

# Práctica 1: Linux básico



Álvaro Santos Romero

## Índice

1. Preparación del entorno	2
2. Órdenes Básicas	3
3. Procesos	6
4. Gestión de Usuarios	8
5. Sistema de ficheros	10
6. Bash Scripting	12

## 1. Preparación del entorno

1. Creamos un vagrantfile con ubuntu. Contenido del vagrantfile:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
  config.vm.network :forwarded_port, guest: 80, host: 4567
  config.vm.provision :shell, path: "bootstrap.sh"
end
```

Figura 1: vagrantfile

- Configuración de red, puerto 80 máquina invitada a puerto 4567 máquina anfitrión.
- Ejecución del script de aprovisionamiento **bootstrap.sh** que configura la máquina durante el arranque.

### 2. Bootstrap.sh

```
#!/usr/bin/env bash
|
apt-get update
apt-get install -y apache2
```

Figura 2: bootstrap

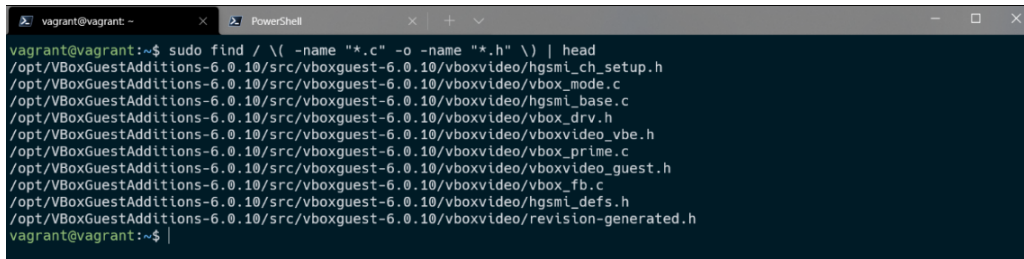
Como se observa en el script, ejecutamos dos comandos:

- a) **apt-get update** — Descarga todas las actualizaciones de los paquetes del sistema.
- b) **apt-get install -y apache2** — Descarga apache2.

## 2. Órdenes Básicas

1. Encontrar los archivos con extensión ".h" o ".c" desde la ruta raíz.

```
sudo find / -type f \( -name "*.c" -o -name "*.h" \) | head
```



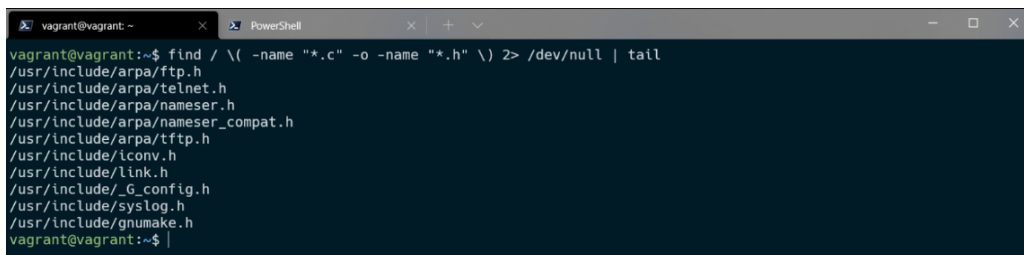
```
vagrant@vagrant:~$ sudo find / \( -name "*.c" -o -name "*.h" \) | head
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/hgsml_ch_setup.h
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vbox_mode.c
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/hgsml_base.c
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vbox_drv.h
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vboxvideo_vbe.h
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vboxvideo_prime.c
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vboxvideo_guest.h
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/vbox_fb.c
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/hgsml_defs.h
/opt/VBoxGuestAdditions-6.0.10/src/vboxguest-6.0.10/vboxvideo/revision-generated.h
vagrant@vagrant:~$
```

Figura 3: ejercicio 1

- **find**
- `\( -name "*.c" -o -name "*.h" \)` — Expresión regular para seleccionar elementos que terminen con esos patrones.
- **head** — Mostrar los 10 primeros elementos.

2. Encontrar los archivos con extensión ".h" o ".c" desde la ruta raíz, enviando los errores al fichero /dev/null.

