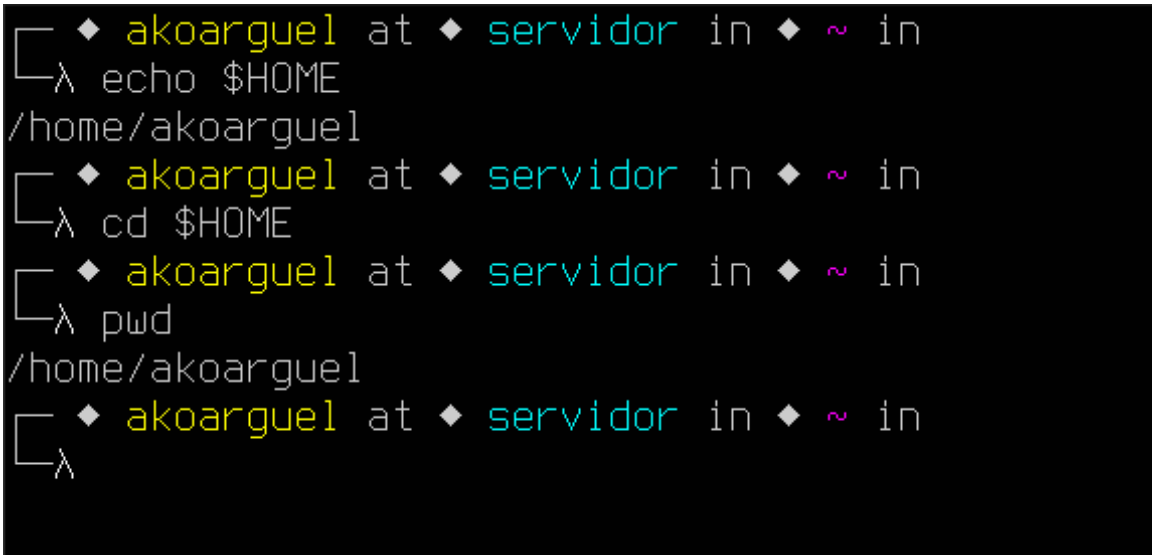
PR_01.1

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

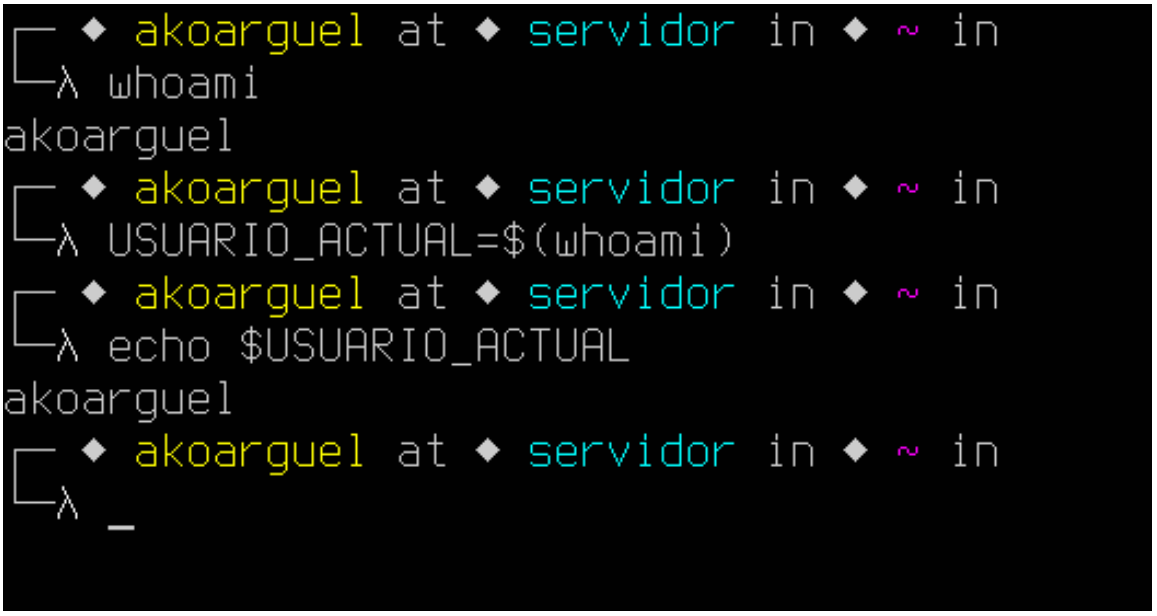
1. Conceptos Básicos y Variables

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**



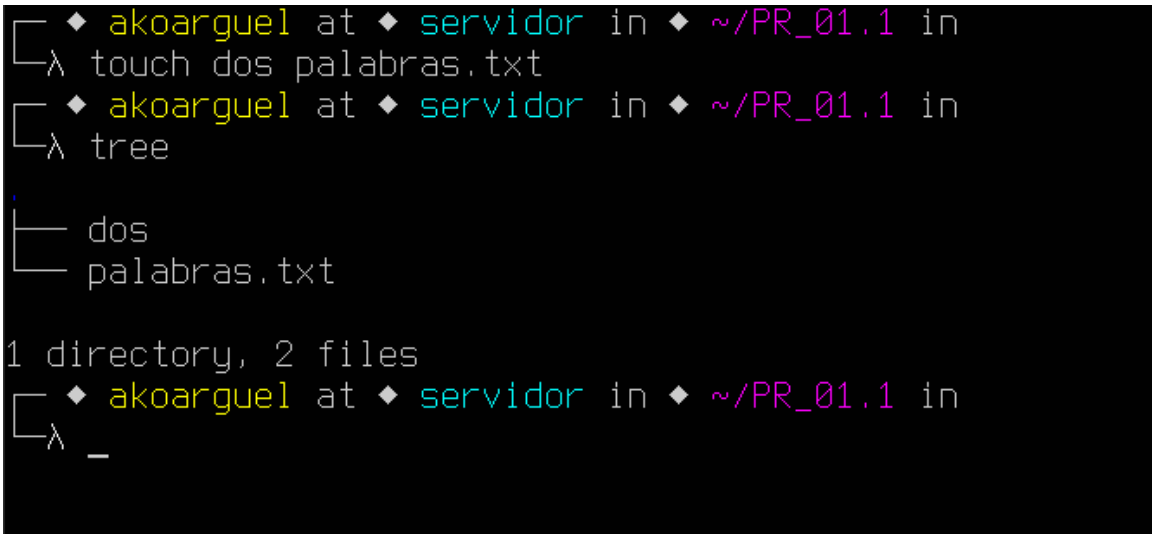
Utilizamos `echo` para mostrar la información por pantalla, usamos esa misma variable para ir hasta la ubicación con `cd` y comprobamos si estamos en la ubicación deseada con `pwd`.

