernel: kauditd_printk_skb: 109 callbacks suppressed
[ 12.777431] kernel: audit: type=1400 audit(1760627439.372:121): apparmor="STATUS" operation="profile_replace" in
o="same as current profile, skipping" profile="unconfined" name="rsyslogd" pid=829 comm="apparmor_parser"
[ 13.667925] kernel: NET: Registered PF_QIPCRTR protocol family
```


6. Redirección, Tuberías y Filtros

1. Guarda la lista de archivos de tu directorio home (formato largo) en un archivo `mis_archivos.txt`.

```
16 servidor in /home
♦ → ls -l ~ > ~/mis_archivos.txt

16 servidor in /home
♦ → ls -l ~/mis_archivos.txt
-rw-rw-r-- 1 akoarguel akoarguel 384 oct 20 16:56 /home/akoarguel/mis_archivos.txt

16 servidor in /home
♦ → cat ~/mis_archivos.txt
total 20
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta01
-rw-rw-r-- 1 akoarguel akoarguel 692 oct 20 16:54 Carpeta01.zip
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta02
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta03
-rw-rw-r-- 1 akoarguel akoarguel 0 oct 20 16:56 mis_archivos.txt
drwxrwxr-x 3 akoarguel akoarguel 4096 oct 20 16:39 PR_01.1

16 servidor in /home
♦ → _
```

`ls -l` lista los archivos y carpetas del directorio y muestra en formato largo.

`>` es un redireccionador de salida (en este caso a la carpeta raíz).

2. Sin borrar el contenido anterior, añade la fecha y la hora al final del archivo `mis_archivos.txt`.

Usamos `>>` en vez de `>` para que no lo reemplace.

```
16 servidor in /home
♦ → date >> ~/mis_archivos.txt

16 servidor in /home
♦ → cat ~/mis_archivos.txt
total 20
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta01
-rw-rw-r-- 1 akoarguel akoarguel 692 oct 20 16:54 Carpeta01.zip
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta02
drwxrwxr-x 2 akoarguel akoarguel 4096 oct 20 16:52 Carpeta03
-rw-rw-r-- 1 akoarguel akoarguel 0 oct 20 16:56 mis_archivos.txt
drwxrwxr-x 3 akoarguel akoarguel 4096 oct 20 16:39 PR_01.1
lun 20 oct 2025 16:59:42 UTC

16 servidor in /home
♦ → _
```

3. Usa `grep` y una tubería (`|`) para contar el número de directorios que hay en `/etc` . Pista `ls -l grep` .

Las tuberías permite conectar dos comandos.

```
17 servidor in ~
◆ → ls -l /etc | grep ^d | wc -l
107
```

`ls -l /etc` → Lista el contenido de `/etc` en formato largo

`grep ^d` → Filtra solo las líneas que empiezan con `d`, es decir, los directorios.

`wc -l` → Cuenta el número de líneas que pasan por la tubería.

4. Muestra las 10 últimas líneas del archivo `/etc/passwd` y, usando otra tubería, extrae solo los nombres de usuario (primer campo).

```
17 servidor in ~
◆ → tail -n 10 /etc/passwd | cut -d: -f1
polkitd
syslog
uidd
tcpdump
tss
landscape
fwupd-refresh
usbmux
akoarguel
vboxadd
```

`tail -n 10` → Muestra las 10 últimas líneas del archivo.

`cut -d: -f1` → selecciona el primer campo, que en `/etc/passwd` corresponde al nombre de usuario.

5. Muestra una lista de todos los procesos del sistema (`ps aux`), ordénala por uso de CPU (tercera columna) y muestra solo las 5 líneas superiores.

```

17 servidor in ~
◆ → ps aux --sort=%cpu | head -5
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2  0.0  0.0      0     0 ?        S    17:07   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    17:07   0:00 [pool_workqueue_release]
root         4  0.0  0.0      0     0 ?        I<   17:07   0:00 [kworker/R-rcu_g]
root         5  0.0  0.0      0     0 ?        I<   17:07   0:00 [kworker/R-rcu_p]

17 servidor in ~
◆ →

```

`ps aux` → Muestra todos los procesos del sistema, con información como usuario, PID, %CPU, %MEM, comando, etc.

`--sort=%cpu` → Ordena los procesos por uso de CPU de mayor a menor ("`-`" indica orden descendente)

`head -5` → Muestra las 5 primeras líneas.

6. ¿Cuál es la diferencia entre usar `>` y `>>` para redirigir la salida de un comando a un archivo? Demuéstralo con un ejemplo.

Símbolo	Función	Qué hace con el archivo
<code>></code>	Redirección simple	Sobrescribe el archivo con la salida del comando. Si no existe, lo crea.
<code>>></code>	Redirección append	Añade la salida al final del archivo. Si no existe, lo crea.