```
sudo find / -type f \( -name "*.c" -o -name "*.h" \) 2> /dev/null | tail
```



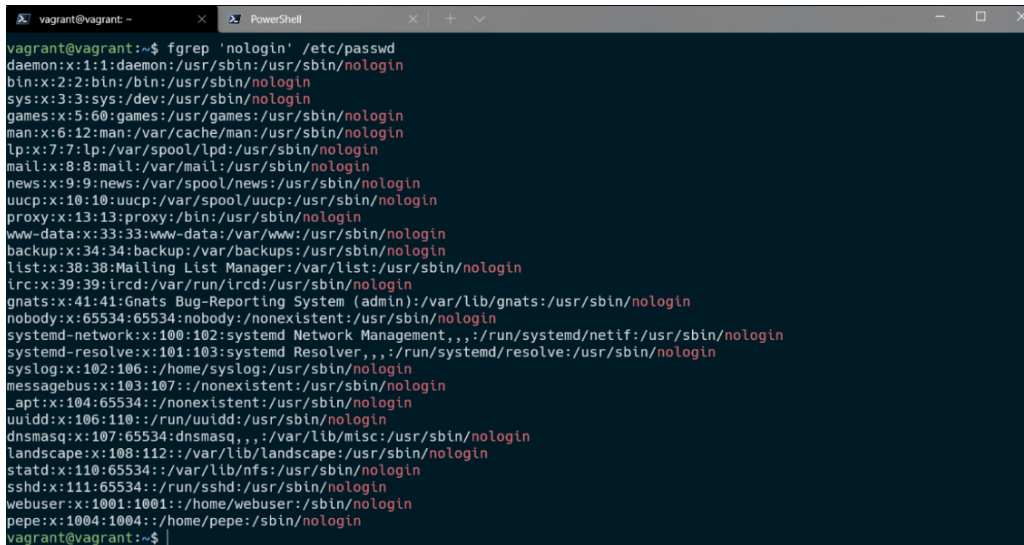
```
vagrant@vagrant:~$ sudo find / -type f \( -name "*.c" -o -name "*.h" \) 2> /dev/null | tail
/usr/include/arpa/ftp.h
/usr/include/arpa/telnet.h
/usr/include/arpa/nameser.h
/usr/include/arpa/nameser_compat.h
/usr/include/arpa/tftp.h
/usr/include/iconv.h
/usr/include/link.h
/usr/include/_G_config.h
/usr/include/syslog.h
/usr/include/gnumake.h
vagrant@vagrant:~$
```

Figura 4: ejercicio 2

- **find**
- `\( -name "*.c" -o -name "*.h" \)` — Expresión regular para seleccionar elementos que terminen con esos patrones.
- **2>/dev/null** — Enviar errores al fichero /dev/null.
- **tail** — Mostrar los 10 primeros elementos.

## 3. Buscar usuarios que puedan iniciar sesión en el equipo.

```
fgrep "nologin" /etc/passwd
```



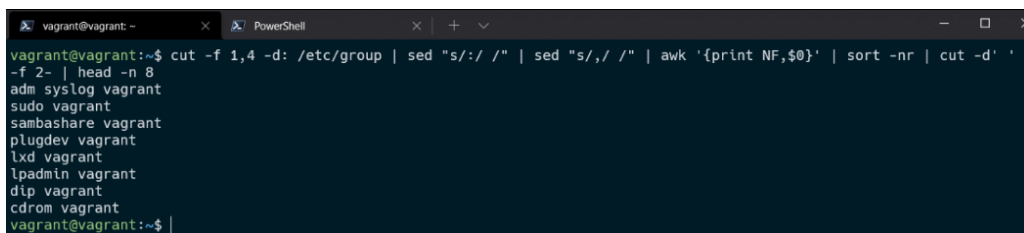
```
vagrant@vagrant:~$ fgrep 'nologin' /etc/passwd
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uidd:x:106:110::/run/uid:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
statd:x:110:65534::/var/lib/nfs:/usr/sbin/nologin
sshd:x:111:65534::/run/ssh:/usr/sbin/nologin
webuser:x:1001:1001::/home/webuser:/sbin/nologin
pepe:x:1004:1004::/home/pepe:/sbin/nologin
vagrant@vagrant:~$
```

Figura 5: ejercicio 3

- **fgrep**
- **fgrep** — Funciona de igual manera que grep, pero para ficheros (grep -f).
- **nologin** — Fichero especial que rechaza el inicio de sesión al usuario.

## 4. Mostrar los grupos de los usuarios con más de un grupo separado por espacios.

```
cut -f 1,4 -d: /etc/group | sed "s:/:/" | sed "s/,/ /" | awk '{print NF,$0}' | sort -nr | cut -d ' ' -f 2-
```