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.**



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.**



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
```

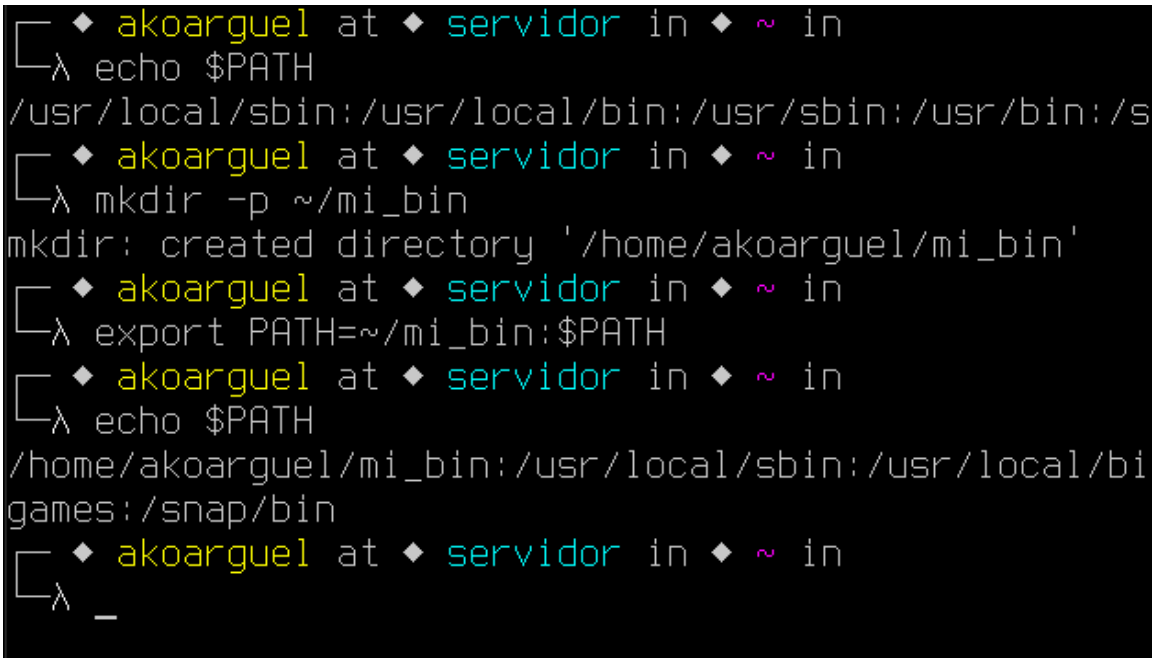
4. **Usa el comando `type` para averiguar si `ls` y `cd` son internos o externos al shell. ¿Qué diferencia prácticas crees que implica esto?**

```
┌─ ◆ 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 qué, los comandos externos son programas independientes que se ejecutan independientemente.

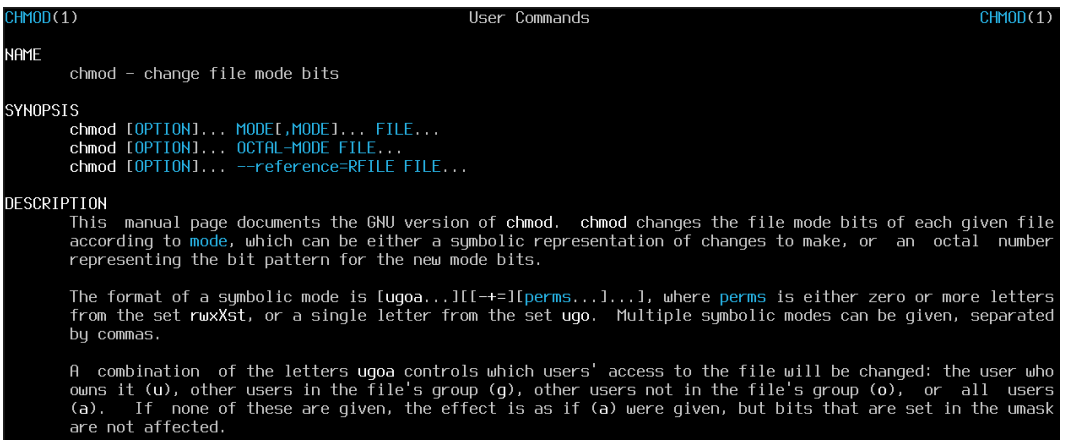
5. **Muestra tu `PATH` actual. Crea un directorio `~/mi_bin` y añádelo temporalmente al principio de tu `PATH`. Verifica que el cambio se ha realizado correctamente.**



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`.

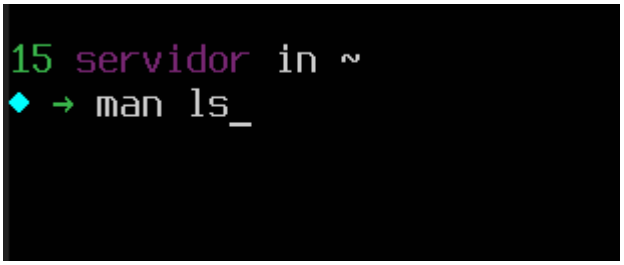
2. Obtener Ayuda y Localizar Archivos

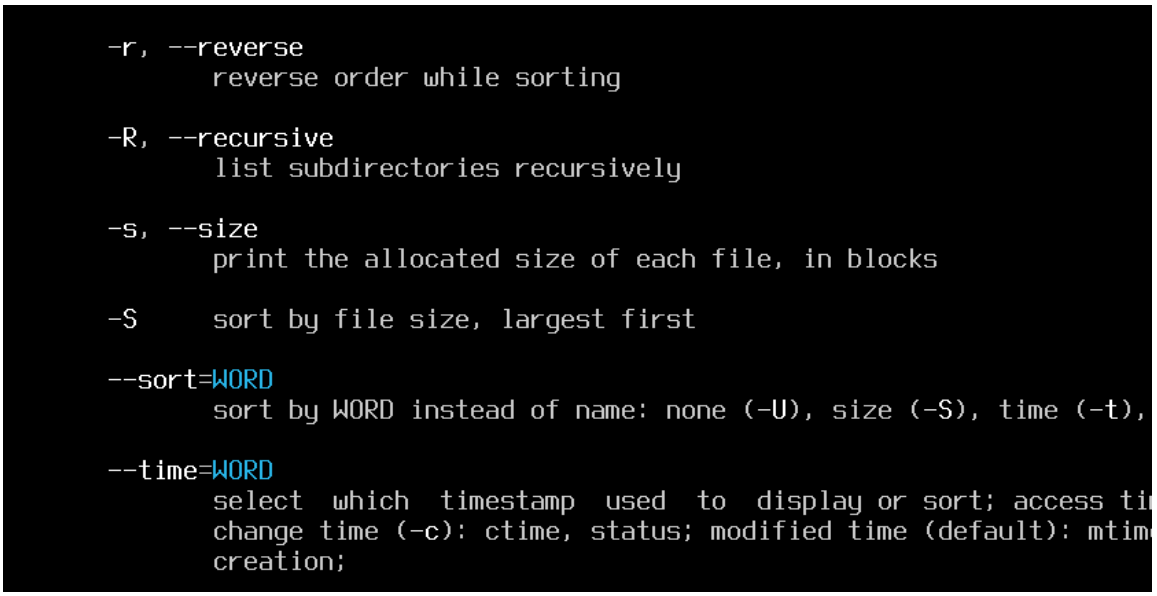
1. **Abre la página del manual para el comando `chmod`. ¿En qué sección del manual se encuentra? ¿Qué indica ese número de sección sobre el tipo de comando?**



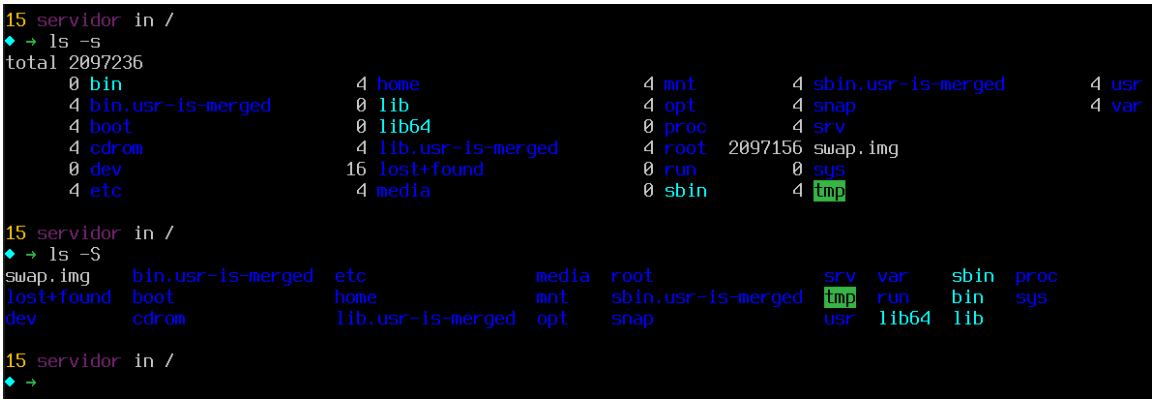
El manual de `chmod` se obtiene con el comando `man`. El manual se encuentra en la *sección 1* que parece indicar que es la parte con comandos de usuario ejecutables.

