

Universidad Tecnológica de Panamá

Sistemas Operativos I

Experiencia Práctica 4

Profesora Aris Castillo de Valencia

Por: Gabriel H. Grimaldo R.

Cedula:8-949-56

Objetivos:

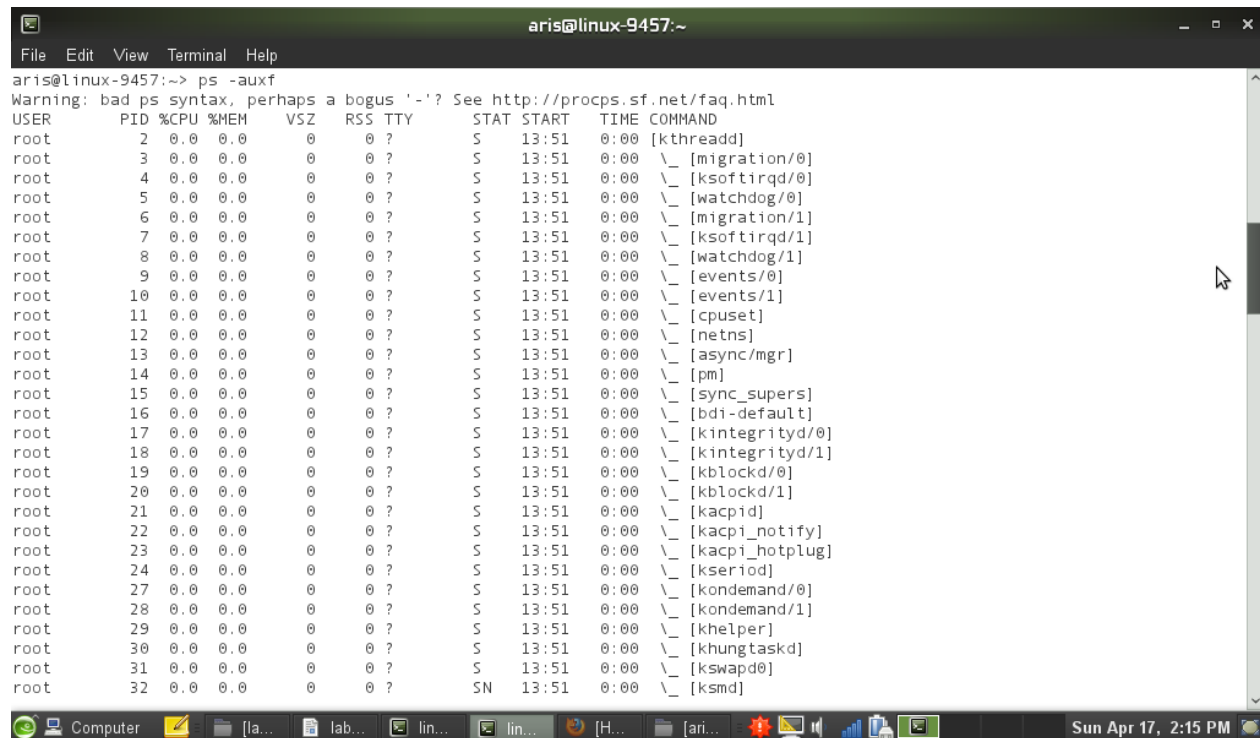
- Probar y distinguir distintos comandos para trabajar con archivos en línea de texto, incluyendo:
 - Desplegar los procesos activos en un momento en particular y ver su estado.
 - Crear procesos nuevos.
 - Eliminar procesos

Procedimiento:

Lea cuidadosamente la guía; pruebe cada uno de los comandos listados prestando especial atención a los resultados obtenidos y a las variantes que le ofrecen las opciones de los comandos. Ponga en práctica los comandos aprendidos haciendo los ejercicios sugeridos. Llene la autoevaluación y retroalimentación y súbala a la plataforma Moodle.

¿Cómo puedo saber qué procesos están corriendo?

