

PRÁCTICA 2 DE SISTEMAS OPERATIVOS

TEMA: Preparación del ambiente de prácticas.

Nombre: Fernando Huilca

Carrera: Ingeniería de Software

Grupo: GR1SW

Fecha: 17 / 08 / 2024

Índice de Contenidos

| | |
|---|----|
| 1. OBJETIVOS | 2 |
| 2. INFORME | 2 |
| 3. CONCLUSIONES Y RECOMENDACIONES | 23 |
| 4. BIBLIOGRAFÍA | 23 |

Índice de Imágenes

| | |
|---|----|
| Ilustración 1 fecha y hora de la computadora..... | 2 |
| Ilustración 2 calendario septiembre 1983..... | 3 |
| Ilustración 3 Usuarios del sistema | 3 |
| Ilustración 4 Comando para limpiar el terminal | 4 |
| Ilustración 5 Contenido de las variables de entorno | 4 |
| Ilustración 6 Mensaje de texto en la terminal comando write | 5 |
| Ilustración 7 Historial de comandos..... | 5 |
| Ilustración 8 Fichero bash_history con uso de cat | 6 |
| Ilustración 9 Comando !-1 | 6 |
| Ilustración 10 Comando !! | 7 |
| Ilustración 11 Comando !7 | 7 |
| Ilustración 12 Figura realizada con el comando Vi..... | 8 |
| Ilustración 13 Ejecución del comando :1,\$s /DE/SA/g | 9 |
| Ilustración 14 Comando Pstree -p | 10 |
| Ilustración 15 Comando pstree -h..... | 11 |
| Ilustración 16 Ejecucion del comando yes | 13 |
| Ilustración 17 Uso de grep yes | 13 |
| Ilustración 18 Directorio para observar PCB | 17 |
| Ilustración 19 Ver procesos con /proc/PID/status | 18 |
| Ilustración 20 Creación del scrip para imprimir numeros | 19 |
| Ilustración 21 Ejecucion del Script en segundo plano..... | 20 |
| Ilustración 22 Confirmación de permiso para ejecución | 20 |
| Ilustración 23 Lo que sucede en segundo plano desde el PIC..... | 21 |
| Ilustración 24 Ejecución del script en primer plano | 22 |

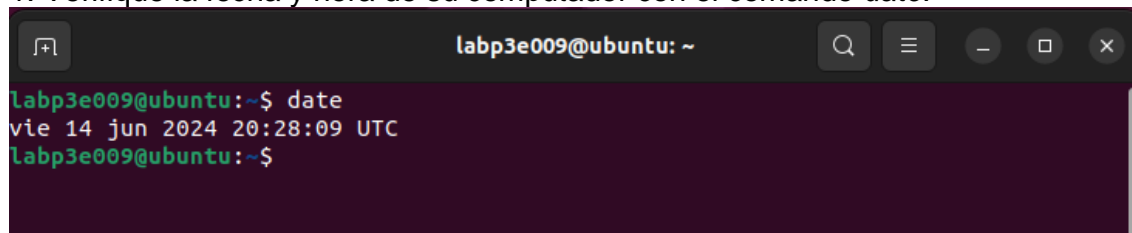
1. OBJETIVOS

- 1.1 Familiarizarse con el sistema operativo Linux, la consola y sus instrucciones.
- 1.2 Ser capaces de entender la diferencia entre los comandos en Linux y Windows.

2. INFORME

3.6. Mediante el uso de los comandos mencionados, realice las siguientes tareas (No mediante la interfaz gráfica).

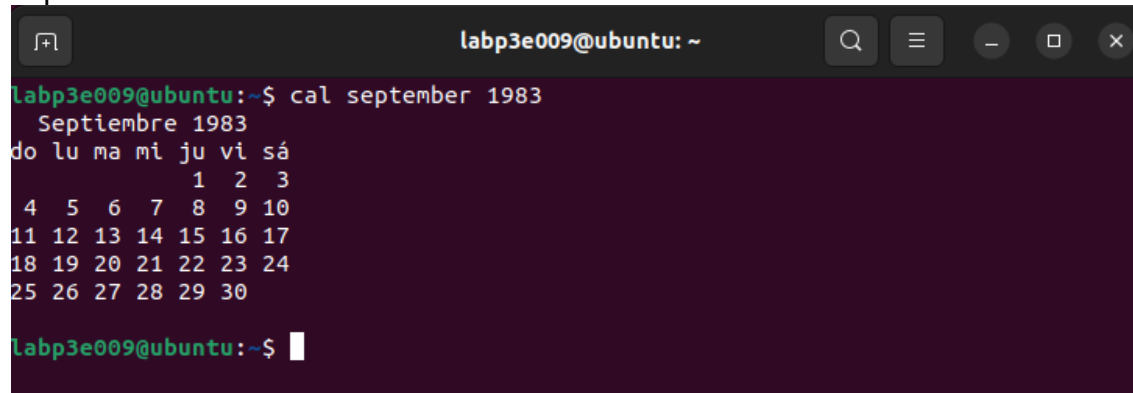
- 1. Verifique la fecha y hora de su computador con el comando `date`.



```
labp3e009@ubuntu: ~  
labp3e009@ubuntu:~$ date  
vie 14 jun 2024 20:28:09 UTC  
labp3e009@ubuntu:~$
```

Ilustración 1 fecha y hora de la computadora

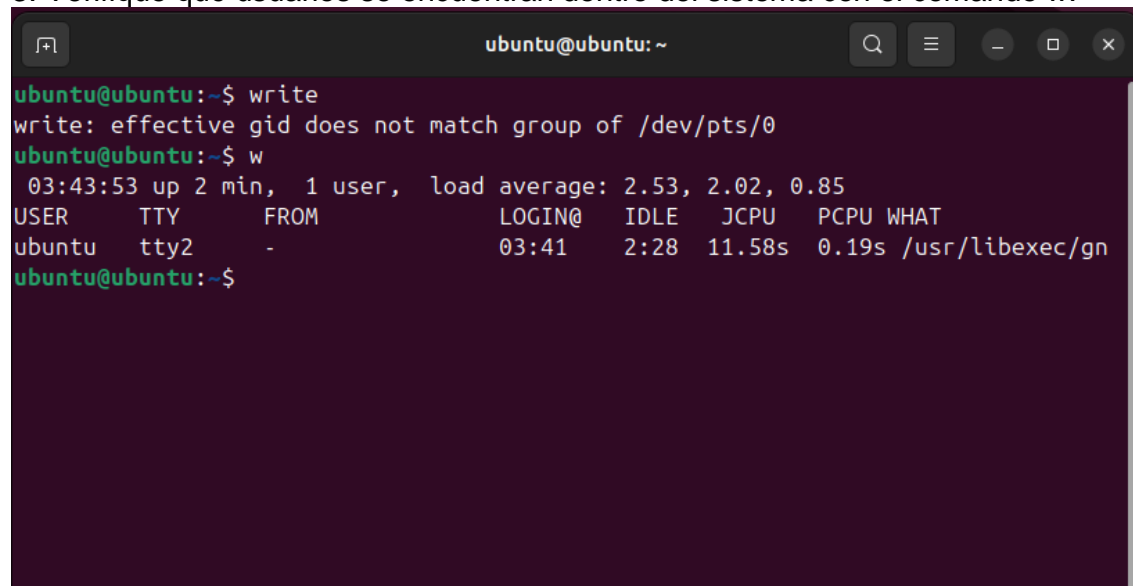
2. Muestre en pantalla el calendario con el comando `cal`. Luego muestre el calendario de septiembre de 1983.



```
labp3e009@ubuntu: ~  
labp3e009@ubuntu:~$ cal september 1983  
Septiembre 1983  
do lu ma mi ju vi sá  
          1  2  3  
 4  5  6  7  8  9 10  
11 12 13 14 15 16 17  
18 19 20 21 22 23 24  
25 26 27 28 29 30  
labp3e009@ubuntu:~$
```