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

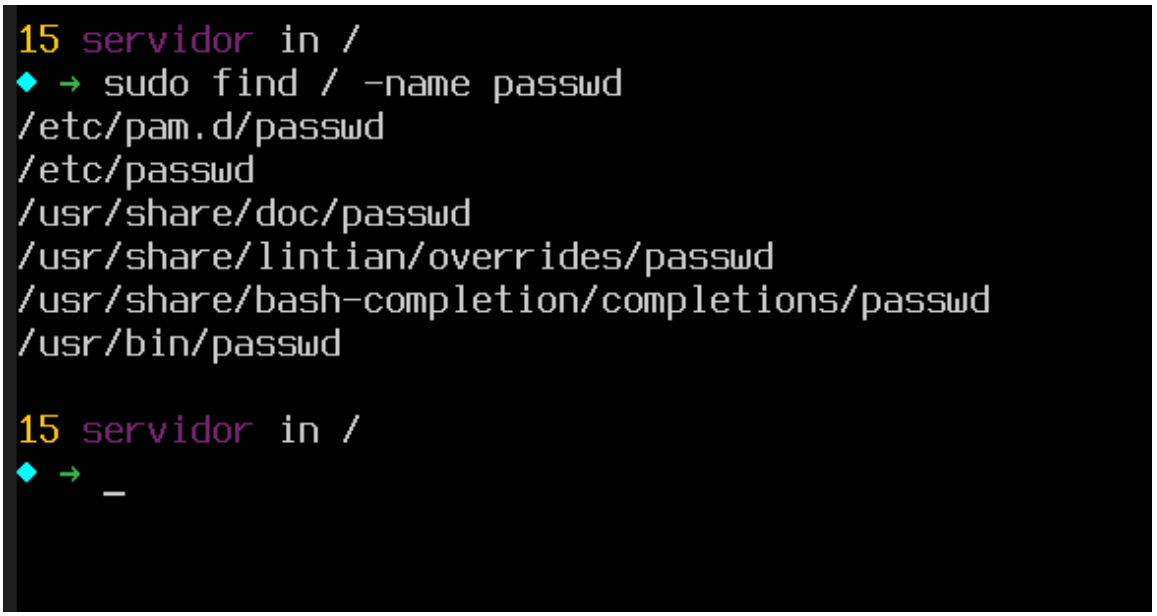




Encontramos dos opciones para organizar por tamaño. `ls -s` organiza por tamaño cada archivo en bloques y `ls -S` organiza los archivos por tamaño, primero los archivos con mayor tamaño.



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.**



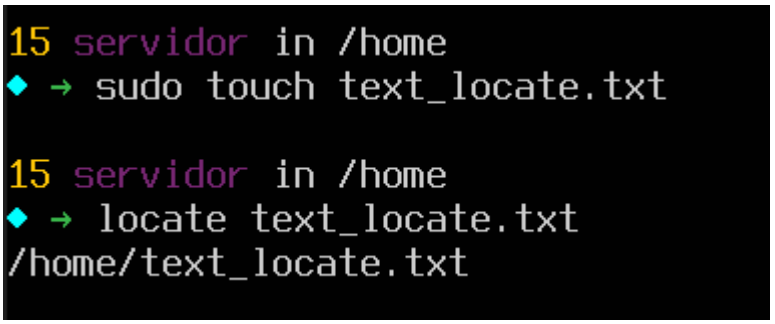
   El comando `find` requiere de usar `sudo` por tema de derechos de administrador. Además le añadimos la etiqueta `-name` para buscar en concreto *passwd*.

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*.

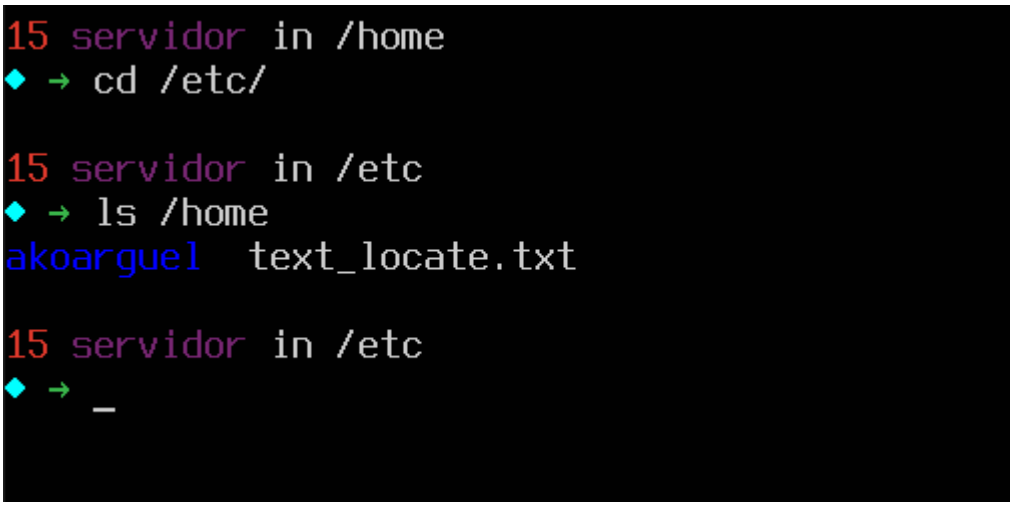


5. **Basado en el ejercicio anterior, ¿qué comando (probablemente con sudo) necesitas ejecutar para que locate sí encuentre tu archivo? Ejecútalo y verifica que ahora sí lo encuentras.**

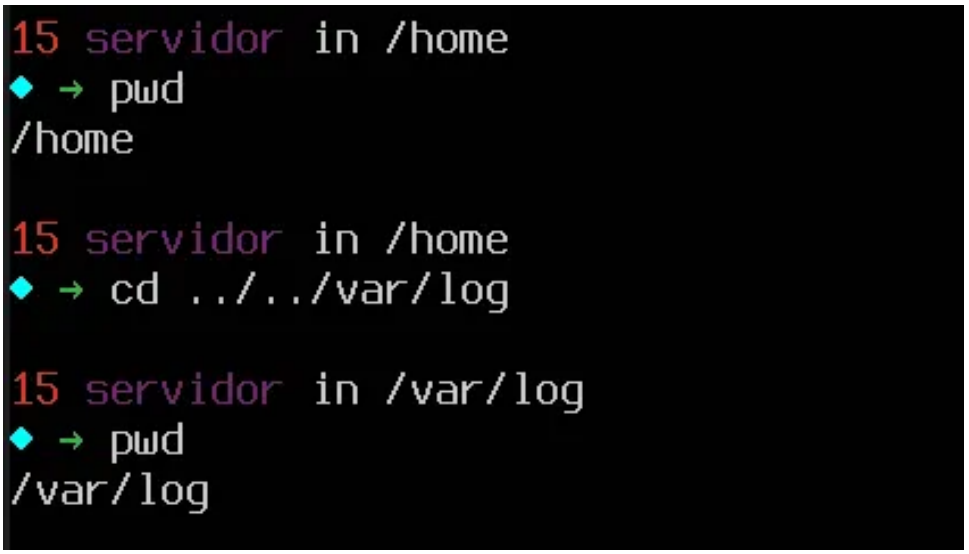
   Como se puede ver en el ejercicio anterior, si es necesario usar `sudo` para crear el archivo pero no para localizarlo.