Para desplegar en la pantalla los procesos activos se puede utilizar el comando **ps**. La salida es como se muestra en la siguiente figura.



```
aris@linux-9457:~$ ps -auxf
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2  0.0  0.0      0     0 ?        S    13:51   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [migration/0]
root         4  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [watchdog/0]
root         6  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [migration/1]
root         7  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [ksoftirqd/1]
root         8  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [watchdog/1]
root         9  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [events/0]
root        10  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [events/1]
root        11  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [cpuset]
root        12  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [netns]
root        13  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [async/mgr]
root        14  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [pm]
root        15  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [sync_supers]
root        16  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [bdi-default]
root        17  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kintegrityd/0]
root        18  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kintegrityd/1]
root        19  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kblockd/0]
root        20  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kblockd/1]
root        21  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kacpid]
root        22  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kacpi_notify]
root        23  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kacpi_hotplug]
root        24  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kseriod]
root        27  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kondemand/0]
root        28  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kondemand/1]
root        29  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [khelper]
root        30  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [khungtaskd]
root        31  0.0  0.0      0     0 ?        S    13:51   0:00 \_ [kswapd0]
root        32  0.0  0.0      0     0 ?        SN   13:51   0:00 \_ [ksmd]
```

¿Qué significa cada una de las columnas de la salida del comando ps?

User – identifica al usuario que generó el proceso.

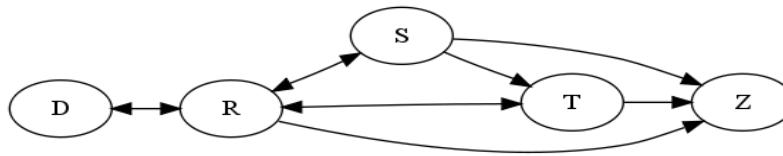
PID – indica el número que identifica al proceso o el Process ID.

%CPU – Indica el porcentaje de CPU que está consumiendo el proceso.
 %MEM - Indica el porcentaje de memoria total que está consumiendo el proceso.
 VSZ – Virtual Set Size, se refiere a la memoria virtual utilizada, en KB (Kilobytes).
 RSS – Resident Set Size, se refiere al tamaño de memoria física (non swapped) usada por el proceso, en KB.
 TTY – Indica la terminal donde está corriendo el proceso.
 STAT – State. Indica el estado del proceso. Si además de la letra que identifica el proceso hay una letra en minúscula, se trata de una bandera o flag - “s” significa que el proceso es líder de sesión; “+” significa que el proceso es parte del grupo de procesos de foreground.
 START – Indica el momento en que el proceso inició.
 TIME – Indica el tiempo que lleva el proceso desde su inicio.
 COMMAND – Indica el comando que generó el proceso y su ubicación en el sistema de archivo.

Las opciones son: -a (all) muestra toda la información de los procesos, -u muestra los usuarios dueños de procesos y -x muestra los procesos ejecutables.

¿Qué significan los estados de los procesos en Linux?

El comando ps puede mostrar los procesos en Linux en uno de estados según se muestra en la figura siguiente.



D – Uninterruptible sleep o dormido ininterrumpible. El proceso está esperando un evento.
 R – running o en ejecución o en cola de ejecución. El proceso está en ejecución.
 S – interruptible sleep o dormido interrumpible. El proceso está en espera de un evento o señal.
 T – Stopped o detenido. El proceso se ha detenido ya sea por una señal de control o porque se le está siguiendo la pista a través de un debugger.
 Z – Zombie. Es un proceso que se ha detenido o terminado pero que no se le ha eliminado por completo del sistema.

Si sólo quiere ver lo que otros usuarios están haciendo, **ps -ag** desplegará información acerca de los procesos que se están ejecutando en ese momento. Haga la prueba.