Ilustración 2 calendario septiembre 1983

3. Verifique qué usuarios se encuentran dentro del sistema con el comando `w`.



```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ write  
write: effective gid does not match group of /dev/pts/0  
ubuntu@ubuntu:~$ w  
03:43:53 up 2 min,  1 user,  load average: 2.53, 2.02, 0.85  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT  
ubuntu    tty2      -                03:41    2:28  11.58s  0.19s /usr/libexec/gn  
ubuntu@ubuntu:~$
```

Ilustración 3 Usuarios del sistema

4. Limpie la pantalla de su terminal con el comando `clear`.

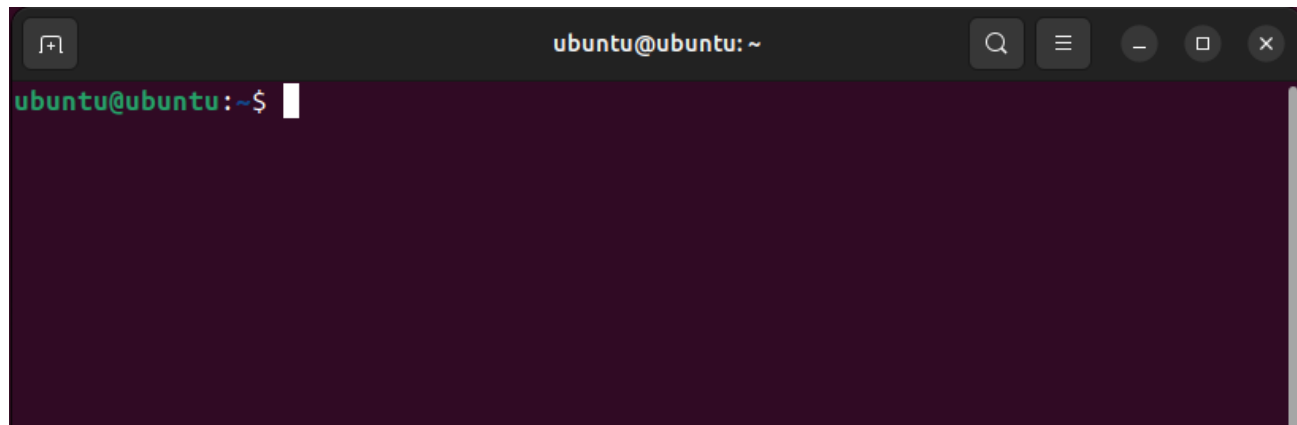


Ilustración 4 Comando para limpiar el terminal

5. Muestre el contenido de las variables de entorno de la tabla debajo con el comando `echo` (respeta el uso de mayúsculas y minúsculas).

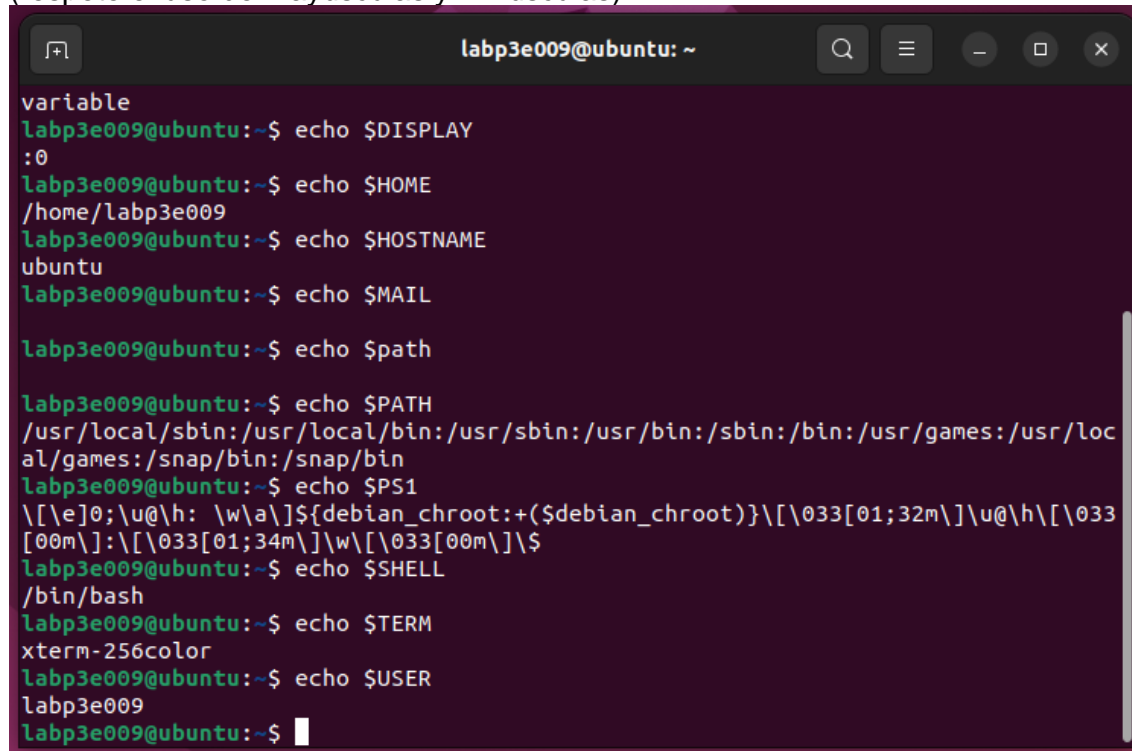


Ilustración 5 Contenido de las variables de entorno

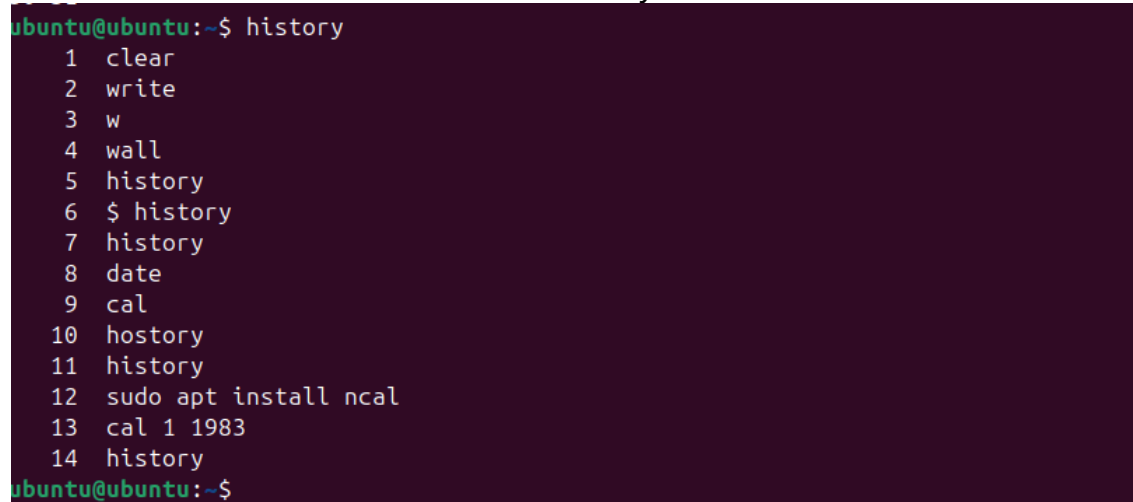
6. Muestre un mensaje de texto en la terminal de un usuario con el comando *write*.



```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ write  
write: effective gid does not match group of /dev/pts/0  
ubuntu@ubuntu:~$
```

Ilustración 6 Mensaje de texto en la terminal comando *write*

8. Verificar el historial de comandos con *history*.



```
ubuntu@ubuntu:~$ history  
1 clear  
2 write  
3 w  
4 wall  
5 history  
6 $ history  
7 history  
8 date  
9 cal  
10 history  
11 history  
12 sudo apt install ncal  
13 cal 1 1983  
14 history  
ubuntu@ubuntu:~$
```

Ilustración 7 Historial de comandos

9. Verificar el contenido del fichero *~/.bash_history* con el comando *cat*. Verificar que se encuentra logueado con el usuario root. (Para utilizar el símbolo *~* pruebe con Alt Gr + 4 o, si posee teclado numérico, también puede utilizar Alt Gr + 9 desactivando previamente el

teclado numérico).

```
date
w
clear
history
cat ~/.bash_history
sudo -l
cat ~/.bash_history
echo $DISPLAY
echo $HOSTNAME
echo $PATH
history
cat ~/.bash_history
echo $SHELL
cat ~/.bash_history
history -a
```