```

17 servidor in /home
◆ → echo "Primera linea" > ~/prueba.txt

17 servidor in /home
◆ → cat ~/prueba.txt
Primera linea

```

7. Ejecuta `find /etc -name "*.conf"` . Redirige la salida estándar a un archivo `config_files.txt` y los errores (si los hay) a `errors.txt` .

```
18 servidor in ~
♦ → find /etc -name "*.conf" > config_files.txt 2> errors.txt

18 servidor in ~
♦ → _
```

Salida estándar (fragmento):

```
/etc/nsswitch.conf
/etc/apparmor/parser.conf
/etc/modprobe.d/amd64-microcode-blacklist.conf
/etc/modprobe.d/iwlwifi.conf
/etc/modprobe.d/blacklist-framebuffer.conf
/etc/modprobe.d/intel-microcode-blacklist.conf
/etc/modprobe.d/blacklist-rare-network.conf
/etc/modprobe.d/blacklist.conf
/etc/modprobe.d/mdadm.conf
/etc/modprobe.d/dkms.conf
/etc/modprobe.d/blacklist-firewire.conf
/etc/modprobe.d/blacklist-ath_pci.conf
/etc/ld.so.conf
/etc/hdparm.conf
/etc/rsyslog.conf
/etc/ucf.conf
/etc/fwupd/fwupd.conf
/etc/fwupd/remotes.d/lvfs-testing.conf
/etc/fwupd/remotes.d/vendor-directory.conf
/etc/fwupd/remotes.d/lvfs.conf
/etc/ufw/ufw.conf
/etc/ufw/sysctl.conf
/etc/udisks2/udisks2.conf
/etc/gai.conf
/etc/updatedb.conf
```

Errores:

```

18 servidor in ~
◆ → cat errors.txt
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/rules.d': Permission denied
find: '/etc/multipath': Permission denied
find: '/etc/credstore.encrypted': Permission denied
find: '/etc/credstore': Permission denied

18 servidor in ~
◆ →

```

7. Scripts Básicos

1. Crea un script que imprima tu nombre de usuario y el directorio de trabajo actual usando las variables de entorno correspondientes.

La sentencia básica para ver el nombre de usuario y el directorio actual sería la siguiente:

```
echo "Usuario: $(whoami), Directorio actual $(pwd)"
```

Vamos a hacerlo un script:

```

18 servidor in ~/PR_01.1/scripts
◆ → echo 'echo "Usuario: $(whoami), Directorio actual: $(pwd)"' > new_script

18 servidor in ~/PR_01.1/scripts
◆ → cat new_script
echo "Usuario: $(whoami), Directorio actual: $(pwd)"

18 servidor in ~/PR_01.1/scripts
◆ →

```

2. Haz el script anterior ejecutable solo para ti (`chmod u+x ...`) y ejecútalo. Luego, intenta ejecutarlo como otro usuario (si es posible) o explica qué pasaría.

Este es nuestro script "new_script.sh"

```
18 servidor in ~/PR_01.1/scripts
♦ → ls
new_script.sh

18 servidor in ~/PR_01.1/scripts
♦ →
```

Hacemos que solo pueda ser ejecutable para mi usuario:

```
18 servidor in ~/PR_01.1/scripts
♦ → chmod u+x ~/PR_01.1/scripts/new_script.sh

18 servidor in ~/PR_01.1/scripts
♦ → ls -l ~/PR_01.1/scripts/new_script.sh
-rwxr--r-- 1 akoarguel akoarguel 53 oct 20 18:45 /home/akoarguel/PR_01.1/scripts/new_script.sh

18 servidor in ~/PR_01.1/scripts
♦ →
```

Si yo lo ejecuto sale la información correctamente:

```
18 servidor in ~/PR_01.1/scripts
♦ → ~/PR_01.1/scripts/new_script.sh
Usuario: akoarguel, Directorio actual: /home/akoarguel/PR_01.1/scripts

18 servidor in ~/PR_01.1/scripts
♦ → _
```

¿Se puede ejecutar con otro usuario?

- Si otro usuario intenta ejecutar el script saldrá un resultado parecido al siguiente:

```
-bash: /home/akoarguel/PR_01.1/scripts/new_script.sh: Permission denied
```

3. **Modifica el script para que acepte un argumento. Si el argumento es "hola", debe imprimir "mundo". Si es cualquier otra cosa, no debe imprimir nada.**

Vamos a utilizar nano para modificar el script de una forma más cómoda.