Otra forma de ver los procesos que están activos es a través del comando **top**. La diferencia estriba en que este comando muestra en tiempo real los procesos que están en ejecución. Allí usted verificar como los procesos van cambiando de estado a medida que entran en estado R (running); es decir cuando están en el CPU. La salida del comando top sería como se muestra en la figura siguiente.

```
aris@linux-9457:~  
File Edit View Terminal Help  
top - 17:54:34 up 4:02, 3 users, load average: 0.25, 0.16, 0.05  
Tasks: 161 total, 2 running, 158 sleeping, 1 stopped, 0 zombie  
Cpu(s): 2.1%us, 0.8%sy, 0.2%ni, 96.8%id, 0.0%wa, 0.2%hi, 0.0%si, 0.0%st  
Mem: 1019196k total, 967936k used, 51260k free, 34536k buffers  
Swap: 2102268k total, 0k used, 2102268k free, 640488k cached  
  
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
6337 aris 20 0 470m 99m 31m S 2 10.0 11:25.44 firefox  
1556 root 20 0 59516 23m 10m S 1 2.3 6:56.79 Xorg  
3065 aris 20 0 320m 93m 67m S 1 9.4 3:00.47 soffice.bin  
3123 aris 20 0 130m 15m 10m S 1 1.5 0:11.62 gnome-terminal  
8254 aris 20 0 2748 1124 824 R 1 0.1 0:00.05 top  
28 root 20 0 0 0 0 S 0 0.0 0:06.51 kondemand/1  
1321 messageb 20 0 3536 1644 792 S 0 0.2 0:09.03 dbus-daemon  
2012 root 20 0 6248 3008 2460 S 0 0.3 0:01.43 upowerd  
2024 root 20 0 19052 4784 3916 S 0 0.5 0:03.44 NetworkManager  
2436 aris 20 0 44796 18m 4808 S 0 1.8 2:07.39 compiz  
1 root 20 0 2200 724 612 S 0 0.1 0:01.41 init  
2 root 20 0 0 0 0 S 0 0.0 0:00.00 kthreadd  
3 root RT 0 0 0 0 S 0 0.0 0:00.02 migration/0  
4 root 20 0 0 0 0 S 0 0.0 0:00.25 ksoftirqd/0  
5 root RT 0 0 0 0 S 0 0.0 0:00.00 watchdog/0  
6 root RT 0 0 0 0 S 0 0.0 0:00.02 migration/1  
7 root 20 0 0 0 0 S 0 0.0 0:00.42 ksoftirqd/1  
8 root RT 0 0 0 0 S 0 0.0 0:00.00 watchdog/1  
9 root 20 0 0 0 0 S 0 0.0 0:00.84 events/0  
10 root 20 0 0 0 0 S 0 0.0 0:01.28 events/1  
11 root 20 0 0 0 0 S 0 0.0 0:00.00 cpuset  
12 root 20 0 0 0 0 S 0 0.0 0:00.00 netns  
13 root 20 0 0 0 0 S 0 0.0 0:00.00 async/mgr  
14 root 20 0 0 0 0 S 0 0.0 0:00.00 pm  
15 root 20 0 0 0 0 S 0 0.0 0:00.02 sync_supers
```

El comando **top** muestra más información que **ps**. Veamos lo que significan los distintos elementos mostrados:

- **us** – user CPU time; % de tiempo el CPU ha estado ejecutando procesos de usuario que no han sido bajados de prioridad.
- **sy** – System CPU time; el porcentaje de tiempo el CPU ha estado ejecutando el kernel y sus procesos.
- **ni** – Nice CPU time; el porcentaje de tiempo el CPU ha estado ejecutando procesos de usuario que han sido bajados de prioridad.
- **wa** – I/O wait; tiempo que el CPU ha estado esperando por que se completen tareas de I/O.
- **hi** – Hardware IRQ; tiempo que el CPU ha estado dando servicio a interrupciones de hardware.
- **si** – software interrupts; tiempo que el CPU ha estado dando servicio a interrupciones de software.

Ejercicio

1. Abrir la terminal de consola y escriba el comando **ps -axuf**. Describa la salida.
2. Ahora abra otra terminal de texto. Nuevamente escriba el comando y redireccione a un archivo llamado **procesoshoy.txt**. Este archivo contendrá la salida del comando **ps**. Después de ejecutar el comando, revise el archivo.

ps -axuf > procesoshoy.txt

Nota: Si tiene problemas, pruebe las opciones del comando sin el guión.

Existen otros comandos para tratar con los procesos, entre ellos **bg** que muestra los procesos de background y **fg** que trae los procesos más recientes a foreground.

¿Cómo puedo saber solamente la cantidad de memoria disponible?

El comando **free** muestra la información completa sobre la memoria del sistema, incluyendo el total, la cantidad siendo usada actualmente, la cantidad disponible. También muestra información sobre la memoria de intercambio o swap y los buffers usados por el procesador.

¿Cómo puedo terminar un proceso arbitrariamente?

El comando **kill** (**exterminar**) permite terminar un proceso con una señal s.

La sintaxis es: `kill pid`

El resultado será se que terminará el proceso cuyo número de identificación (pid) usted colocó con el comando.