Ilustración 8 Fichero bash_history con uso de cat

10. Verificar el resultado obtenido con la ejecución de los siguientes comandos. Explicar su uso.

\$!-1

\$!! 10

\$!7

El comando !-1 ejecuta el último comando que se ejecutó en la terminal.

```
cat ~/.bash_history
date
w
clear
history
cat ~/.bash_history
sudo -l
cat ~/.bash_history
echo $DISPLAY
echo $HOSTNAME
echo $PATH
history
cat ~/.bash_history
echo $SHELL
cat ~/.bash_history
history -a
cat ~/.bash_history
$cat ~/.bash_history
sudo update
sudo apt update
apt list
```

Ilustración 9 Comando !-1

El comando `!!` también ejecuta el último comando que se ejecutó en la terminal.

```
cat ~/.bash_history
date
w
clear
history
cat ~/.bash_history
sudo -l
cat ~/.bash_history
echo $DISPLAY
echo $HOSTNAME
echo $PATH
history
cat ~/.bash_history
echo $SHELL
cat ~/.bash_history
history -a
cat ~/.bash_history
$cat ~/.bash_history
sudo update
sudo apt update
apt list
```

Ilustración 10 Comando !!

El comando `!7` ejecuta el séptimo comando en el historial de la sesión actual.

```
cat ~/.bash_history
date
w
clear
history
cat ~/.bash_history
sudo -l
cat ~/.bash_history
echo $DISPLAY
echo $HOSTNAME
echo $PATH
history
cat ~/.bash_history
echo $SHELL
cat ~/.bash_history
history -a
cat ~/.bash_history
$cat ~/.bash_history
sudo update
sudo apt update
apt list
apt list --upgrade
clear
```

Ilustración 11 Comando !7

3.11. Realizar la siguiente figura con el comando vi dentro de un fichero que tendrá su nombre:

```
ubuntu@ubuntu:~$ vi Fernando-huilca
ubuntu@ubuntu:~$ cat Fernando-huilca
( __ ^__ )      ( __ ^__ )      ( __ ^__ )
|_____|-----|_____|-----|_____|
|///|          |///|          |///|
|///|  -----oo0( _ )0oo----- |///|
|///|-----|_____|-----|_____|
|///|  DEBER  |         |         |  SO  |///|
|///|  DE     |         |         |  LINUX  |///|
|///|         |         |         |         |///|
(   )         |         |         |         |
ubuntu@ubuntu:~$
```

Ilustración 12 Figura realizada con el comando Vi

3.12 Ejecutar el siguiente comando, verificar qué sucede y explicar de manera detallada su uso. El comando inicia con el símbolo “dos puntos” (:) para acceder al Modo Línea. Utilizar todos los demás símbolos indicados dentro del comando como la “coma” (,) y el símbolo de “dólar” (\$).

:1,\$s /DE/SA/g

El comando :1,\$s /DE/SA/g en vi/Vim realiza una sustitución global en todo el archivo desde la primera hasta la última línea, reemplazando todas las instancias de "DE" por "SA".

□ **Comando y Modo Línea (:):** El comando comienza con : para indicar que se está ejecutando en el Modo Línea de vi/Vim. Este modo permite ejecutar comandos específicos.

□ **Rango (1,\$):** 1,\$ especifica que el comando se aplicará desde la primera línea hasta la última línea del archivo.

□ **Sustitución (s /DE/SA/g):**

- s: Indica que se va a realizar una sustitución.
- /DE/SA/: Es la sintaxis para la sustitución. En este caso, se buscará la cadena "DE" y se reemplazará por "SA".

- g: Es una bandera opcional que significa "global", lo que indica que todas las apariciones de "DE" en cada línea serán reemplazadas, no solo la primera.

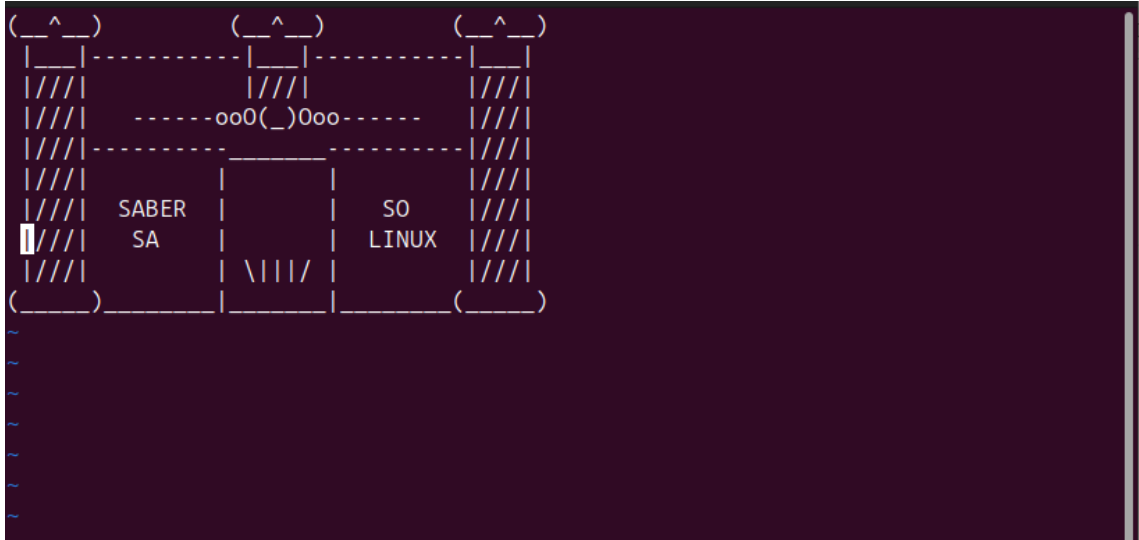


Ilustración 13 Ejecución del comando :1,\$s /DE/SA/g

3.16 Mediante el comando *ps tree* determinar el orden jerárquico de los procesos que se están ejecutando actualmente en su terminal bash. Ventaja: El árbol muestra los procesos en una relación padre-hijo.

Utilice también `pstree -p` y `pstree -h`. ¿Cuál es la diferencia?