```
vagrant@vagrant:~$ cut -f 1,4 -d: /etc/group | sed "s:/:/" | sed "s/,/ /" | awk '{print NF,$0}' | sort -nr | cut -d ' ' -f 2-
adm syslog vagrant
sudo vagrant
sambashare vagrant
plugdev vagrant
lxd vagrant
lpadmin vagrant
dip vagrant
cdrom vagrant
vagrant@vagrant:~$
```

Figura 6: ejercicio 4

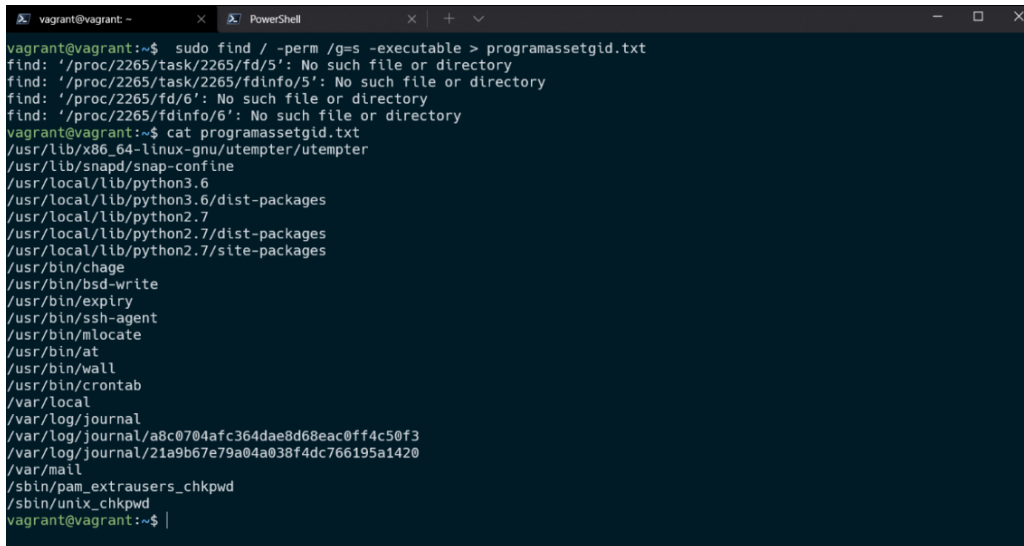
- **cut**
- **sed**
- **awk**
- **sort**

- **cut -f 1,4 -d: /etc/group** — Indicamos las líneas a imprimir (1, 4) y -d para indicar el delimitador de estas.
- **sed "s/:/ /" | sed "s/,/ /"** — Reemplazar toda aparición de ":" y "," por espacios.
- **awk '{print NF, \$0}'** — Imprime las líneas del fichero.
- **sort -nr** — Ordena numéricamente y decrecientemente.
- **cut -d ' ' -f 2-** — Muestra desde la segunda línea para evitar mostrar el contador de palabras de awk.

### 3. Procesos

1. Buscar todos los ficheros ejecutables con permisos setgid activado y guardarlo en un fichero.

`sudo find / -perm /g=s -executable > programasSetgid.txt`



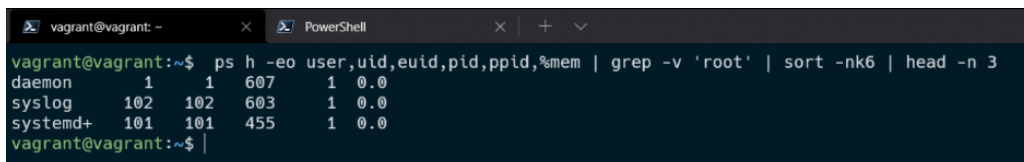
```
vagrant@vagrant:~$ sudo find / -perm /g=s -executable > programasSetgid.txt
find: '/proc/2265/task/2265/fd/5': No such file or directory
find: '/proc/2265/task/2265/fdinfo/5': No such file or directory
find: '/proc/2265/fd/6': No such file or directory
find: '/proc/2265/fdinfo/6': No such file or directory
vagrant@vagrant:~$ cat programasSetgid.txt
/usr/lib/x86_64-linux-gnu/utempter/utempter
/usr/lib/snapd/snap-confine
/usr/local/lib/python3.6
/usr/local/lib/python3.6/dist-packages
/usr/local/lib/python2.7
/usr/local/lib/python2.7/dist-packages
/usr/local/lib/python2.7/site-packages
/usr/bin/chage
/usr/bin/bsd-write
/usr/bin/expiry
/usr/bin/ssh-agent
/usr/bin/mlocate
/usr/bin/at
/usr/bin/wall
/usr/bin/crontab
/var/local
/var/log/journal
/var/log/journal/a8c0704afc364dae8d68eac0ff4c50f3
/var/log/journal/21a9b67e79a04a038f4dc766195a1420
/var/mail
/sbin/pam_extrausers_chkpwd
/sbin/unix_chkpwd
vagrant@vagrant:~$
```

Figura 7: ejercicio 5

- `-perm /g=s` — Filtrar por permisos, en este caso, setgid activado.
- `-executable` — Filtrar sólo por ficheros ejecutables.

2. Mostrar los 3 procesos no root del sistema que menos memoria están consumiendo.

`ps h -eo user,uid,euid,pid,ppid,%mem | grep -v 'root' | sort -nk6 | head -n 3`



```
vagrant@vagrant:~$ ps h -eo user,uid,euid,pid,ppid,%mem | grep -v 'root' | sort -nk6 | head -n 3
daemon      1      1    607      1  0.0
syslog     102    102    603      1  0.0
systemd+   101    101    455      1  0.0
vagrant@vagrant:~$
```

Figura 8: ejercicio 6

- `ps`
- `ps h -eo user,uid,euid,pid,ppid,%mem` — Muestra los procesos (-e) del formato (-o user) sin mostrar la cabecera (h) para evitar interpretar esa línea.
- `grep -v 'root'` — Mostrar toda línea que no coincida con el patrón.
- `sort -nk6` — Ordenar por la columna 6 numéricamente y ascendente.
- `head -n 3` — Mostrar las 3 primeras líneas.

3. Mostrar cada 5 segundos la carga media del sistema.