Ejercicio:

Abra la aplicación de la calculadora, verifique el ID del proceso y luego termine dicho proceso con `kill`

`#kill processID`

También se puede utilizar el comando agregando señales. La sintaxis sería **kill -s pid**.

Donde -s es el número de señal para eliminar el proceso. En ese caso, 9.

Ejemplo: `#kill -9 processID`

¿Cómo puedo crear procesos en Linux?

Se puede crear procesos con sólo iniciar aplicaciones.

Ejercicio.

En el ambiente gráfico abra dos aplicaciones distintas. Verifique con el comando correspondiente e identifique la información sobre dicho proceso. ¿Qué ID tiene el proceso? ¿Qué otra información le brinda el sistema sobre dicha aplicación? Investigue y describa el significado de la información desplegada.

También puede crear procesos programándolos con un archivo ejecutable, a través de la función **fork()**. Las funciones `getPID()` obtiene el ID del proceso, mientras que `getPPID()` obtiene el ID del proceso padre. Hagamos un programa en C para esto. Usaremos el programa VI para escribir el código y luego se compilará y ejecutará el programa.

Paso 1: Ingresar vi: vi nombre del archivo.c

Escriba el código del siguiente programa:

```
#include <sys/types.h>
#include <stdio.h>
main( )
{
    pid_t pid;
    int i;    int n = 10;
```

```

    for (i = 0; i < n; i++)
    {
        pid = fork( );
        if (pid != 0)
            break;
    }
    printf("El padre del proceso %d es %d\n", getpid( ), getppid( ));
}

```

Para guardar el código y salir del editor vi, ejecute :wq

Paso 2: Compile el programa: `gcc -o programa programa.c`

Paso 3: Ejecutar el programa: `./programa`

Describa la salida. Haga un diagrama de la jerarquía de los procesos creados.

¿Se le puede cambiar la prioridad a un proceso?

El comando **nice** ejecuta un proceso con prioridad modificada.

10 by default, range goes from -20 (highest priority) to 19 (lowest).

Ejemplo: (sleep 10 seconds with lowest priority)

```
# nice -19 sleep 10
```

Si Linux es un sistema operativo multiusuario, ¿cómo pueden dos usuarios trabajar juntos en la misma computadora a la vez?

El comando **su** (switch user) permite iniciar una sesión con otro usuario. Ejemplo: `su root`

Resultado: una vez que el usuario introduce su contraseña, el sistema operativo inicia una sesión en paralelo del usuario root y el usuario actualmente en sesión.

Ambos usuarios pueden tener trabajos distintos en ejecución; el sistema operativo es capaz de mantener separadamente cada uno de los trabajos y asignar recursos entre ambos. Se pueden tener hasta X sesiones simultáneas de distintos usuarios. Las cuentas de usuarios tienen que haberse creado antes.

Ejercicio.

Si ha iniciado con su usuario regular, abra otra consola de comando y allí inicie una sesión con el usuario root. Inicie algún proceso con este usuario (puede ser una aplicación cualquiera). Ejecute el comando `ps` y `top` con ambos usuarios. ¿En qué se diferencia la información de cada usuario?

¿Cómo puedo saber qué usuarios están conectados en el sistema?

El comando **who** (quién) muestra los usuarios conectados, en qué terminal tienen su sesión y la fecha y hora desde su inicio. En este caso pts significa “pseudo terminal slave” que es una consola de teclado y monitor desde donde el usuario ejecuta los comandos. Con el comando **who**

am i, se muestra el nombre del usuario con el que se encuentra conectado en el sistema actualmente.

El comando **last** muestra los últimos usuarios que han iniciado sesión en el sistema y la fecha en que lo han hecho. Cuando vemos TTY en la salida del comando, esto se refiere generalmente al dispositivo de salida tipo video; en nuestro caso la pantalla.

¿Cómo sé por cuánto tiempo el sistema ha estado corriendo?

El comando **uptime** muestra el tiempo que el sistema ha estado sesionando. Le imprime la hora actual, el periodo de tiempo que el sistema ha estado corriendo, la cantidad de usuarios que están activos y el promedio de trabajos que han estado en ejecución.

Retroalimentación y autoevaluación.