Pstree -p

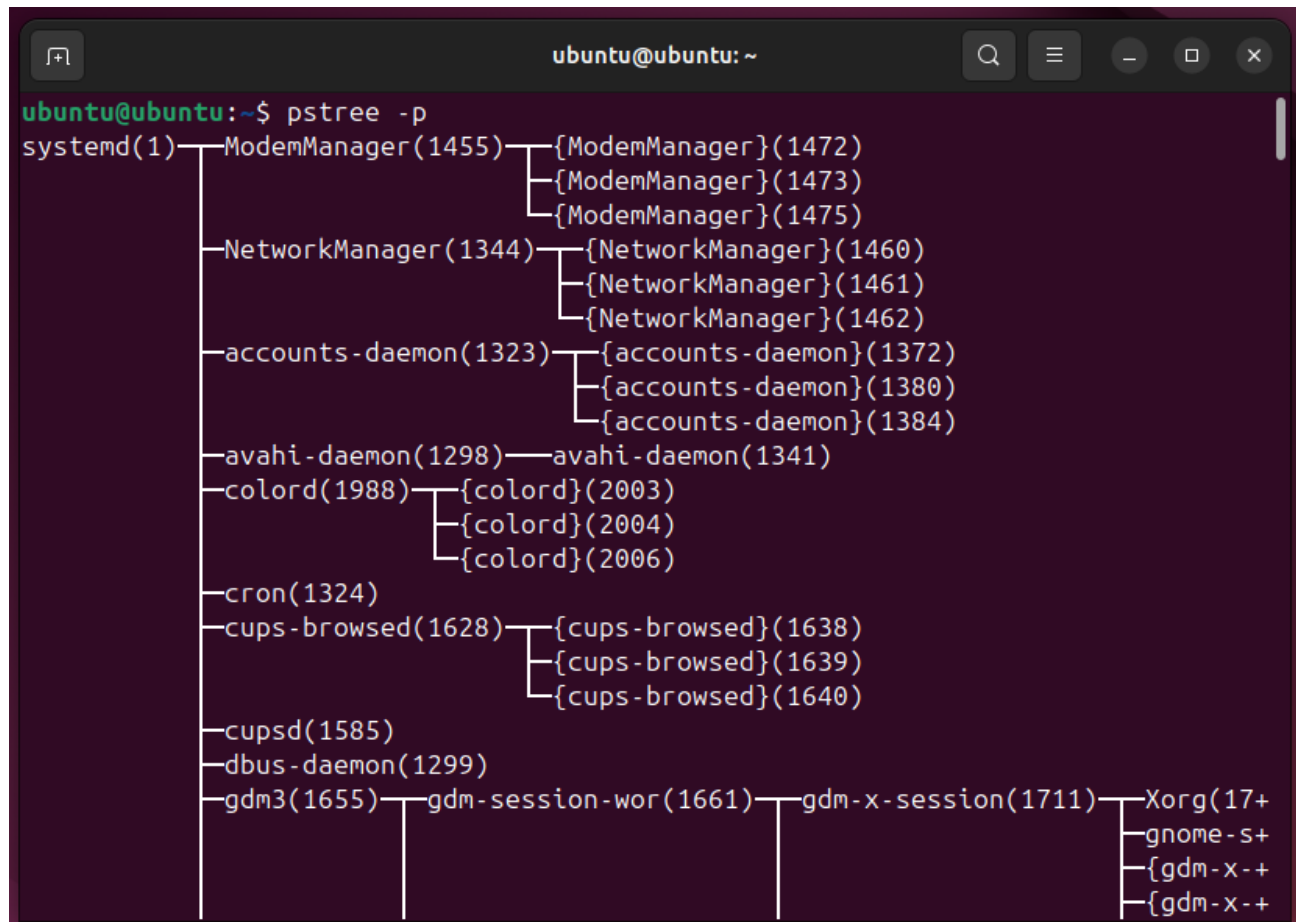


Ilustración 14 Comando Pstree -p

Pstree -h

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ pstree -h  
systemd├─ModemManager─3*[{ModemManager}]  
      └─NetworkManager─3*[{NetworkManager}]  
      └─accounts-daemon─3*[{accounts-daemon}]  
      └─avahi-daemon─avahi-daemon  
      └─colord─3*[{colord}]  
      └─cron  
      └─cups-browsed─3*[{cups-browsed}]  
      └─cupsd  
      └─dbus-daemon  
      └─gdm3├─gdm-session-wor├─gdm-x-session├─Xorg─{Xorg}  
          │                  │               │   │   │  
          │                  │               │   │   └─gnome-session-b─3*[{gnome-+  
          │                  │               │   │   └─3*[{gdm-x-session}]  
          │                  └─3*[{gdm-session-wor}]  
          └─3*[{gdm3}]  
      └─gnome-remote-de─3*[{gnome-remote-de}]  
      └─2*[{kerneloops}]  
      └─polkitd─3*[{polkitd}]  
      └─power-profiles-─3*[{power-profiles-}]  
      └─rsyslogd─3*[{rsyslogd}]  
      └─rtkit-daemon─2*[{rtkit-daemon}]  
      └─snapd─10*[{snapd}]  
      └─subiquity-serve─python3.10─4*[{python3.10}]  
      └─switcheroo-cont─3*[{switcheroo-cont}]
```

Ilustración 15 Comando `pstree -h`

Diferencia entre `pstree -p` y `pstree -h`

- `pstree -p`: Añade información sobre los PIDs de los procesos, lo que puede ser útil para identificar procesos específicos y sus relaciones.
- `pstree -h`: Resalta el proceso actual y sus ancestros, facilitando la identificación del proceso que estás ejecutando y cómo se relaciona con otros procesos en el sistema

Pstree

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ pstree  
systemd--ModemManager--3*[{ModemManager}]  
--NetworkManager--3*[{NetworkManager}]  
--accounts-daemon--3*[{accounts-daemon}]  
--avahi-daemon--avahi-daemon  
--colord--3*[{colord}]  
--cron  
--cups-browsed--3*[{cups-browsed}]  
--cupsd  
--dbus-daemon  
--gdm3--gdm-session-wor--gdm-x-session--Xorg--{Xorg}  
--gdm-session-b--3*[{gnome-+  
--3*[{gdm-x-session}]  
--3*[{gdm-session-wor}]  
--3*[{gdm3}]  
--gnome-remote-de--3*[{gnome-remote-de}]  
--2*[kerneloops]  
--polkitd--3*[{polkitd}]  
--power-profiles--3*[{power-profiles-}]  
--rsyslogd--3*[{rsyslogd}]  
--rtkit-daemon--2*[{rtkit-daemon}]  
--snapd--10*[{snapd}]  
--subiquity-serve--python3.10--4*[{python3.10}]  
--switcheroo-cont--3*[{switcheroo-cont}]
```

3.17 Ejecutar el comando `yes` en una terminal y mediante el comando `ps` (investigar las opciones) determinar los PID y PPID del proceso asociado.

Hay varias opciones útiles en `ps`:

`ps aux`: Muestra información de todos los procesos en el sistema.

`ps -ef`: Otra forma de mostrar todos los procesos con una estructura de árbol.

`ps -o pid,ppid,cmd -C yes`: Muestra los PID, PPID y el comando de los procesos específicos con el nombre `yes`.