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 `~`.**



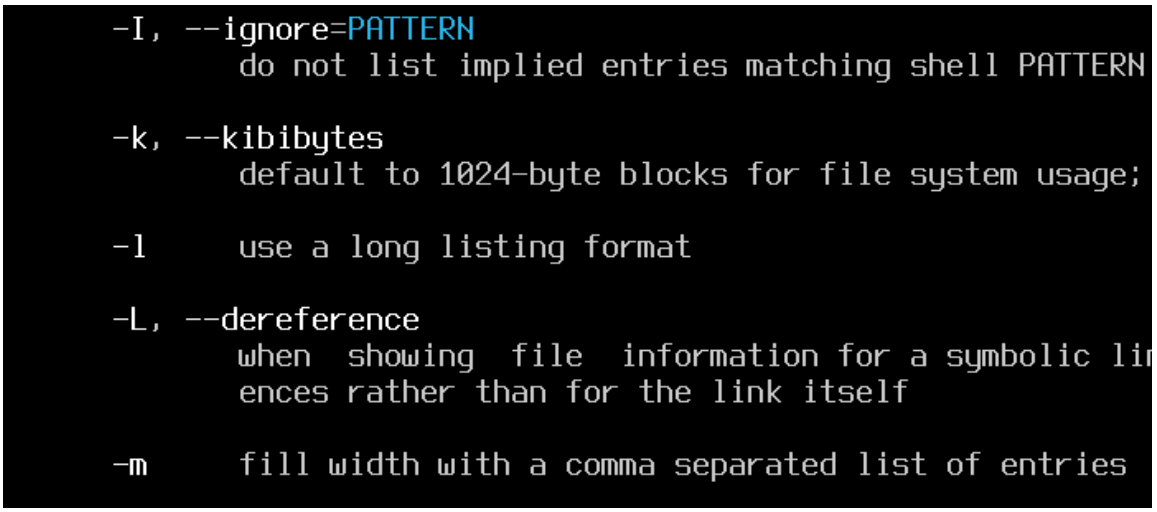
2. **Desde tu directorio `home`, navega a `/` y luego a `var` y finalmente a `log` usando una sola línea de comando y rutas relativas**



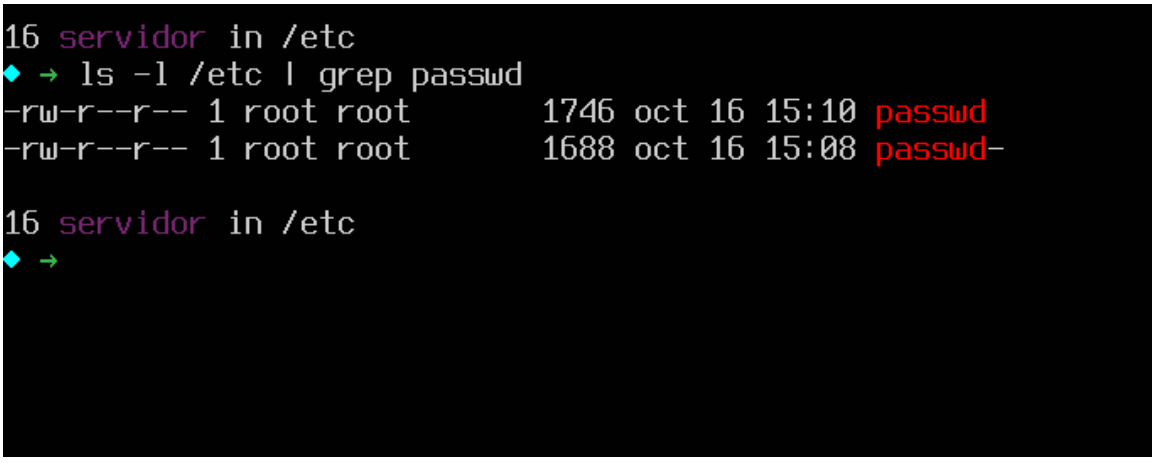
3. **Lista el contenido de `/etc` en formato largo. En la salida, identifica el propietario, el grupo y los permisos del archivo `passwd`.**

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

   ```
   man ls
   ```



   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*.



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 /etce` → muestra la información del tamaño en bytes. (2048, 4096, etc.).
- `ls -lh /etce` → 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 /
◆ → _
```

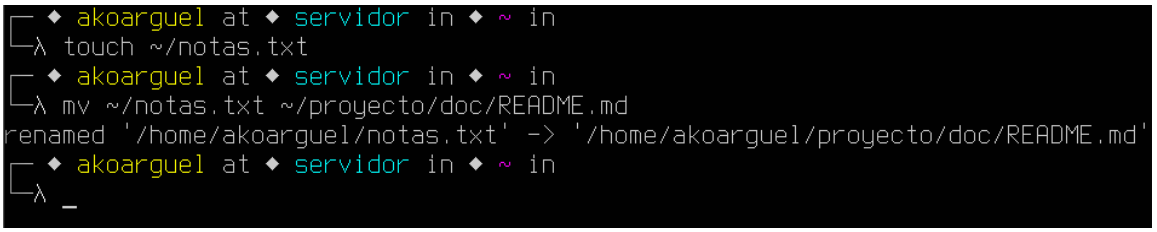
Lo peligroso del comando `R` es que muestra el directorio indicando todos sus subdirectorios, uno dentro de otro.

> ‼️ 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.

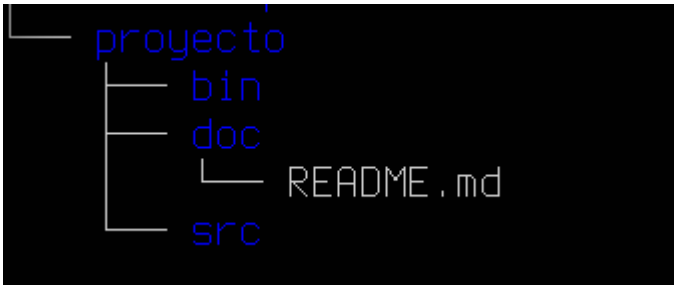
4. Manipulación de Archivos y Directorios

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

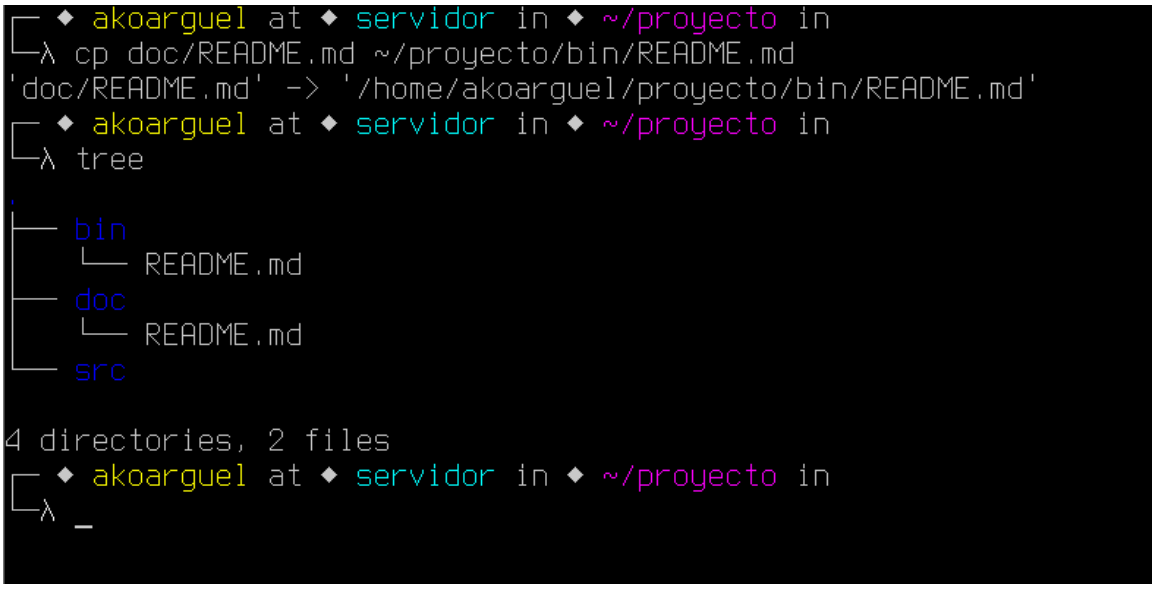
2. **Crea un archivo `~/notas.txt`. Muévelo a `~/proyecto/doc` y, en el mismo comando, renómbralo a README.md**



`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`.**



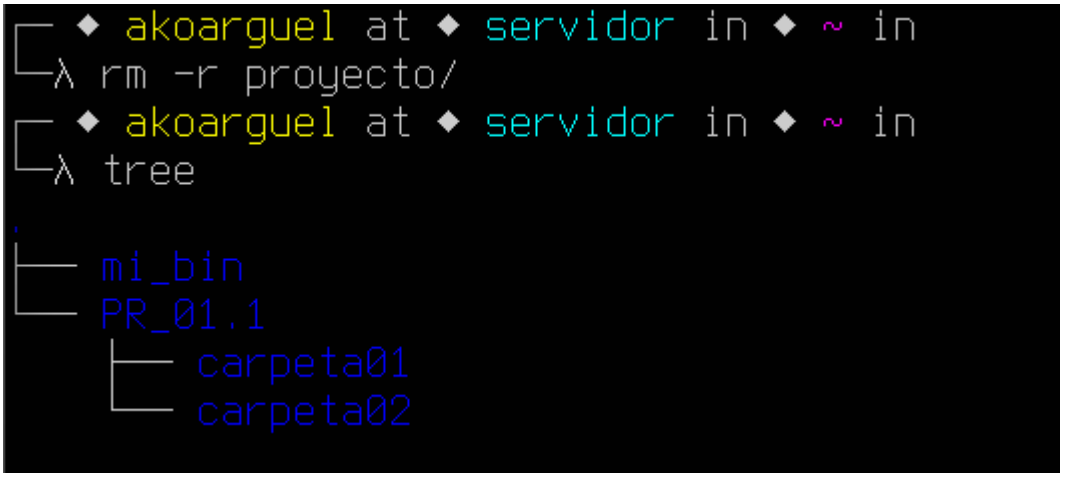
4. **Intente borrar el directorio proyecto con rmdir. ¿Qué error obtienes? Ahora, usa `rm` con la opción correcta para borrar el directorio y todo lo que contiene.**