1. Entregue cada una de las preguntas de ejercicio.

1. Ejercicio 1

1. Abrir la terminal de consola y escriba el comando `ps -axuf`. Describa la salida. Se nos muestra el estatus de todos los procesos y los porcentajes de cpu y de memoria que se encuentran usando.

```
ghr@ghr-VirtualBox:~$ ps -axuf
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	2	0.0	0.0	0	0	?	S	13:56	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	13:56	0:00	_ [rcu_gp]
root	4	0.0	0.0	0	0	?	I<	13:56	0:00	_ [rcu_par
root	5	0.0	0.0	0	0	?	I	13:56	0:00	_ [kworker
root	6	0.0	0.0	0	0	?	I<	13:56	0:00	_ [kworker
root	7	0.0	0.0	0	0	?	I	13:56	0:00	_ [kworker
root	8	0.0	0.0	0	0	?	I	13:56	0:00	_ [kworker
root	9	0.0	0.0	0	0	?	I<	13:56	0:00	_ [mm_perc
root	10	0.0	0.0	0	0	?	S	13:56	0:00	_ [ksoftir
root	11	0.0	0.0	0	0	?	I	13:56	0:00	_ [rcu_sch
root	12	0.0	0.0	0	0	?	S	13:56	0:00	_ [migrati
root	13	0.0	0.0	0	0	?	S	13:56	0:00	_ [idle_in
root	14	0.0	0.0	0	0	?	S	13:56	0:00	_ [cpuhp/0
root	15	0.0	0.0	0	0	?	S	13:56	0:00	_ [cpuhp/1
root	16	0.0	0.0	0	0	?	S	13:56	0:00	_ [idle_in
root	17	0.0	0.0	0	0	?	S	13:56	0:00	_ [migrati
root	18	0.0	0.0	0	0	?	S	13:56	0:00	_ [ksoftir
root	19	0.0	0.0	0	0	?	I	13:56	0:00	_ [kworker
root	20	0.0	0.0	0	0	?	I<	13:56	0:00	_ [kworker
root	21	0.0	0.0	0	0	?	S	13:56	0:00	_ [cpuhp/2
root	22	0.0	0.0	0	0	?	S	13:56	0:00	_ [idle_in
root	23	0.0	0.0	0	0	?	S	13:56	0:00	_ [migrati

2. Ahora abra otra terminal de texto. Nuevamente escriba el comando y redireccione a un archivo llamado procesoshoy.txt. Este archivo contendrá la salida del comando ps. Después de ejecutar el comando, revise el archivo.
ps -auxf > procesoshoy.txt

El comando crea un archivo de texto con la información que sobre el estado de los procesos. El archivo de texto es garficamente igual a cuando sale en el terminal.

```
ghr@ghr-VirtualBox:~$ ps auxf > procesoshoy.txt
```



2. Ejercicio 2

1. Abra la aplicación de la calculadora, verifique el ID del proceso y luego termine dicho proceso con kill
#kill processID

ID del proceso :12312

```
ghr@ghr-VirtualBox:~$ ps -ef | grep calculator
ghr      12312    1553    0 15:38 ?        00:00:00 gnome-calculator
ghr      12506    12337    0 15:52 pts/0    00:00:00 grep --color=auto calculato
```

```
ghr@ghr-VirtualBox:~$ kill -9 12312
```

3. Ejercicio 3:

1. En el ambiente gráfico abra dos aplicaciones distintas. Verifique con el comando correspondiente e identifique la información sobre dicho proceso. ¿Qué ID tiene el proceso? ¿Qué otra información le brinda el sistema sobre dicha aplicación? Investigue y describa el significado de la información desplegada.

- Aplicación: Firefox web browser
 - ID: 2659
 - %CPU: 5
 - %MEM:1.8
 - VSZ (“Tamaño de la Memoria Virtual”):2410472
 - RSS (“Conjunto Residente de Tamaño”):114224
 - Stat :Sl
 - Start: 14:13
 - Time: 00:00

```
ghr      2659    5.0    1.8 2410472 114224 ?        Sl   14:13    0:00    \_ /usr
184 ghr      2659    0.7    1.8 2409448 113888 ?        Sl   14:13    0:00    \_ /usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -
prefLen 85 -prefMapSize 214147 -parentBuildID 20200720193547 -appdir /usr/
lib/firefox/browser 2409 true tab
```

- Aplicación: Thunderbird Mail
 - ID: 2808
 - %CPU: 25.5
 - %MEM:4.2