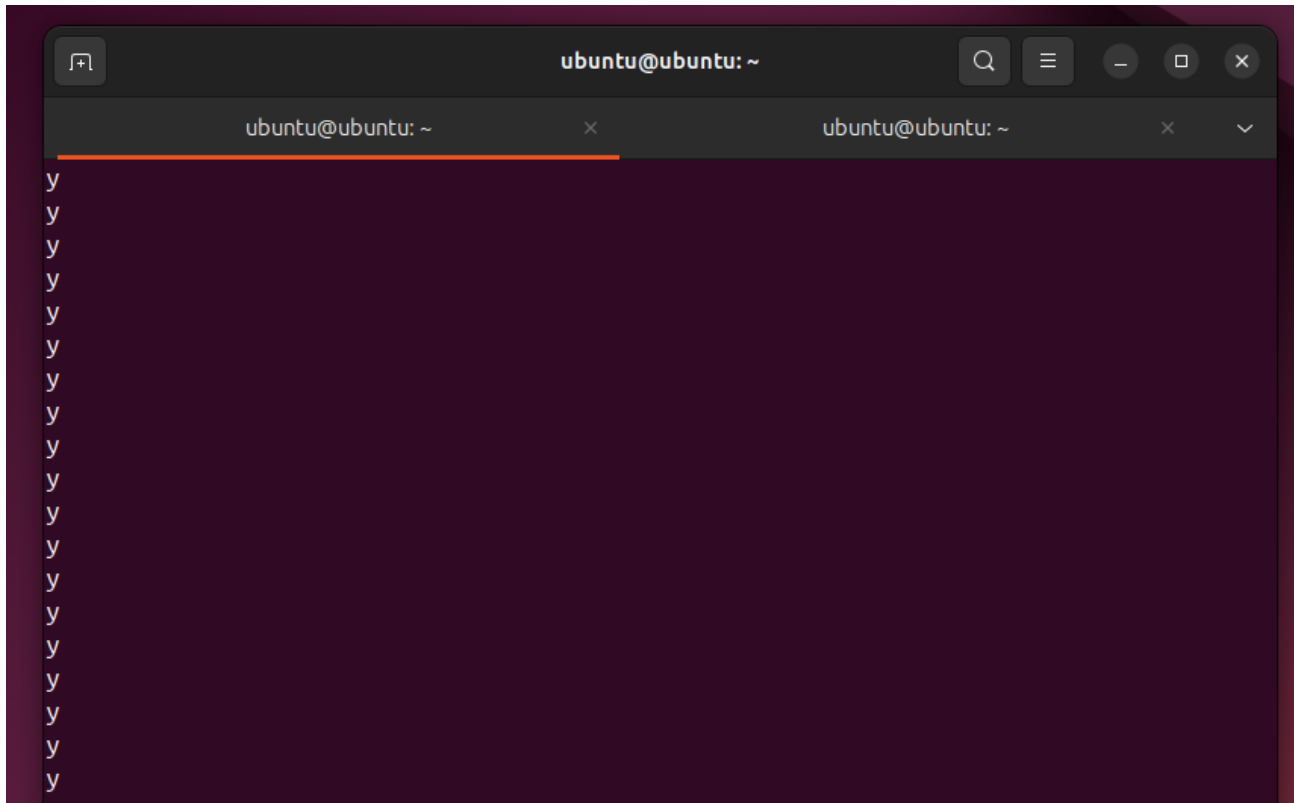


Ilustración 16 Ejecucion del comando yes

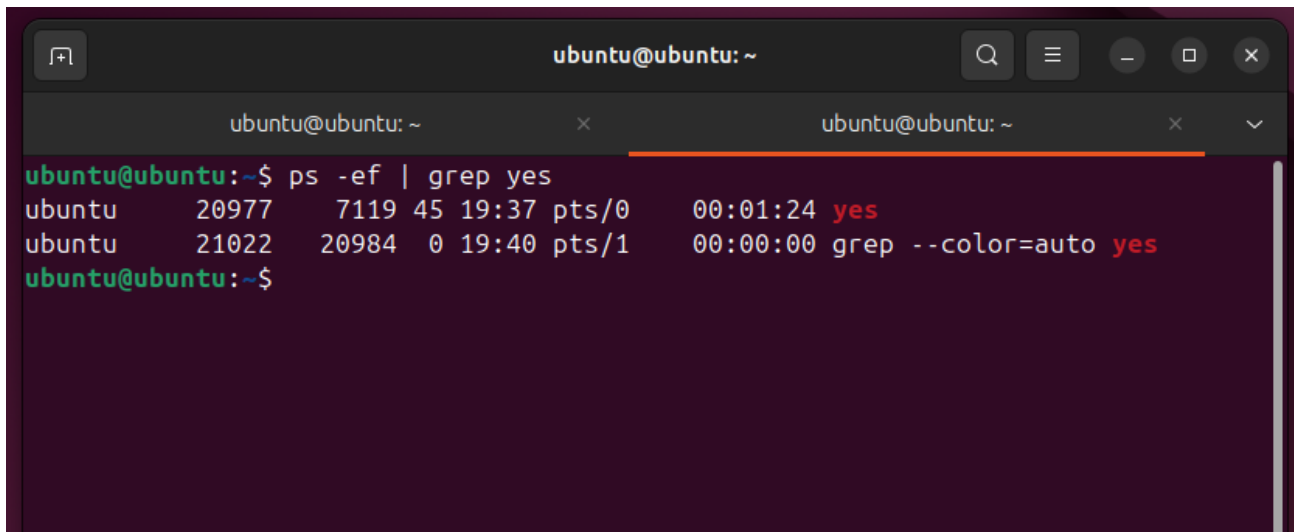


Ilustración 17 Uso de grep yes

3.18 Mediante el comando **ps** (investigar las opciones) determinar para el proceso asociado al comando **yes**:

- F – Indicadores asociados con el proceso.
- S – Estado del proceso.
- UID – Identificación del usuario (ID) propietario del proceso.
- CPU – Utilización del proceso.
- PRI – Prioridad del proceso.
- WCHAN – El suceso por el cual el proceso está esperando.
- TTY – El Terminal que controla el proceso
- TIME – Tiempo de ejecución acumulativa del proceso.

Procesos Linux **ps**

El comando **ps** es utilizado para mostrar de forma enumerada los procesos que se están ejecutando.

Modificadores (Opciones):

- A - muestra todos los procesos del sistema.
- u <nombre usuario> - El comando muestra los procesos del usuario, además puede mostrar los procesos de un usuario determinado.
- F - Indicadores asociados al proceso.
- s - Estado del proceso.
- ef - Los procesos con el ID de proceso y predecesor.
- l - permite contar los procesos.

Elementos de consola:

- UID – Identificación del usuario (ID) propietario del proceso.
- CPU – Utilización del proceso.
- PRI – Prioridad del proceso.
- WCHAN – El suceso por el cual el proceso está esperando.
- TTY – El Terminal que controla el proceso
- TIME – Tiempo de ejecución acumulativa del proceso.

```
fernando-huilca@VirtualBox:~$ yes
```

Figura. - Ejecución comando yes.

```
y
y
y
y
y
y
y
y
y
y
^C
fernando-huilca@VirtualBox:~$
```

Figura. - Repetición del carácter “y”, asociado al comando yes.

```
fernando-huilca@VirtualBox:~$ ps -A
PID TTY          TIME CMD
  1 ?            00:00:05 systemd
  2 ?            00:00:00 kthreadd
  3 ?            00:00:00 pool_workqueue_release
  4 ?            00:00:00 kworker/R-rcu_g
  5 ?            00:00:00 kworker/R-rcu_p
  6 ?            00:00:00 kworker/R-slub_
  7 ?            00:00:00 kworker/R-netns
  8 ?            00:00:00 kworker/0:0-events
  9 ?            00:00:00 kworker/0:1-events
 10 ?            00:00:00 kworker/0:0H-events_highpri
 11 ?            00:00:00 kworker/u4:0-events_unbound
 12 ?            00:00:00 kworker/R-mm_pe
 13 ?            00:00:00 rcu_tasks_kthread
 14 ?            00:00:00 rcu_tasks_rude_kthread
 15 ?            00:00:00 rcu_tasks_trace_kthread
 16 ?            00:00:00 ksoftirqd/0
```

Figura. – Ejecución comando para impresión de todos los procesos del sistema.

3.19. – Utilice el comando top para mostrar todos los procesos, usuarios a los que pertenecen los procesos, y la serie de recursos que ocupan en memoria los procesos. Usar también el comando top -u <Usuario>.

```
fernando-huilca@VirtualBox: ~
top - 19:02:07 up 4 min,  1 user,  load average: 0,82, 1,55, 0,77
Tareas: 192 total,  1 ejecutar, 191 hibernar,  0 detener,  0 zombie
%Cpu(s):  1,0 us,  1,2 sy,  0,0 ni, 97,8 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem : 3916,1 total, 2128,6 libre, 1024,4 usado, 1015,6 búf/caché
MiB Intercambio: 2621,0 total, 2621,0 libre,  0,0 usado. 2891,7 dispon
```

Figura. – Ejecución comando top

La utilización del comando *top*, presenta información detallada acerca de los procesos que realiza cada uno de los usuarios en tiempo real por lo que se va actualizando constantemente hasta que el usuario con ayuda de un teclado ingrese *Ctrl + C*, además esto se puede filtrar por usuario.

```
fernando-huilca@VirtualBox:~$ top -u fernando-huilca
```

Figura. – Comando ingresado en consola.

```
top - 19:04:02 up 6 min, 1 user, load average: 0,15, 1,07, 0,69
Tareas: 186 total, 1 ejecutar, 185 hibernar, 0 detener, 0 zombie
%Cpu(s): 3,6 us, 3,4 sy, 0,0 ni, 93,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 3916,1 total, 2128,8 libre, 1023,3 usado, 1016,6 búf/caché
MiB Intercambio: 2621,0 total, 2621,0 libre, 0,0 usado. 2892,8 dispon
```

Figura. – Resumen general de los procesos en ejecución y uso del hardware.