```
┌─ ◆ 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 que nos saldrá el siguiente error

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

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



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
apport.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
```

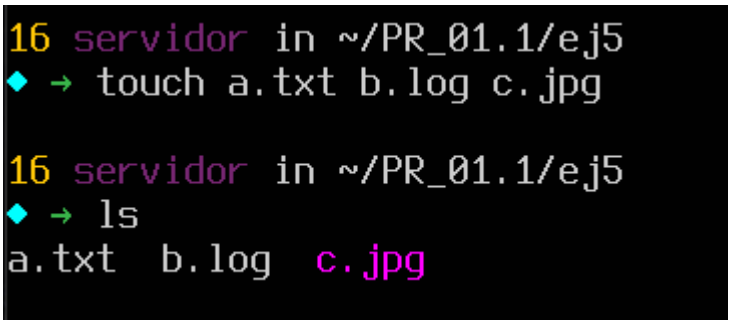
`-x` → extrae archivos del tar

`-z` → indica que está comprimido con gzip

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

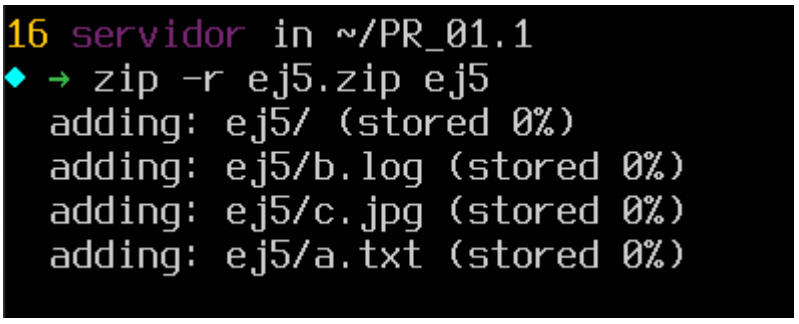
`-C` → destino de extracción

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



Deberemos instalar zip para poder continuar con la tarea.

```
sudo apt install zip
```



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`.

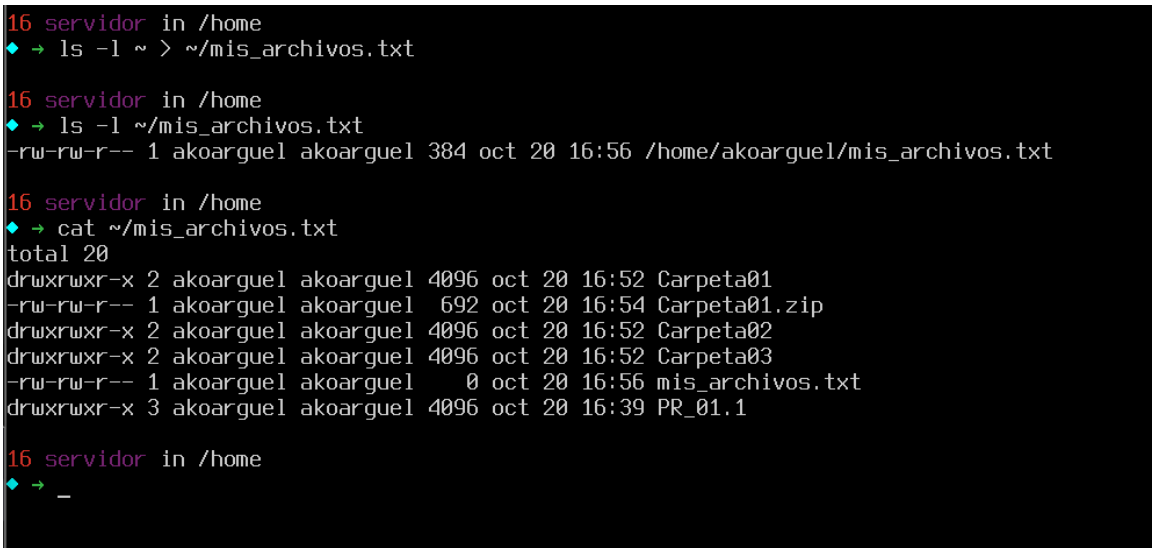
```
16 servidor in /var/log
◆ → ls
alternatives.log  bootstrap.log          dist-upgrade  dmesg.2.gz  installer
apport.log        btmp                   dmesg         dmesg.3.gz  journal
apt               cloud-init.log         dmesg.0       dpkg.log    kern.log
auth.log          cloud-init-output.log  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