- ```
152 ghr 2808 3.3 3.7 2866304 228680 ? SL 14:13 0:03 | _ /usr/lib/thunderbird/thunderbird
ghr 2808 25.5 4.2 2869064 254664 ? SL 14:13 0:03 | _ /usr
```

```
graph TD; 1213 --> 18403; 1213 --> 18409; 1213 --> 18410; 18403 --> 18404; 18403 --> 18405; 18403 --> 18406; 18404 --> 18405; 18404 --> 18406; 18405 --> 18406; 18405 --> 18407; 18409 --> 18411; 18410 --> 18411; 12871 --> 18401; 18401 --> 18402;
```



```

ghr@ghr-VirtualBox:~$./ejecutable1
El padre del proceso 18401 es12871
El padre del proceso 18402 es18401
ghr@ghr-VirtualBox:~$ El padre del proceso 18403 es1213
El padre del proceso 18404 es18403
El padre del proceso 18405 es18404
El padre del proceso 18406 es18405
El padre del proceso 18407 es18406
El padre del proceso 18408 es18407
El padre del proceso 18409 es1213
El padre del proceso 18410 es1213
El padre del proceso 18411 es18410

```

#### 4. Ejercicio 4:

1. Si ha iniciado con su usuario regular, abra otra consola de comando y allí inicie una sesión con el usuario root. Inicie algún proceso con este usuario (puede ser una aplicación cualquiera). Ejecute el comando ps y top con ambos usuarios. ¿En qué se diferencia la información de cada usuario?

En el usuario root se observan procesos que no aparecen para el otro usuario, esto supongo que se debe a que el usuario root tiene los niveles de privilegios mas altos.

- Usuario

```

ghr@ghr-VirtualBox:~$ ps
PID root@ghr-VirtualBox: /home/ghr
20404 pts/2 00:00:00 bash
20410 pts/2 00:00:00 ps
ghr@ghr-VirtualBox:~$ top
top - 17:42:28 up 3:46, 1 user, load average: 0.31, 0.10, 0.03
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.8 us, 0.2 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 5890.0 total, 2717.6 free, 1201.7 used, 1970.7 buff/cache
MiB Swap: 929.4 total, 929.4 free, 0.0 used, 4384.4 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1553 ghr 20 0 4765932 371364 125988 S 2.0 6.2 4:47.76 gnome-+
 9469 ghr 20 0 3382788 283656 141880 S 0.7 4.7 0:39.95 MainTh+
 9636 ghr 20 0 2592572 205392 124428 S 0.7 3.4 0:35.90 Web Co+
 11 root 20 0 0 0 0 I 0.3 0.0 0:02.27 rcu_sc+
 1277 ghr 20 0 692628 79392 46576 S 0.3 1.3 1:13.74 Xorg
20411 ghr 20 0 20472 3992 3480 R 0.3 0.1 0:00.14 top
 1 root 20 0 168832 13048 8516 S 0.0 0.2 0:06.88 systemd
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthrea+
 3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp+
 4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_pa+
 6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworke+
 9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_per+
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.11 ksofti+
 12 root rt 0 0 0 0 S 0.0 0.0 0:00.11 migrat+

```

- Usuario Root

```

ghr@ghr-VirtualBox:~$ sudo su
[sudo] password for ghr:
root@ghr-VirtualBox: /home/ghr# ps
 PID TTY TIME CMD
 20391 pts/1 00:00:00 sudo
 20392 pts/1 00:00:00 su
 20393 pts/1 00:00:00 bash
 20400 pts/1 00:00:00 ps
root@ghr-VirtualBox: /home/ghr# top

```