| PID | USUARIO | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | HORA+ | ORDEN |
|------|----------|----|-----|---------|--------|--------|---|------|------|---------|----------|
| 2531 | fernand+ | 20 | 0 | 3960932 | 359068 | 139216 | S | 12,0 | 9,0 | 0:24.52 | gnome-s+ |
| 3525 | fernand+ | 20 | 0 | 552632 | 54404 | 43372 | S | 2,7 | 1,4 | 0:02.95 | gnome-t+ |
| 3668 | fernand+ | 20 | 0 | 12220 | 6016 | 3840 | R | 0,8 | 0,2 | 0:00.09 | top |
| 2645 | fernand+ | 20 | 0 | 386648 | 12504 | 7296 | S | 0,3 | 0,3 | 0:00.67 | ibus-da+ |
| 3404 | fernand+ | 20 | 0 | 2804640 | 62796 | 47824 | S | 0,3 | 1,6 | 0:01.07 | gjs |
| 2186 | fernand+ | 20 | 0 | 21100 | 12420 | 9600 | S | 0,0 | 0,3 | 0:16.75 | systemd |
| 2189 | fernand+ | 20 | 0 | 21456 | 3576 | 1792 | S | 0,0 | 0,1 | 0:00.00 | (sd-pam) |
| 2207 | fernand+ | 9 | -11 | 112668 | 14124 | 8960 | S | 0,0 | 0,4 | 0:00.19 | pipewire |
| 2209 | fernand+ | 20 | 0 | 95220 | 6016 | 5248 | S | 0,0 | 0,2 | 0:00.02 | pipewire |
| 2217 | fernand+ | 9 | -11 | 404664 | 19200 | 14208 | S | 0,0 | 0,5 | 0:00.21 | wireplu+ |
| 2218 | fernand+ | 9 | -11 | 112804 | 12704 | 9760 | S | 0,0 | 0,3 | 0:00.08 | pipewir+ |
| 2219 | fernand+ | 20 | 0 | 10744 | 6528 | 4608 | S | 0,0 | 0,2 | 0:00.79 | dbus-da+ |
| 2220 | fernand+ | 20 | 0 | 314216 | 10240 | 9216 | S | 0,0 | 0,3 | 0:00.10 | gnome-k+ |
| 2250 | fernand+ | 20 | 0 | 682844 | 7936 | 7168 | S | 0,0 | 0,2 | 0:00.07 | xdg-doc+ |

Figura. – Procesos en ejecución asociados al usuario Fernando-Huilca.

3.24 Investigue en qué directorio se puede observar el PCB de un proceso en Linux.

En sistemas Linux, el PCB (Process Control Block) de un proceso se puede observar en el directorio `/proc`, que es un sistema de archivos especial que presenta información sobre el sistema y los procesos. Cada proceso tiene un subdirectorio identificado por su PID, como `/proc/<PID>`. Dentro de este subdirectorio, se pueden encontrar archivos como `stat` y `status` que contienen los detalles del proceso. Por ejemplo, en `/proc/<PID>/status`, se puede encontrar información detallada sobre el proceso.

Para saber cuáles son esos procesos podemos usar el comando `top`:


```

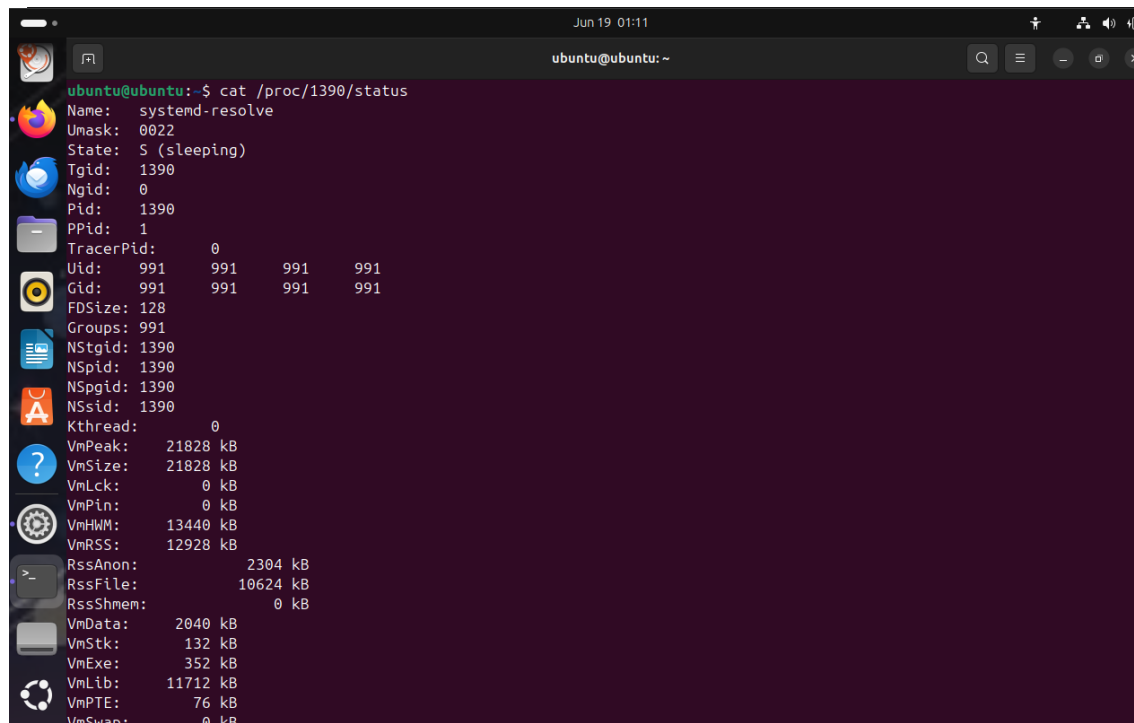
Jun 19 00:37
ubuntu@ubuntu: ~
top - 00:37:30 up 1:34, 1 user, load average: 0.01, 0.18, 0.50
Tasks: 262 total, 1 running, 261 sleeping, 0 stopped, 0 zombie
%Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem : 3316.9 total, 314.9 free, 1724.6 used, 1948.0 buff/cache
MiB Swap:  0.0 total,  0.0 free,  0.0 used, 1592.2 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 2090 ubuntu    20   0 344184 99580 65116 S   3.0   2.9   2:10.79 Xorg
 2618 ubuntu    20   0 4953600 364028 145380 S   2.4  10.7   2:29.54 gnome-shell
10072 ubuntu    20   0 851156 54888 42948 S   1.2   1.6   0:01.44 gnome-terminal-
1389 systemd+  20   0 17556 7424 6656 S   0.3   0.2   0:02.85 systemd-oomd
 5048 root       20   0      0      0      0 I   0.3   0.0   0:01.50 kworker/4:1-events
      1 root       20   0 23516 14316 9580 S   0.0   0.4   0:08.62 systemd
      2 root       20   0      0      0      0 S   0.0   0.0   0:00.11 kthreadd
      3 root       20   0      0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
      5 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
      6 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
      7 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
     11 root       20   0      0      0      0 I   0.0   0.0   0:00.28 kworker/u10:0-netns
     12 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_pe
     13 root       20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     14 root       20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     15 root       20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     16 root       20   0      0      0      0 S   0.0   0.0   0:09.91 ksoftirqd/0
     17 root       20   0      0      0      0 I   0.0   0.0   0:05.00 rcu_preempt
     18 root       rt   0      0      0      0 S   0.0   0.0   0:00.27 migration/0
     19 root      -51   0      0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     20 root       20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     21 root       20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     22 root      -51   0      0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     23 root       rt   0      0      0      0 S   0.0   0.0   0:00.56 migration/1

```

Ilustración 18 Directorio para observar PCB

Luego, podemos ver el estado de uno de esos procesos con `/proc/PID/status`:



```
ubuntu@ubuntu:~$ cat /proc/1390/status
Name: systemd-resolve
Umask: 0022
State: S (sleeping)
Tgid: 1390
Ngid: 0
Pid: 1390
PPid: 1
TracerPid: 0
Uid: 991 991 991 991
Gid: 991 991 991 991
FDSize: 128
Groups: 991
NSTgid: 1390
NSpid: 1390
NSpgid: 1390
NSsid: 1390
Kthreadd: 0
VmPeak: 21828 kB
VmSize: 21828 kB
VmLck: 0 kB
VmPin: 0 kB
VmHWM: 13440 kB
VmRSS: 12928 kB
RssAnon: 2304 kB
RssFile: 10624 kB
RssShmem: 0 kB
VmData: 2040 kB
VmStk: 132 kB
VmExe: 352 kB
VmLib: 11712 kB
VmPTE: 76 kB
VmSwap: 0 kB
```

Ilustración 19 Ver procesos con `/proc/PID/status`

3.25 Cree un proceso que imprima en pantalla números del 1 al 100 000 000, ejecútelo en background y revise la información de su PCB.