```
watch -n 5 " cut -d' ' -f 2 /proc/loadavg"
```

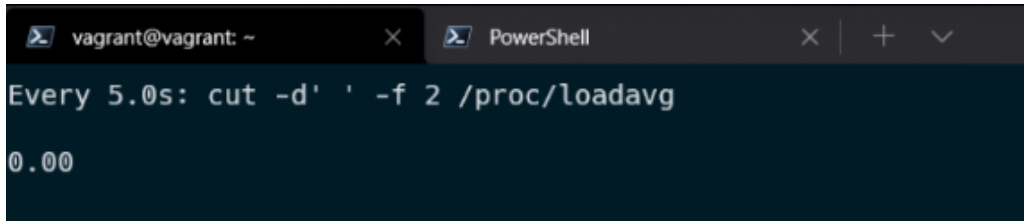
A screenshot of a terminal window with a dark background. The window has two tabs: 'vagrant@vagrant: ~' and 'PowerShell'. The terminal shows the command 'Every 5.0s: cut -d' ' -f 2 /proc/loadavg' followed by the output '0.00'.

Figura 9: ejercicio 7

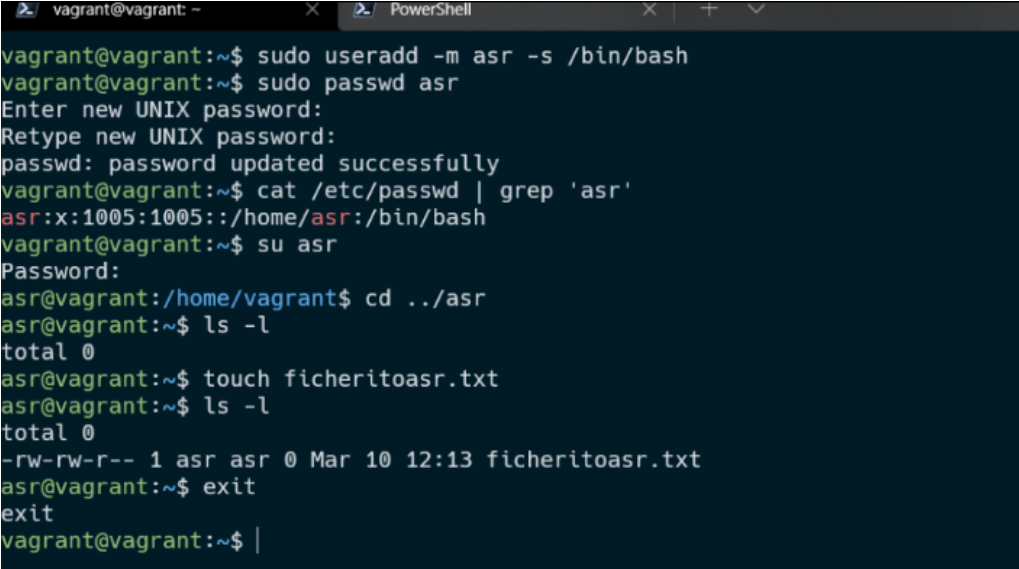
- **watch**
- **proc**
- **loadavg** — Contiene información sobre la carga media del sistema, mostrando 3 campos con la media cada 1, 5 y 15 minutos.
- **watch -n 5** — Ejecución del comando cada 5 segundos.
- **" cut -d' ' -f 2** — Selecciona el segundo campo, que es cada 5 minutos.



## 4. Gestión de Usuarios

1. Crear un nuevo usuario con directorio /home propio , con la shell bash y con contraseña.

```
sudo useradd -m asr -s /bin/bash
sudo passwd asr
```

A terminal window titled 'vagrant@vagrant: ~' with a PowerShell icon in the title bar. The terminal shows the following commands and output:

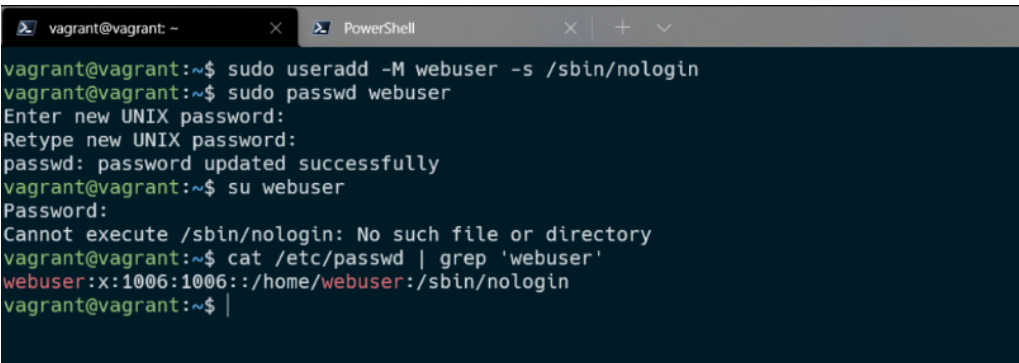
```
vagrant@vagrant:~$ sudo useradd -m asr -s /bin/bash
vagrant@vagrant:~$ sudo passwd asr
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
vagrant@vagrant:~$ cat /etc/passwd | grep 'asr'
asr:x:1005:1005:./home/asr:/bin/bash
vagrant@vagrant:~$ su asr
Password:
asr@vagrant:/home/vagrant$ cd ../asr
asr@vagrant:~$ ls -l
total 0
asr@vagrant:~$ touch ficheritoasr.txt
asr@vagrant:~$ ls -l
total 0
-rw-rw-r-- 1 asr asr 0 Mar 10 12:13 ficheritoasr.txt
asr@vagrant:~$ exit
exit
vagrant@vagrant:~$ |
```

Figura 10: ejercicio 8

- `useradd -m asr` — Crear un nuevo usuario con directorio home.
- `-s /bin/bash` — Asignamos al usuario la shell bash.
- `sudo passwd asr` — Permite asignar una contraseña al usuario.

2. Crear un nuevo usuario pero sin directorio casa y sin posibilidad de iniciar sesión.