```

top - 17:41:42 up 3:45, 1 user, load average: 0.15, 0.05, 0.01
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie
top - 17:43:44 up 3:47, 1 user, load average: 0.16, 0.11, 0.04
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.2 us, 0.0 sy, 0.0 ni, 97.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 5890.0 total, 2715.6 free, 1203.7 used, 1970.7 buff/cache
MiB Swap: 929.4 total, 929.4 free, 0.0 used, 4382.5 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1553 ghr 20 0 4765932 371352 125988 S 3.6 6.2 4:49.11 gnome-+
 1277 ghr 20 0 692628 79392 46576 S 1.3 1.3 1:14.12 Xorg
19937 ghr 20 0 970960 52836 39936 S 1.0 0.9 0:01.93 gnome-+
 9469 ghr 20 0 3383812 286016 141880 S 0.3 4.7 0:40.21 MainTh+
 9636 ghr 20 0 2592572 205392 124428 S 0.3 3.4 0:36.29 Web Co+
20401 root 20 0 20476 3852 3336 R 0.3 0.1 0:00.49 top
20411 ghr 20 0 20472 3992 3480 S 0.3 0.1 0:00.33 top
 1 root 20 0 168832 13048 8516 S 0.0 0.2 0:06.89 systemd
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthrea+
 3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
 4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_pa+
 6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworke+
 9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_per+
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.11 ksoftt+
 11 root 20 0 0 0 0 I 0.0 0.0 0:02.28 rcu_sc+
 12 root rt 0 0 0 0 S 0.0 0.0 0:00.11 migrat+

```

- Busque 5 comandos relacionados con los discutidos en esta guía. Pruébelos. Describa sus usos y escriba ejemplos específicos completos, incluyendo la sintaxis y opciones utilizadas.ps

1. Sudo iotop: Muestra la utilización de memoria del disco.

```

Total DISK READ: 0.00 B/s | Total DISK WRITE: 0.00 B/s
Current DISK READ: 0.00 B/s | Current DISK WRITE: 0.00 B/s

 TID PRIO USER DISK READ DISK WRITE SWAPIN IO> COMMAND
 1 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [init splash
 2 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [kthreadd]
 3 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [rcu_gp]
 4 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [rcu_par_gp]
 6 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [kworker-kblockd]
 9 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [mm_percpu_wq]
 10 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [ksoftirqd/0]
 11 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [rcu_sched]
 12 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [migration/0]
 13 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [idle_inject/0]
 14 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [cpuhp/0]
 15 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [cpuhp/1]
 16 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [idle_inject/1]
 17 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [migration/1]
 18 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [ksoftirqd/1]
 20 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [kworker-kblockd]
 21 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [cpuhp/2]
 22 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [idle_inject/2]
 23 rt/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [migration/2]
 24 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [ksoftirqd/2]
 26 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [kworker-kblockd]
 27 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [kdevtmpfs]
 28 be/0 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [netns]
 29 be/4 root 0.00 B/s 0.00 B/s 0.00 % 0.00 % [rcu_tasks_kthre]

keys: any: refresh q: quit i: ionice o: active p: proc a: accum
sort: r: asc left: SWAPIN right: COMMAND home: TID end: COMMAND

```

- Glances: Es una herramienta de monitorización para sistemas operativos Linux que te permitirá tener bajo control los principales aspectos del sistema. Con Glances podremos ver la máxima información en el mínimo espacio posible en consola

```

ghr-VirtualBox - IP 10.0.2.15/24 Pub 181.197.119.199 Uptime: 1:34:53

CPU [7.0%] CPU: 7.0% MEM 22.8% SWAP 0.0% LOAD 3-core
MEM [22.8%] user: 5.0% total: 5.75G
SWAP [0.0%] system: 0.8% used: 1.31G
idle: 94.2% free: 4.44G free: 929M 1 min: 0.36
15 min: 0.29

NETWORK Rx/s Tx/s TASKS 202 (592 thr), 1 run, 153 slp, 48 oth
enp0s3 1Kb 792b CPU% 11.6 MEM% 0.2 PID USER THR NI S Command
lo 232b 232b 3.3 0.9 9758 root 1 0 S /usr/bin/
DefaultGateway 19ms 1.7 5.5 1553 ghr 15 0 S /usr/bin/
DISK I/O R/s W/s 1.3 1.1 1277 ghr 8 0 S /usr/lib/
sda 0 0 0.7 0.9 8845 ghr 5 0 S /usr/libe
sda1 0 0 0.0 3.8 9469 ghr 49 0 S /usr/lib/
sda2 0 0 0.0 2.5 9533 ghr 23 0 S /usr/lib/
sda5 0 0 0.0 1.9 9583 ghr 22 0 S /usr/lib/
sr0 0 0 0.0 1.4 9636 ghr 19 0 S /usr/lib/
FILE SYS Used Total 0.0 1.2 1424 ghr 10 0 S /usr/libe
/ (sda5) 6.57G 19.2G 0.0 1.0 1676 ghr 6 0 S /usr/libe
0.0 1.0 3290 ghr 6 0 S /usr/bin/
0.0 1.0 1430 ghr 9 0 S /usr/libe
0.0 0.9 10745 root 1 0 S /usr/bin/