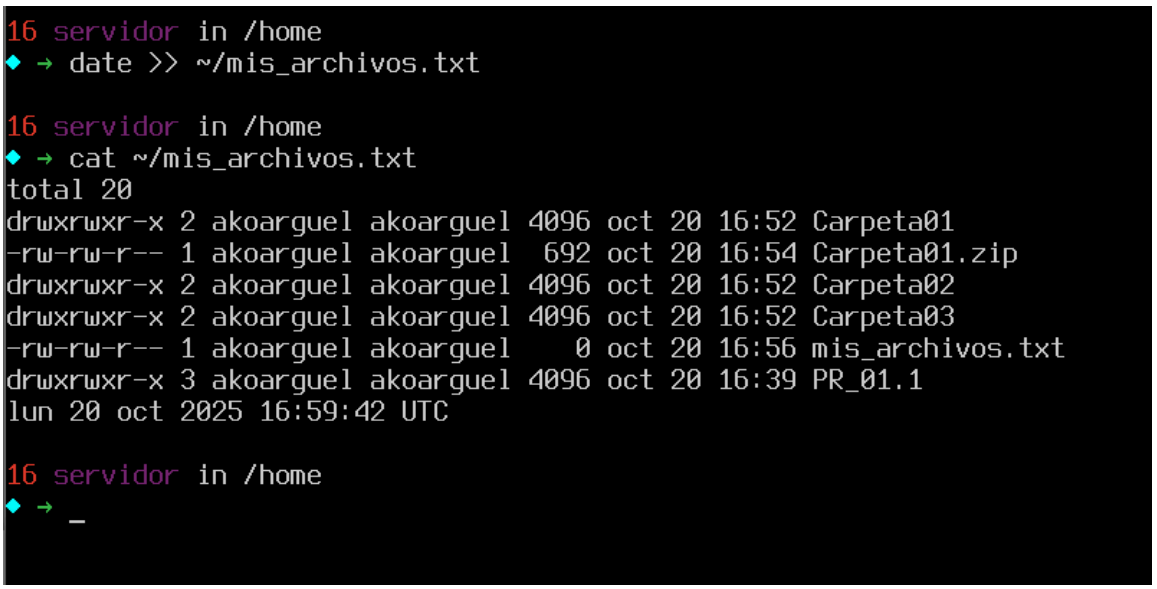
1. **Guarda la lista de archivos de tu directorio home (formato largo) en un archivo** `mis_archivos.txt`**.**



`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.



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

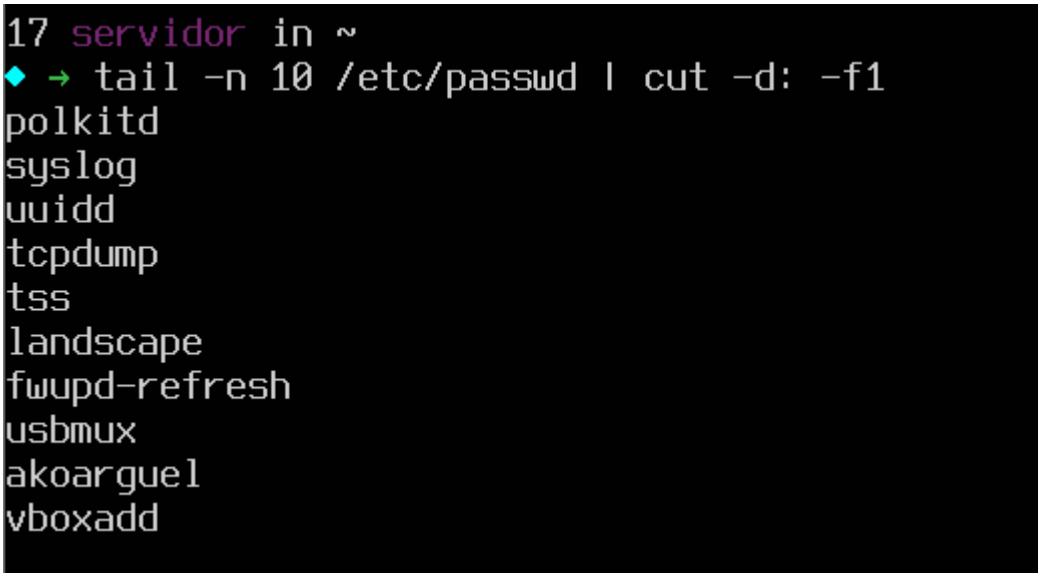
   Las tuberías permite conectar dos comandos.

   `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 tuberia.

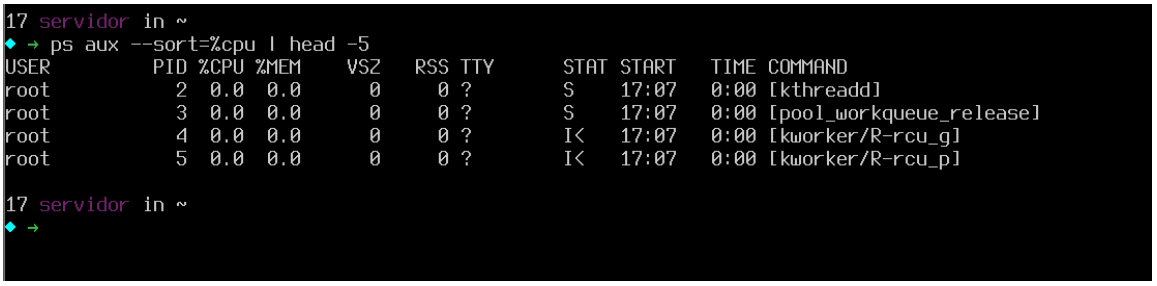
4. **Muestra las 10 últimas líneas del archivo /etc/passwd y, usando otra tubería, extrae solo los nombres de usuario (primer campo).**



   `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.**



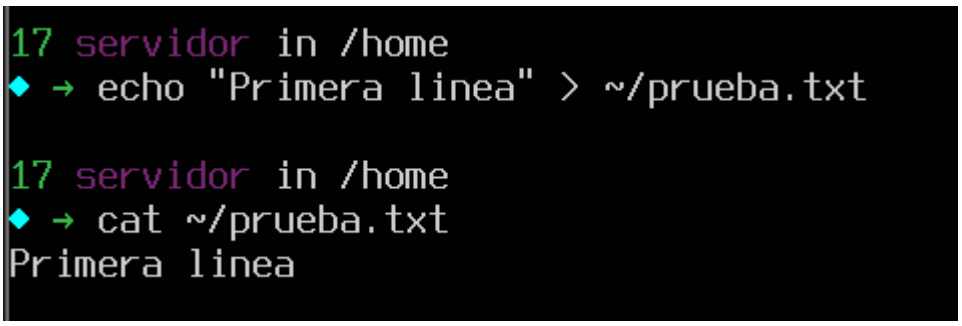
`ps aux` → Muestra todos los procesos del sistema, con información como usuario, PID, %CPU, %MEM, comando, etc.

`--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
`>`	Redirección simple	Sobrescribe el archivo con la salida del comando. Si no existe, lo crea.
`>>`	Redirección append	Añade la salida al final del archivo. Si no existe, lo crea.