```
sudo useradd -M webuser -s /sbin/nologin
sudo passwd webuser
```

A terminal window titled 'vagrant@vagrant: ~' with a PowerShell icon in the title bar. The terminal shows the following commands and output:

```
vagrant@vagrant:~$ sudo useradd -M webuser -s /sbin/nologin
vagrant@vagrant:~$ sudo passwd webuser
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
vagrant@vagrant:~$ su webuser
Password:
Cannot execute /sbin/nologin: No such file or directory
vagrant@vagrant:~$ cat /etc/passwd | grep 'webuser'
webuser:x:1006:1006:./home/webuser:/sbin/nologin
vagrant@vagrant:~$ |
```

Figura 11: ejercicio 9

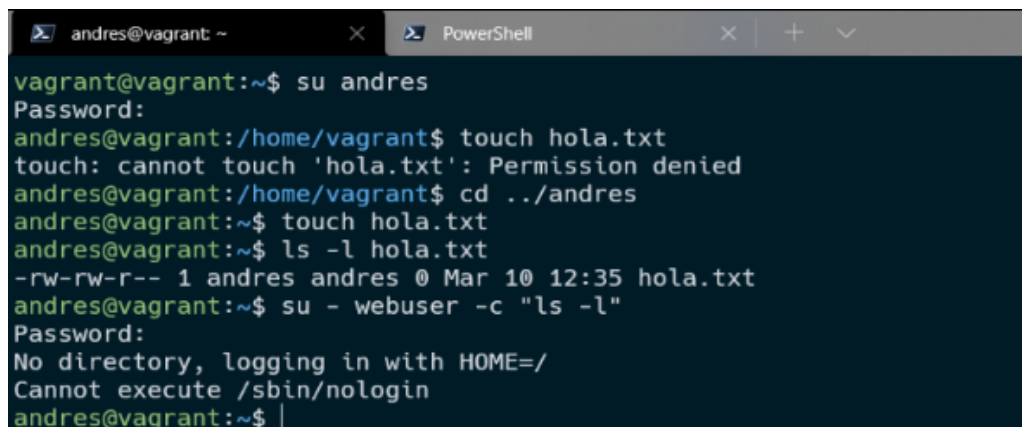
- `useradd -M webuser` — Crea al usuario sin directorio casa.
- `-s /sbin/nologin` — Le asignamos como shell nologin para que no pueda iniciar sesión.

3. Crear un nuevo usuario con directorio casa y con shell bash que tenga permisos de webuser.

```
sudo useradd -m andres -s /bin/bash
sudo passwd andres
sudo visudo
```

```
# User alias specification
andres ALL=(webuser) ALL
```

Figura 12: ejercicio 10.1



```
andres@vagrant: ~
PowerShell

vagrant@vagrant:~$ su andres
Password:
andres@vagrant:/home/vagrant$ touch hola.txt
touch: cannot touch 'hola.txt': Permission denied
andres@vagrant:/home/vagrant$ cd ../andres
andres@vagrant:~$ touch hola.txt
andres@vagrant:~$ ls -l hola.txt
-rw-rw-r-- 1 andres andres 0 Mar 10 12:35 hola.txt
andres@vagrant:~$ su - webuser -c "ls -l"
Password:
No directory, logging in with HOME=/
Cannot execute /sbin/nologin
andres@vagrant:~$ |
```

Figura 13: ejercicio 10.2

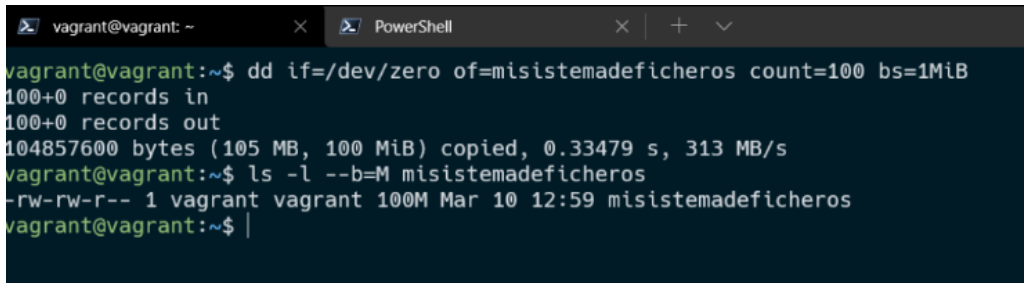
- **visudo** — Permite editar el fichero sudoers.
- **sudoers** — Fichero que contiene una lista de usuarios autorizados a usar el comando sudo (también permite que otros usuarios puedan ejecutar ciertos comandos como otro usuario, entre otras funcionalidades).
- **andres ALL=(webuser) ALL** — Sudoers utiliza EBNF, que es una forma de notación recursiva complicada de entender, pero muy potente y versátil.  
En nuestro caso, se traduciría de la siguiente forma:  
El usuario **andres** puede ejecutar **cualquier comando** únicamente como el usuario **webuser**.

Como se observa en la captura, el usuario andres no puede ejecutar ningún comando como **webuser** aunque tenga permisos, ya que **webuser** no tiene shell (tampoco puede hacer sudo porque no pertenece al grupo root).

## 5. Sistema de ficheros

1. Crea un espacio para albergar un sistema de ficheros de 100MiB

`dd if=/dev/zero of=misistemadeficheros count=100 bs=1MiB`



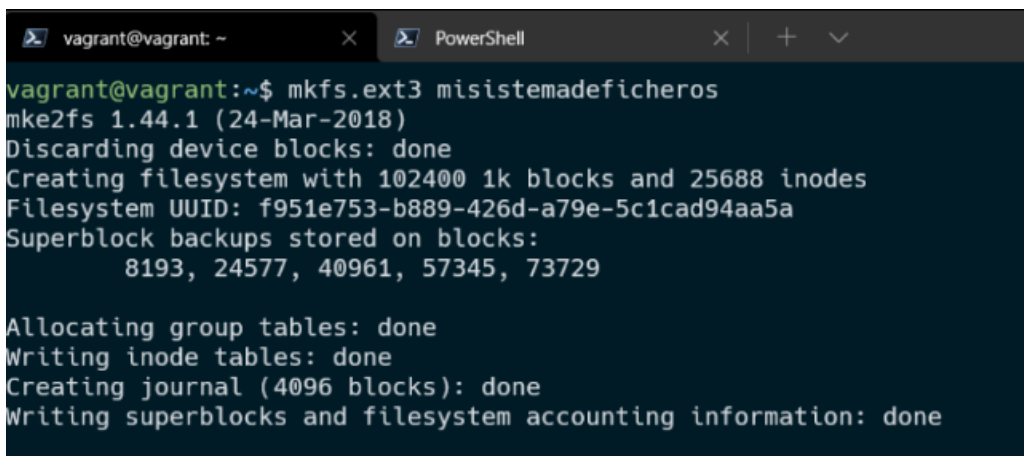
```
vagrant@vagrant: ~$ dd if=/dev/zero of=misistemadeficheros count=100 bs=1MiB
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.33479 s, 313 MB/s
vagrant@vagrant: ~$ ls -l --b=M misistemadeficheros
-rw-rw-r-- 1 vagrant vagrant 100M Mar 10 12:59 misistemadeficheros
vagrant@vagrant: ~$
```

Figura 14: ejercicio 11

- **dd** — Permite convertir o copiar un fichero.
- **if=/dev/zero** — desde
- **of=misistemadeficheros** — a
- **count=100** — Número de bloques.
- **bs=1MiB** — Tamaño de bloque.

2. Formatea el fichero anterior con ext3

`mkfs.ext3 misistemadeficheros`



```
vagrant@vagrant: ~$ mkfs.ext3 misistemadeficheros
mke2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 102400 1k blocks and 25688 inodes
Filesystem UUID: f951e753-b889-426d-a79e-5c1cad94aa5a
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

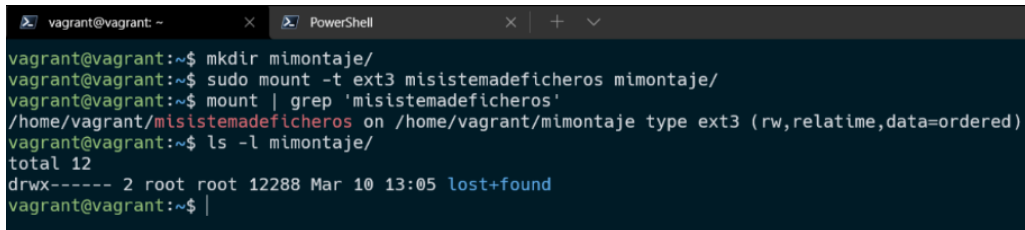
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Figura 15: ejercicio 12

- **mkfs** — Permite crear un sistema de ficheros con un formato determinado.
- **ext3** — Formato utilizado en el sistema de ficheros. Cabe destacar que cada formato tiene su propósito, en este caso, utilizamos **ext3** porque es rápido y permite expansión/reducción.