```
19 servidor in ~/PR_01.1/scripts
◆ → nano new_script.sh _
```

```
GNU nano 7.2
#!/bin/bash

# Comprobar si se recibe el argumento
if [ "$1" = "hola" ]; then
    echo "mundo"
fi
```

`#!/bin/bash` → Esto no es un comentario normal, sino un shebang. Significa que el script se ejecutará usando **Bash** (el shell de Linux ubicado en `/bin/bash`).

1. `$1` → Representa el primer argumento que se pasa al script
2. `if ["$1" = "hola"]` → Comprueba si el argumento es exactamente "hola"
3. `echo "mundo"` → imprime `mundo` solo si la condición se cumple

Vamos a ejecutarlo. Primero hacemos que sea ejecutable. (No es necesario ya que ya se hizo anteriormente)

```
19 servidor in ~
◆ → chmod u+x ~/PR_01.1/scripts/new_script.sh

19 servidor in ~
◆ → _
```

Y ejecutamos:

```
19 servidor in ~/PR_01.1/scripts
♦ → ./new_script.sh hola
    mundo

19 servidor in ~/PR_01.1/scripts
♦ →
```

Si no escribimos "hola":

```
19 servidor in ~/PR_01.1/scripts
♦ → ./new_script.sh

19 servidor in ~/PR_01.1/scripts
♦ → _
```

4. Mejora el script anterior para que, si no se proporciona ningún argumento, muestre un mensaje de uso: "Error: Debe proporcionar un argumento."

Modificamos con nano el script:

```
GNU nano 7.2 new_script
#!/bin/bash

# Comprobar si hay argumento
if [ $# -eq 0 ]; then
    echo "Error: Debe proporcionar un argumento."
    exit 1
fi

# Comprobar si el argumento es "hola"
if [ "$1" = "hola" ]; then
    echo "mundo"
fi
```

1. `$#` → número de argumentos pasados al script
2. `if [$# -eq 0]` → si equivale a 0

3. `echo ...` → imprime el error
4. `exit 1` → termina el script con un código de error (1 indica fallo, 0 no hay errores)

Ejecutamos sin argumento:

```
19 servidor in ~/PR_01.1/scripts
♦ → ./new_script.sh
Error: Debe proporcionar un argumento.

19 servidor in ~/PR_01.1/scripts
♦ →
```

5. Escribe un script que reciba dos números. Debe imprimir "iguales" si son iguales y "diferentes" si no lo son.

Nuevo script:

```
GNU nano 7.2 script2.sh
#!/bin/bash

# Comprobar si se pasaron dos argumentos
if [ $# -ne 2 ]; then
    echo "Error: Debe proporcionar dos numeros."
    exit 1
fi

# Guardamos los argumentos en variables
num1 = $1
num2 = $2

# Comparar los numeros
if [ "$num1" -eq "$num2" ]; then
    echo "iguales"
else
    echo "diferentes"
fi
```

1. `$#` → número de argumentos.
 - `if [$# -ne 2]` → comprueba que se pasen exactamente **dos** argumentos.

2. `num1=$1` y `num2=$2` → guardan los argumentos en variables.
3. `eq` → operador de comparación **numérica**.
4. `if ... else` → imprime `iguales` o `diferentes` según corresponda.

Hacemos ejecutable el script y que solo lo ejecute el usuario.

```
19 servidor in ~/PR_01.1/scripts
♦ → chmod u+x ~/PR_01.1/scripts/script2.sh

19 servidor in ~/PR_01.1/scripts
♦ →
```

Y ejecutamos:

```
19 servidor in ~/PR_01.1/scripts
♦ → ./script2.sh 2 2
iguales

19 servidor in ~/PR_01.1/scripts
♦ → ./script2.sh 2 1
diferentes

19 servidor in ~/PR_01.1/scripts
♦ → ./script2.sh 2
Error: Debe proporcionar dos numeros.

19 servidor in ~/PR_01.1/scripts
♦ →
```

6. Escribe un script que, dado un directorio como argumento, use un bucle `for` para iterar sobre su contenido (`ls $1`) y añada la extensión `.bak` a cada archivo.

```
#!/bin/bash

# Comprobamos argumentos
if [ $# -ne 1 ]
then
    echo "Error: Debe proporcionar un directorio."
    exit 1
fi

dir="$1"

# Comprobar si el argumento es un directorio
if [ ! -d "$dir" ]
then
    echo "Error: $dir no es un directorio válido"
    exit 1
fi

# Iteramos
for archivo in "$dir"/*
do
    if [ -f "$archivo" ]
    then
        mv "$archivo" "$archivo.bak"
    fi
done
```

Explicación del código:

`if [$# -ne 1]` → asegura que se pase solo un argumento.

`if [! -d "$dir"]` → comprueba si el argumento es un directorio real

`for archivo in "$dir"/*` → itera el contenido del directorio y si encuentra un archivo (`if [-f "$archivo"]`) añade al archivo la extensión `.bak` .

Antes de ejecutar, este es nuestro árbol:

```
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ tree
.
├── carpeta01
├── carpeta02
├── prueba2
├── prueba3
├── prueba4.txt
├── scripts
│   └── script3.sh
```

Ejecutamos:

```
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ ./scripts/script3.sh ~/PR_01.1/
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ
```

Comprobamos el resultado:

```
└─◆ akoarguel at ◆ servidor in ◆ ~/PR_01.1 in
└─λ tree
.
├── carpeta01
├── carpeta02
├── prueba2.bak
├── prueba3.bak
├── prueba4.txt.bak
├── scripts
│   └── script3.sh
4 directories, 4 files
```