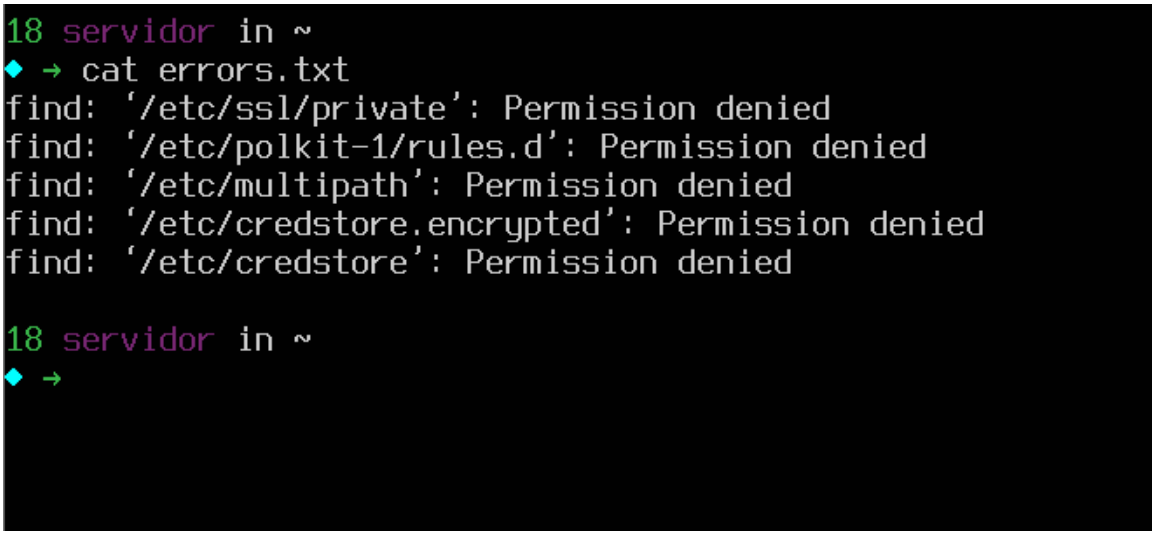
7. **Ejecuta `find /etc -name "*.conf"`. Redirige la salida estándar a un archivo `config_files.txt` y los errores (si los hay) a `errors.txt`.**

```
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:



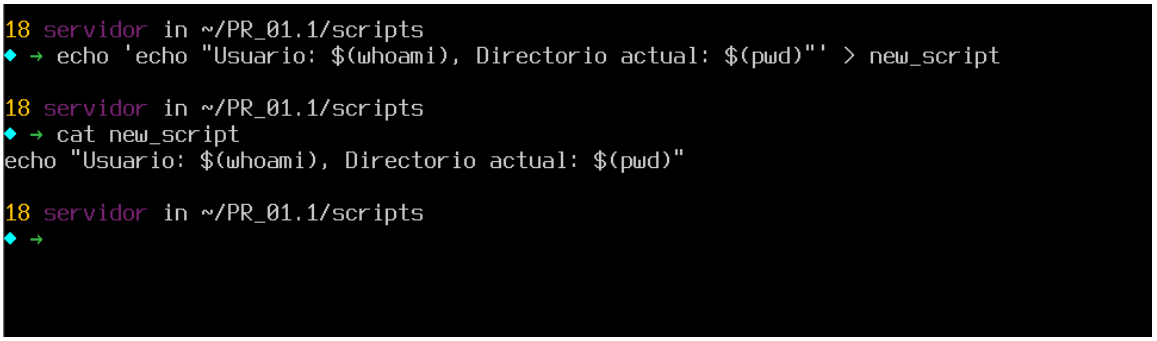
7. Scripts Básicos

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:

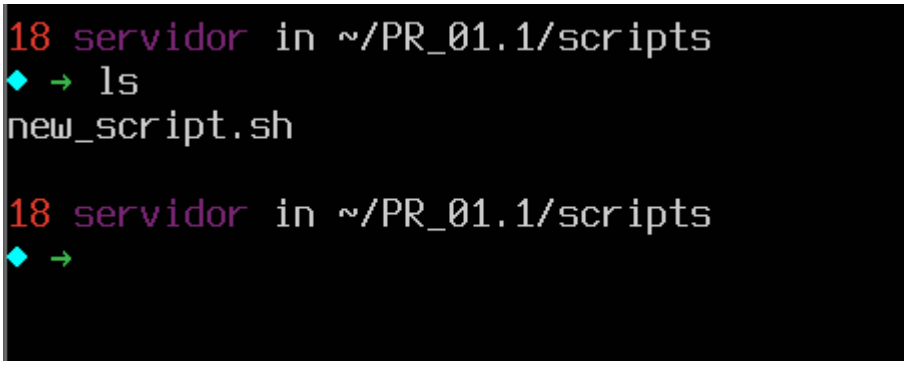
   ```
   echo "Usuario: $(whoami), Directorio actual $(pwd)"
   ```

   Vamos a hacerlo un script:

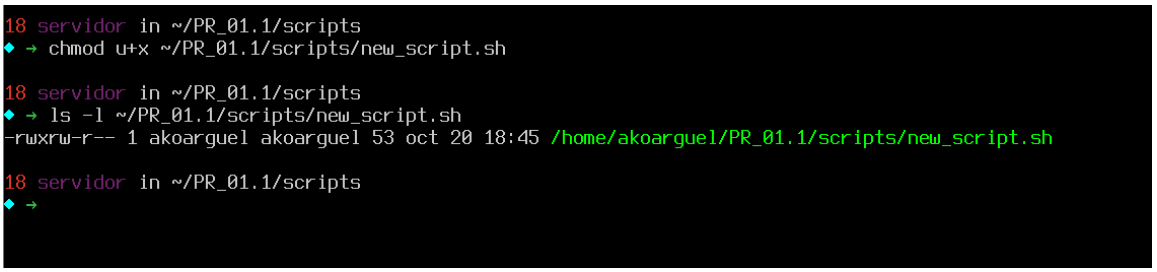


2. **Haz el script anterior ejecutable solo para ti (`chmod u+x ...`) y ejecútalo. Luego, intenta ejecutarlo como otro usuario (si es posible) o explica qué pasaría.**

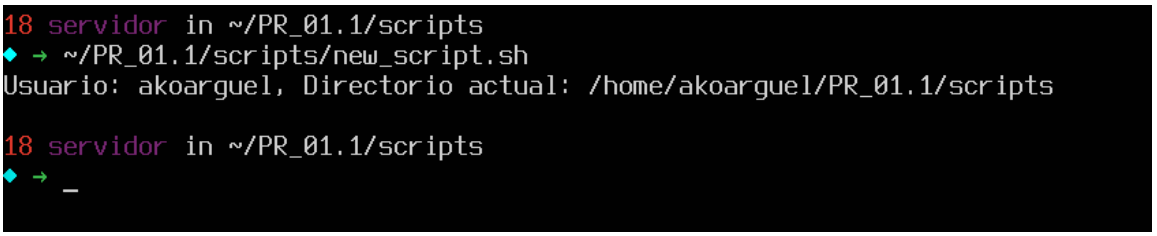
   Este es nuestro script *“new_script.sh”*



Hacemos que solo pueda ser ejecutable para mi usuario:



Si yo lo ejecuto sale la información correctamente:



¿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

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

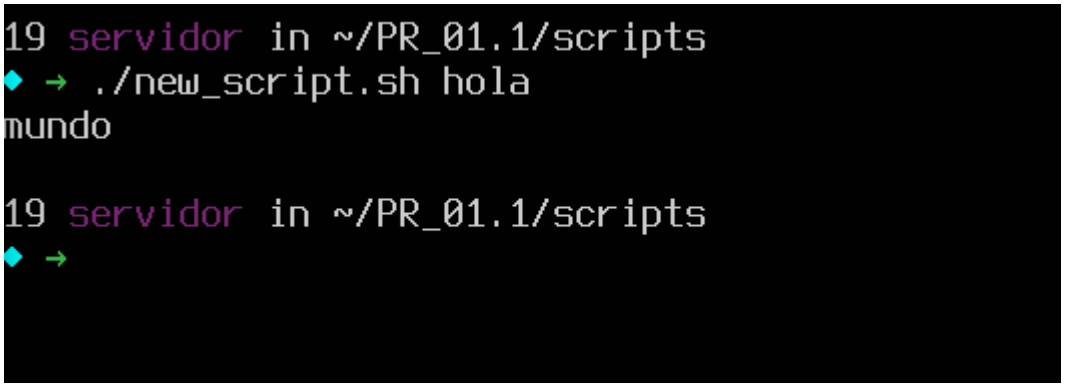
`#!/bin/bash` → Esto no es un comentario normal, sino un shebang. Significa que el script se ejecutará usando **Bash** (el shell de Linux ubicado en `/bin/bash`).

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

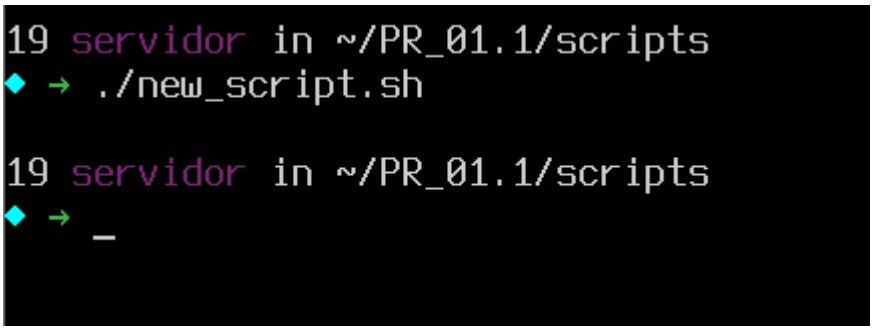
Vamos a ejecutarlo. Primero hacemos que sea ejecutable. (No es necesario ya que ya se hizo anteriormente)

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

19 servidor in ~
◆ → _
```