### 3. Montar el sistema de ficheros creado anteriormente.

`sudo mount -t ext3 misistemadeficheros mimontaje/`



```
vagrant@vagrant: ~  
vagrant@vagrant:~$ mkdir mimontaje/  
vagrant@vagrant:~$ sudo mount -t ext3 misistemadeficheros mimontaje/  
vagrant@vagrant:~$ mount | grep 'misistemadeficheros'  
/home/vagrant/misistemadeficheros on /home/vagrant/mimontaje type ext3 (rw,relatime,data=ordered)  
vagrant@vagrant:~$ ls -l mimontaje/  
total 12  
drwx----- 2 root root 12288 Mar 10 13:05 lost+found  
vagrant@vagrant:~$
```

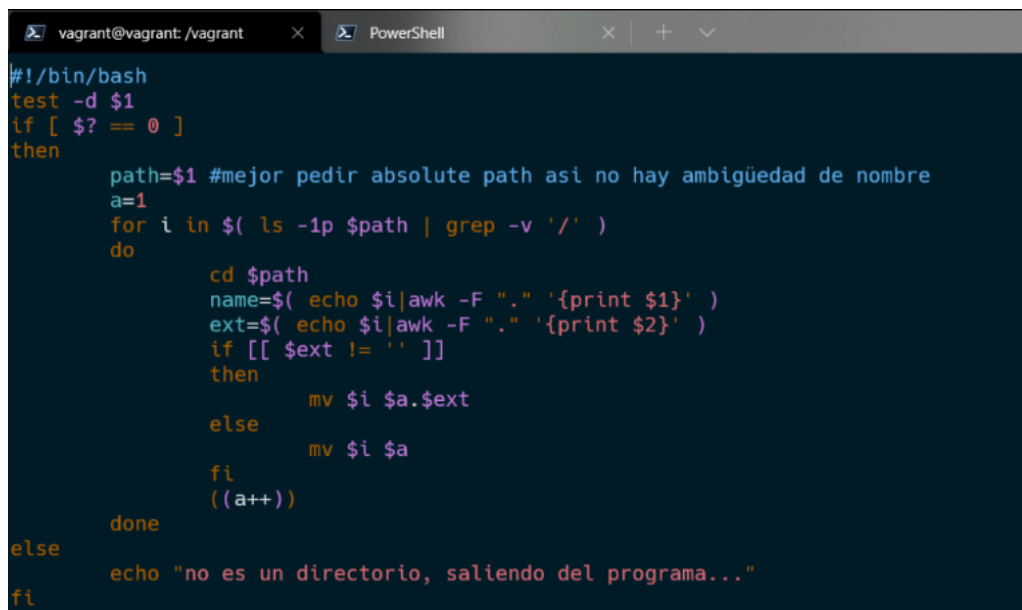
Figura 16: ejercicio 13

- **mount** — Inserta un sistema de ficheros en nuestro árbol del sistema.
- **-t ext3** — Indicamos el formato, no es necesario, pero es más visual.
- **misistemadeficheros** — sistema de ficheros a montar.
- **mimontaje/** — Directorio donde vamos a montar el sistema de ficheros.

**Nota:** ¡El directorio debe existir antes de ejecutar el comando!

## 6. Bash Scripting

### 1. Script 1



```
#!/bin/bash
test -d $1
if [ $? == 0 ]
then
    path=$1 #mejor pedir absolute path asi no hay ambigüedad de nombre
    a=1
    for i in $( ls -lp $path | grep -v '/' )
    do
        cd $path
        name=$( echo $i|awk -F "." '{print $1}' )
        ext=$( echo $i|awk -F "." '{print $2}' )
        if [[ $ext != '' ]]
        then
            mv $i $a.$ext
        else
            mv $i $a
        fi
        ((a++))
    done
else
    echo "no es un directorio, saliendo del programa..."
fi
```

Figura 17: ejercicio 14

- **mv** — Permite mover, renombrar ficheros o ambas.  
Se ha supuesto que el usuario introduce la ruta absoluta del fichero.

#### Funcionamiento del script:

Previamente a la ejecución de este, se ha comprobado que el directorio introducido por argumento existe, y es un directorio.

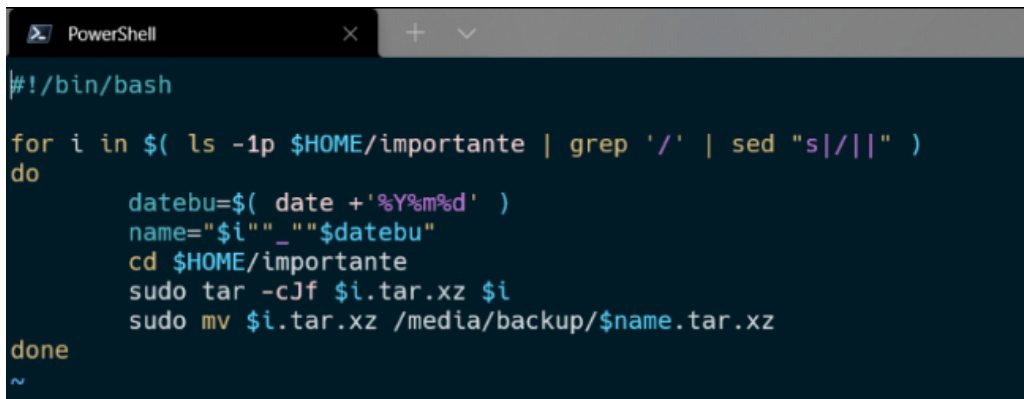
Se ha implementado un bucle for en el cual cada interacción contiene el nombre de un fichero (previamente se ha restringido la salida del comando **ls** con **grep**).

Se inicializa una variable numérica para asignárselas a los ficheros.

Dentro del bucle:

- Cambiamos de directorio para cambiar los nombres cómodamente.
- mediante **awk**, obtenemos las dos partes del fichero; el nombre y la extensión.
- Si el fichero tiene extensión (variable **ext** no vacía), entonces el nuevo nombre será un número secuencial + **ext**.
- Si no tiene extensión, simplemente cambiamos el nombre.
- número ++

## 2. Script 2



```
#!/bin/bash

for i in $( ls -lp $HOME/importante | grep '/' | sed "s|/||" )
do
    datebu=$( date +%Y%m%d' )
    name="$i"_"$datebu"
    cd $HOME/importante
    sudo tar -cJf $i.tar.xz $i
    sudo mv $i.tar.xz /media/backup/$name.tar.xz
done
```

Figura 18: ejercicio 15

- **tar** — Permite crear y manipular archivos tar que son colecciones de archivos. Básicamente nos permite comprimir archivos en uno solo.

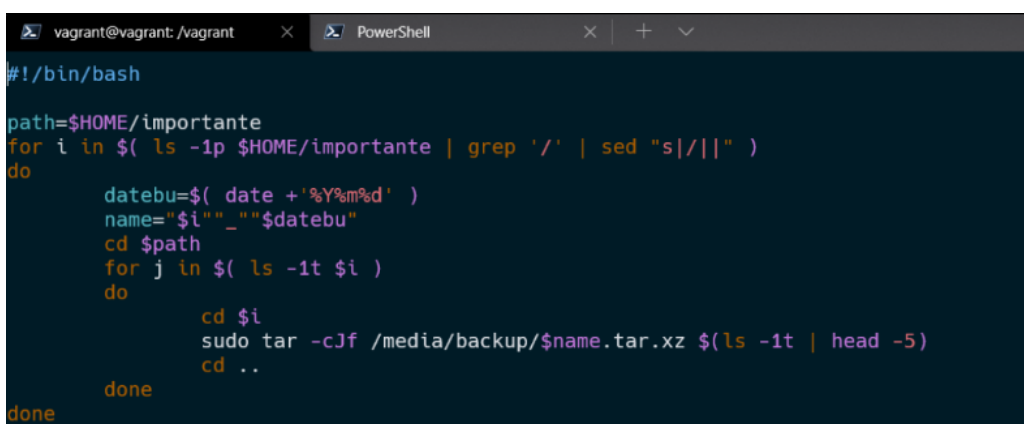
### Funcionamiento del script:

Se ha implementado un bucle for en el cuál cada iteración contiene el nombre de un directorio (previamente se ha restringido la salida del comando **ls** con **grep** y posteriormente se ha eliminado la barra del nombre del directorio usando **sed**).

Dentro del bucle:

- Se obtiene la fecha actual.
- se formatea el nombre del directorio para que sea **directorio\_yyyymmdd**.
- **sudo tar -cJf \$i.tar.xz \$i** — comprime el directorio **\$i** en el archivo **\$i.tar.xz**.
- finalmente, se mueven al directorio **backup** junto con un cambio de nombre final.

## 3. Script 3



```
#!/bin/bash

path=$HOME/importante
for i in $( ls -lp $HOME/importante | grep '/' | sed "s|/||" )
do
    datebu=$( date +%Y%m%d' )
    name="$i"_"$datebu"
    cd $path
    for j in $( ls -lt $i )
    do
        cd $i
        sudo tar -cJf /media/backup/$name.tar.xz $(ls -lt | head -5)
        cd ..
    done
done
```

Figura 19: ejercicio 16

### Funcionamiento del script:

Se ha implementado un bucle for en el cual cada iteración contiene el nombre de un directorio (previamente se ha restringido la salida del comando con **ls** con **grep** y posteriormente eliminado la barra del nombre del directorio usando **sed**).

Dentro del bucle:

- se obtiene la fecha actual.
- se formatea el nombre del directorio a **directorio\_yyyymmdd**.
- se cambia de directorio al directorio **~/importante**.

Implementamos otro bucle for en el que cada iteración vamos a ir comprimiendo los 5 primeros archivos del directorio actual.


Dentro del bucle:

- nos movemos al directorio **~/importante**.
- comprimimos los archivos de este usando el nombre formateado del directorio en **tar.xz**
- volvemos al directorio **~/importante** para seleccionar otro directorio.

#### Explicación de los comandos:

- **ls -lt \$i** — Listamos el directorio \$i, ordenándolos por fecha de creación (-t).
- **sudo tar -cJf /media/backup/\$name.tar.xz \$(ls -lt) | head -5** — Comprime los 5 primeros archivos en el archivo **\$name.tar.xz**.

#### 4. Ejecuta automáticamente todos los días a las 9:30 el script backup.sh



```
30 09 * * * sudo /vagrant/backup.sh
```

Figura 20: ejercicio 17

- **cron** — Es un demonio que ejecuta comandos de forma planificada.
- **crontab** — Son ficheros (tablas) de configuración para ejecutar órdenes cron.
- **sudo crontab -u root -e** — Abrir el fichero crontab del usuario root para modificarlo (-e).
- **30 09 \* \* \* sudo sh /vagrant/backup.sh** — Minutos, horas, todos los días ejecuta el siguiente comando.