Para esto vamos a:

1) Crear un script en la terminal desde la máquina virtual de Ubuntu:

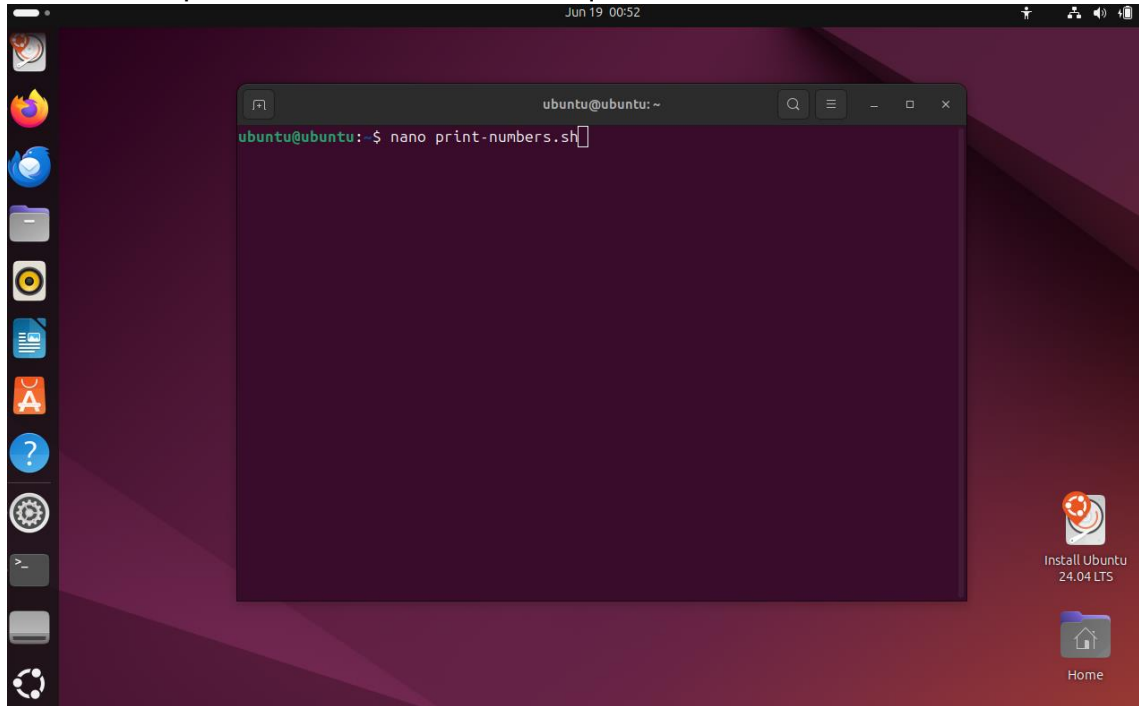
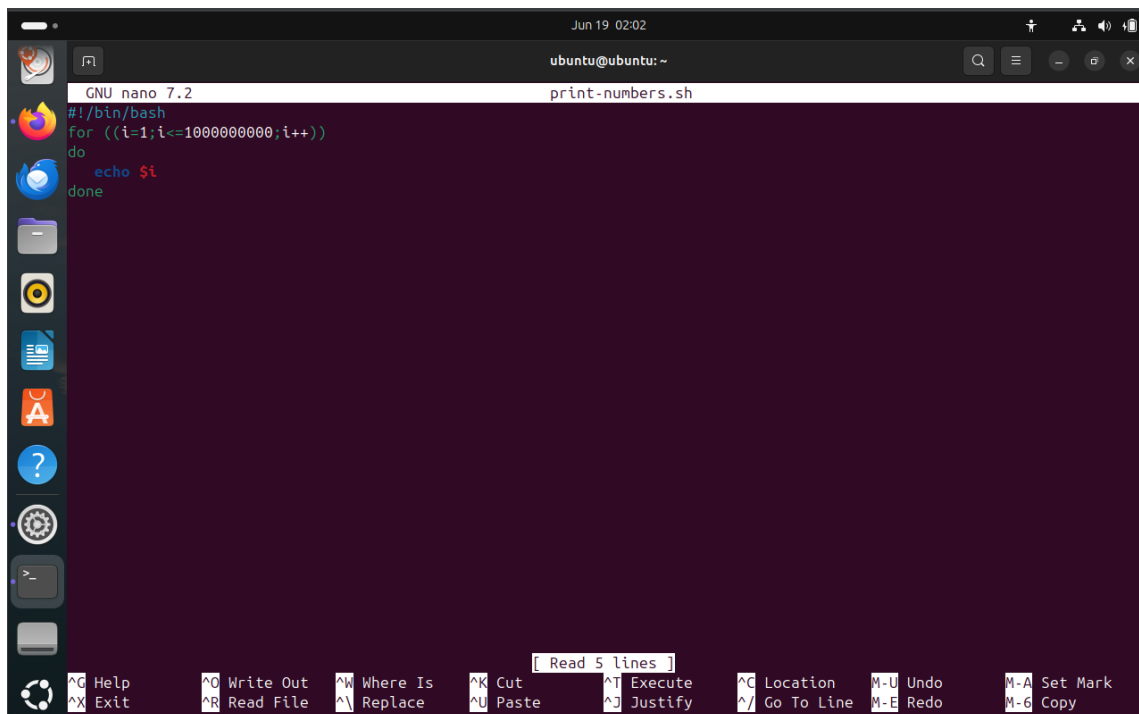


Ilustración 20 Creación del script para imprimir numeros

2)



3) Ejecutar el Script en segundo plano

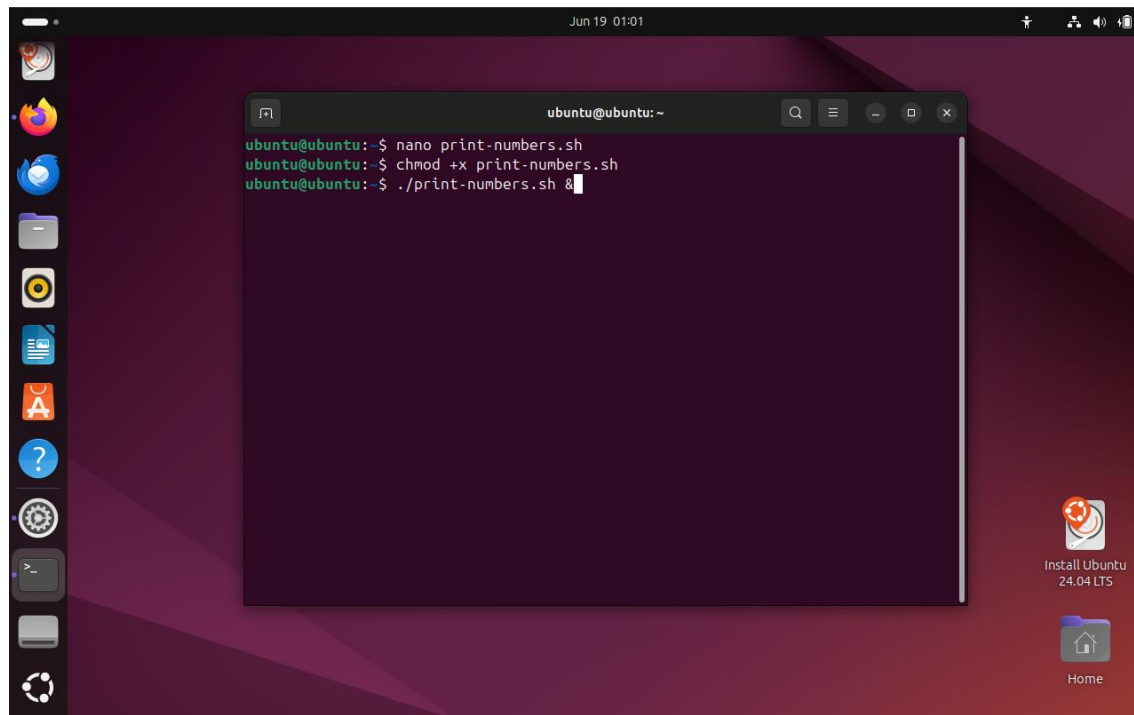


Ilustración 21 Ejecución del Script en segundo plano

- 4) Veremos si podemos ejecutar el script, si tenemos permiso o no. De no tenerlo ejecutaremos el comando `bash ./print-numbers.sh`

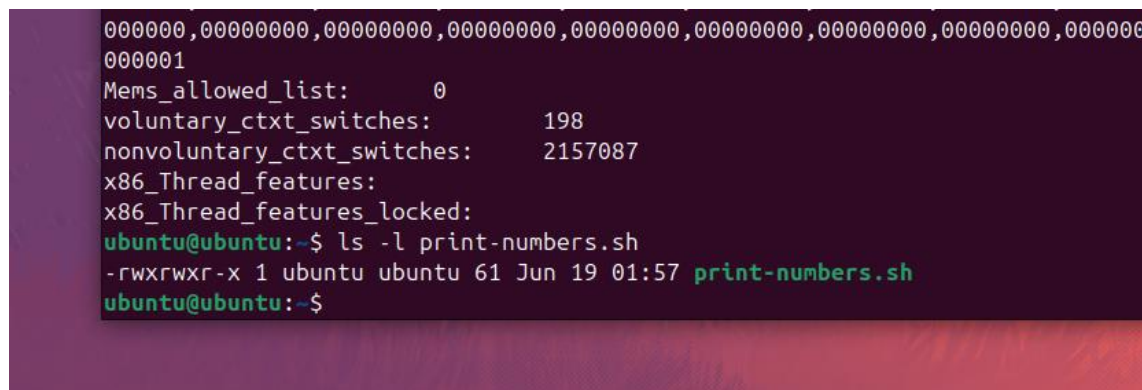
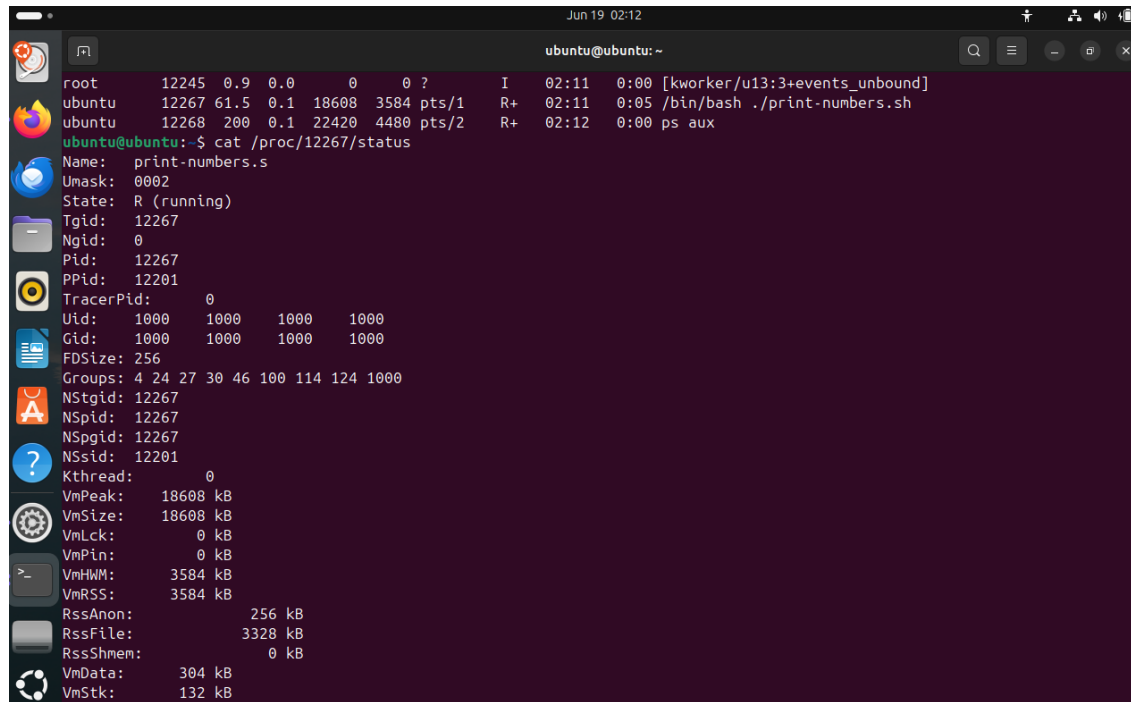


Ilustración 22 Confirmación de permiso para ejecución

1) Revisamos con `cat /proc/PID/status` cómo va la ejecución del programa



```
Jun 19 02:12
ubuntu@ubuntu: ~
root      12245  0.9  0.0   0   0 ?        I   02:11   0:00 [kworker/u13:3+events_unbound]
ubuntu    12267 61.5  0.1  18608 3584 pts/1    R+  02:11   0:05 /bin/bash ./print-numbers.sh
ubuntu    12268  200  0.1  22420 4480 pts/2    R+  02:12   0:00 ps aux
ubuntu@ubuntu:~$ cat /proc/12267/status
Name: print-numbers.s
Umask: 0002
State: R (running)
Tgid: 12267
Ngid: 0
Pid: 12267
PPid: 12201
TracerPid: 0
Uid: 1000 1000 1000 1000
Gid: 1000 1000 1000 1000
FDSize: 256
Groups: 4 24 27 30 46 100 114 124 1000
NSTgid: 12267
NSpid: 12267
NSpgid: 12267
NSSid: 12201
Kthreed: 0
VmPeak: 18608 kB
VmSize: 18608 kB
VmLck: 0 kB
VmPin: 0 kB
VmHWM: 3584 kB
VmRSS: 3584 kB
RssAnon: 256 kB
RssFile: 3328 kB
RssShmem: 0 kB
VmData: 304 kB
VmStk: 132 kB
```

Ilustración 23 Lo que sucede en segundo plano desde el PIC

[illegible]

Podemos ver la ejecución en primer plano solo escribiendo `./print-numbers` sin el `&` porque éste es el que lo hace ejecutarse en segundo plano.

```
ubuntu@ubuntu: ~  
27520302  
27520303  
27520304  
27520305  
27520306  
27520307  
27520308  
27520309  
27520310  
27520311  
27520312  
27520313  
27520314  
27520315  
27520316  
27520317  
27520318  
27520319  
27520320  
27520321  
27520322  
27520323  
27520324  
27520325  
27520326  
27520327  
27520328
```

Ilustración 24 Ejecución del script en primer plano

3. CONCLUSIONES Y RECOMENDACIONES

Conclusiones de la Práctica

1. **Dominio de Comandos Básicos:** La práctica permitió adquirir un dominio sólido de los comandos básicos de la terminal de Ubuntu, esenciales para la administración eficiente del sistema operativo.
2. **Eficiencia Operativa:** Se observó una mejora significativa en la eficiencia al ejecutar tareas como la gestión de archivos, la navegación por directorios y la administración de procesos mediante comandos específicos.
3. **Aplicación Práctica de Conocimientos:** La experiencia práctica facilitó la aplicación de conceptos teóricos aprendidos en clase sobre sistemas operativos, reforzando la comprensión de su funcionamiento en entornos reales.

Recomendaciones

1. **Práctica Continua:** Es recomendable continuar practicando regularmente los comandos aprendidos para mantener y mejorar las habilidades en la terminal de Ubuntu.
2. **Exploración de Funcionalidades Avanzadas:** Se recomienda explorar comandos más avanzados y sus aplicaciones para ampliar el conjunto de habilidades en la administración de sistemas.
3. **Uso de Recursos Adicionales:** Utilizar recursos adicionales como documentación oficial, tutoriales en línea y comunidades de usuarios para seguir aprendiendo y resolver dudas específicas.
4. **Experimentación Segura:** Realizar pruebas en entornos de desarrollo o máquinas virtuales para evitar riesgos en sistemas en producción mientras se experimenta con nuevos comandos y técnicas.

4. BIBLIOGRAFÍA

[1] Hostinger. "Linux comandos." [En línea]. Disponible en:
<https://www.hostinger.es/tutoriales/linux-comandos>. [Consulta: 18-jun-2024].

[2] Dell. "Introducción a los comandos básicos de solución de problemas en Ubuntu Linux." [En línea]. Disponible en: <https://www.dell.com/support/kbdoc/es-cl/000123974/introduccion-a-los-comandos-basicos-de-solucion-de-problemas-en-ubuntu-linux>. [Consulta: 18-jun-2024].