2020-09-10 15:31:12 EST

```

- Pstree: Muestra los procesos en una estructura jerárquica, igual a un diagrama de árbol



```
ghr@ghr-VirtualBox:~$ pstree
systemd
├── ModemManager──2*[{ModemManager}]
├── NetworkManager──2*[{NetworkManager}]
├── accounts-daemon──2*[{accounts-daemon}]
├── acpid
├── avahi-daemon──avahi-daemon
├── colord──2*[{colord}]
├── cron
├── cups-browsed──2*[{cups-browsed}]
├── cupsd
├── dbus-daemon
├── fwupd──4*[{fwupd}]
├── gdm3
│ ├── gdm-session-wor
│ │ ├── gdm-x-session
│ │ │ ├── Xorg──7*[{Xorg}]
│ │ │ └── gnome-session-b
│ │ │ ├── ssh-agent
│ │ │ └── 2*[{gnome+
│ │ └── 2*[{gdm-x-session}]
│ └── 2*[{gdm-session-wor}]
├── 2*[{gdm3}]
├── glances
├── gnome-keyring-d──3*[{gnome-keyring-d}]
├── gpg-agent
└── ibus-daemon
 ├── ibus-engine-sim──2*[{ibus-engine-sim}]
 ├── ibus-extension──9*[{ibus-extension-}]
 ├── ibus-memconf──2*[{ibus-memconf}]
 └── ibus-ui-gtk3──9*[{ibus-ui-gtk3}]
```

4. Top -o %CPU: Muestra los valores de forma ordenada tomando en cuenta el %CPU, es decir que muestra los procesos de forma descendente considerando el %CPU.

```
ghr@ghr-VirtualBox:~$ top -o %CPU
top - 16:16:32 up 2:20, 1 user, load average: 0.01, 0.02, 0.00
Tasks: 196 total, 1 running, 195 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.4 us, 0.0 sy, 0.0 ni, 99.4 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
MiB Mem : 5890.0 total, 3194.7 free, 1096.1 used, 1599.2 buff/cache
MiB Swap: 929.4 total, 929.4 free, 0.0 used, 4504.0 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1553 ghr 20 0 4755816 356704 124256 S 1.0 5.9 3:48.12 gnome-+
12275 ghr 20 0 973840 53404 40648 S 0.3 0.9 0:06.91 gnome-+
12664 ghr 20 0 20476 3900 3380 R 0.3 0.1 0:00.02 top
 1 root 20 0 168832 13024 8516 S 0.0 0.2 0:05.50 systemd
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthrea+
 3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
 4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_pa+
 6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
 9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_per+
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.10 ksoftl+
 11 root 20 0 0 0 0 I 0.0 0.0 0:01.71 rcu_sc+
 12 root rt 0 0 0 0 S 0.0 0.0 0:00.06 migrat+
 13 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_l+
```

5. **Strace -ls:** Rastrear la ejecución de cualquier ejecutable.

```
ghr@ghr-VirtualBox:~$ strace ls
execve("/usr/bin/ls", ["ls"], 0x7ffd0a498190 /* 58 vars */) = 0
brk(NULL) = 0x55f98c597000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff143af820) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=71727, ...}) = 0
mmap(NULL, 71727, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff2a8911000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0p\0\0\0\0\0\0"... , 832)
 = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=163200, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff2a890f000
f2a890f000
mmap(NULL, 174600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff2a88e4000
mprotect(0x7ff2a88ea000, 135168, PROT_NONE) = 0
mmap(0x7ff2a88ea000, 102400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7ff2a88ea000
yWRITE, 3, 0x6000) = 0x7ff2a88ea000
```

3. Relacione la fig. 3.17 del libro de texto (sobre los estados de los procesos en Linux) con el contenido del pslog.txt. Haga el diagrama con transiciones y explique.  
El diagrama muestra la forma en la que interactúan los proceso según su estado y en el caso de este diagrama se muestran los PID de los procesos que se encontraban ese estado.