Y ejecutamos:

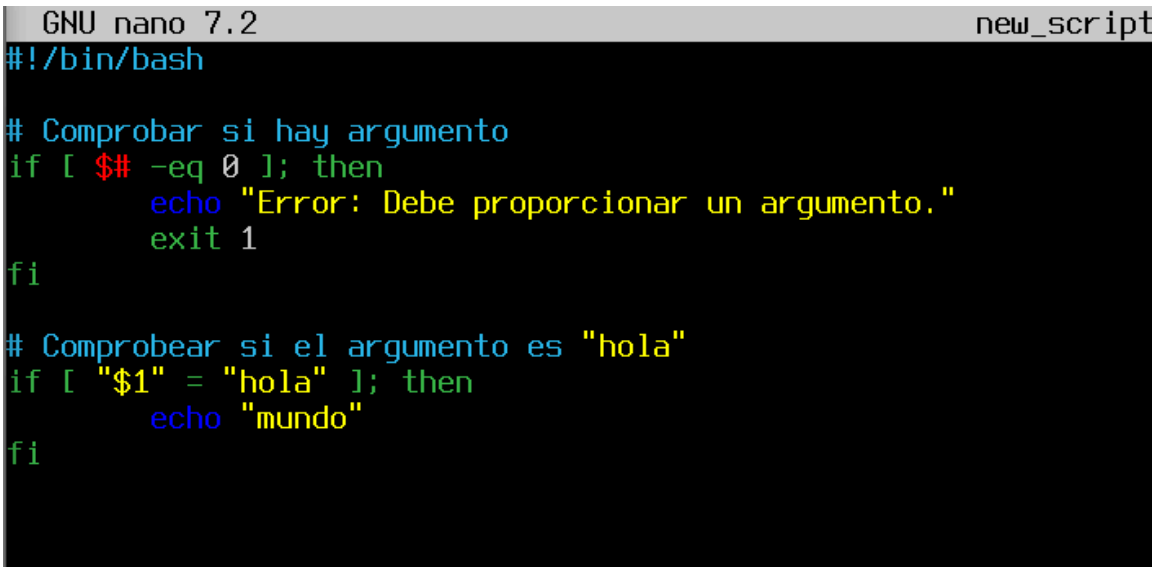


Si no escribimos “hola”:



4. **Mejora el script anterior para que, si no se proporciona ningún argumento, muestre un mensaje de uso: “Error: Deber proporcionar un argumento.”**

   Modificamos con nano el script:



   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.**

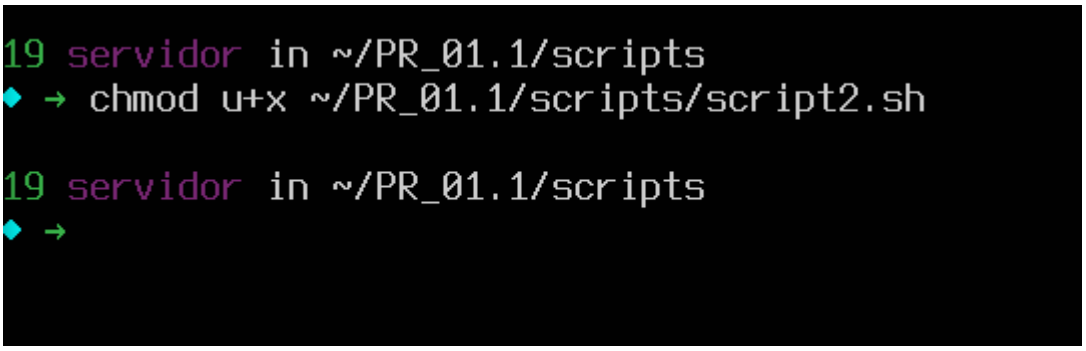
   Nuevo script:

```
 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
```

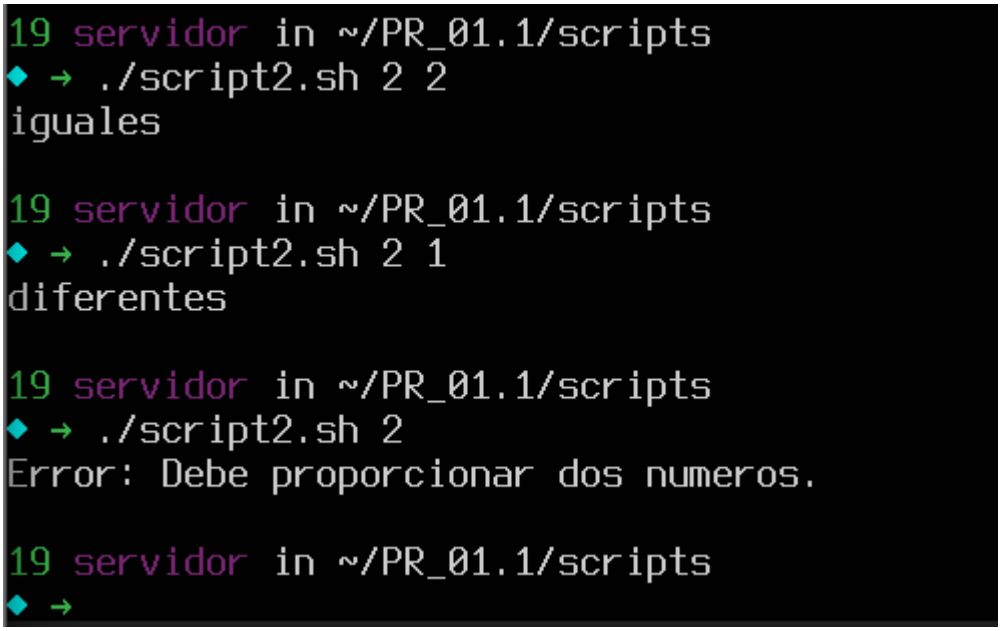
1. `$#` → número de argumentos.
   - `if [ $# -ne 2 ]` → comprueba que se pasen exactamente **dos argumentos**.

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.



Y ejecutamos:



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.**

```bash
#!/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 direcotorio válido"
        exit 1
fi

# Iteramos

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

Explicación del código:

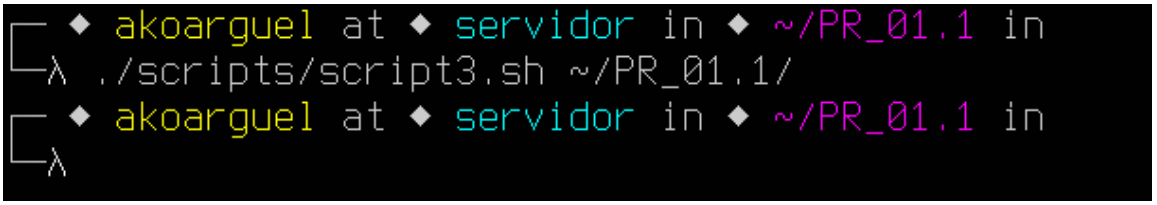
`if [ $# -ne 1 ]` → asegura que se pase solo un argumento.

`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:

Ejecutamos:



Comprobamos el resultado: