

**LABORATORIO DE SISTEMAS OPERATIVOS**

**PERÍODO ACADÉMICO:** 2024 – A

**EQUIPO:**

**PROFESOR:** Marco Sánchez PhD.

**TIPO DE INSTRUMENTO:** Guía de Laboratorio

**TEMA: ADMINISTRACIÓN DE PROCESOS EN LINUX**

**Tópico 2: Comandos básicos, auditoría, de gestión de procesos, memoria y conectividad en Linux**

**ÍNDICE DE CONTENIDOS**

1. OBJETIVOS .....	3
2. MARCO TEÓRICO .....	3
Comandos en Linux .....	3
3. PROCEDIMIENTO .....	4
3.1 Iniciando un emulador de terminal .....	4
3.2 Prompt de Comandos .....	5
3.3 Sintaxis de comandos Linux.....	5
Ejemplos de formato de comandos.....	6
3.4 Algunos comandos básicos .....	6
3.5 Sistema de ayuda .....	7
3.7 Herramientas para visualización de ficheros .....	8
3.8 Herramientas para edición de ficheros .....	9
3.9 Otros editores .....	9
3.10 INICIO DE VI:.....	9
MODO COMANDO: .....	9
MODO EDICIÓN: .....	10
MODO LÍNEA O EX: .....	12
Resumen de Comandos: .....	13
3.11. Realizar la siguiente figura con el comando vi dentro de un fichero que tendrá su nombre: .....	13
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” (\$). .....	13
3.13 Verificar los procesos que se encuentran ejecutando: .....	14
3.14 En GNU/Linux se puede crear una estructura jerárquica de procesos (relación padre-hijo).	

.....	16
3.15 Procesos en segundo plano.....	16
3.16 Mediante el comando <i>ps tree</i> 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. ....	16
3.17 Ejecutar el comando <i>yes</i> en una terminal y mediante el comando <i>ps</i> (investigar las opciones) determinar los PID y PPID del proceso asociado.....	17
3.18 Mediante el comando <i>ps</i> (investigar las opciones) determinar para el proceso asociado al comando <i>yes</i> : .....	17
3.19 Utilice el comando <i>top</i> 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 <i>top -u &lt;Usuario&gt;</i> .....	17
3.20 Los procesos pueden ser controlados en dos formas: .....	17
Procesos Foreground.....	17
Procesos de Background .....	18
3.21 Señales Kill.....	18
3.22 Cree el fichero bucle con el código: .....	20
3.23 Cambio de prioridades a los procesos .....	20
El comando <i>renice</i> .....	21
3.24 Investigue en qué directorio se puede observar el PCB de un proceso en Linux. ....	21
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. ....	21
4. INFORME.....	21

## ÍNDICE DE FIGURAS

Figura 1. Terminal de comandos dentro del ambiente gráfico CentOS.....	5
Figura 2. Formato de comandos para la terminal de CentOS.....	6
Figura 3. Resumen de los comandos para navegar dentro de vi.....	10
Figura 4. Texto de ayuda para saber si se está dentro del Modo Edición.....	11
Figura 5. Resumen de comandos del editor vi.....	13
Figura 6. Figura creada utilizando símbolos y comandos en vi.....	13
Figura 7. Ejecución del comando <i>ps</i> .....	14
Figura 8. Ejecución del comando <i>ps aux</i> .....	14
Figura 9. Ejecución del comando <i>ps l</i> .....	15
Figura 10. Ejecución del comando <i>ps -o ppid,pid,cmd   grep ps</i> .....	16
Figura 11. Verificación de los procesos activos mediante el comando <i>ps -l</i> .....	16
Figura 12. Ejecución del comando <i>ps -fe more</i> .....	16
Figura 13. Resumen de los comandos para el control de procesos.....	18
Figura 14. Ejemplo de uso del comando <i>nice</i> .....	21
Figura 15. Ejemplo de uso del comando <i>renice</i> .....	21

## ÍNDICE DE TABLAS

Tabla 1. Comandos básicos (parte 1) .....	6
Tabla 2. Comandos básicos (parte 2) .....	7
Tabla 3. Variables de entorno y su descripción .....	8
Tabla 4. Señales más importantes enviadas a un proceso .....	19

## 1. OBJETIVOS

- 1.1. Familiarizar al estudiante con el uso de los comandos básicos del sistema operativo Linux.

## 2. MARCO TEÓRICO

- Linux es un S.O. multiusuario y multitarea:
- Muchos usuarios pueden ejecutar muchas tareas al mismo tiempo, independientes entre sí.
- Siempre es necesario hacer log in, antes de usar el sistema.
- Identificarse con nombre de usuario y contraseña.
- Múltiples formas de hacer login en el sistema:
  - Consola: teclado, mouse, monitor, conectados directamente.
  - Terminal serial, Shell.
  - Conexión vía red (telnet, ssh).

### Comandos en Linux

Cualquier actividad o tarea puede hacerse en Linux mediante comandos, se incluye:

- Navegar en la web.
- Activar la GUI (Interfaz gráfica de usuario) (Sistema X Window) no es necesaria para correr el S.O. Linux.
- Para ejecutar comandos en X Window, se necesita un emulador de terminal.

Una de las tareas más importantes del sistema operativo (SO) es la gestión de los procesos que se están ejecutando en una máquina.



El hecho de que todos los procesos deban compartir los recursos hardware disponibles (memoria RAM, CPU, etc.) hace que el SO juegue un papel primordial en gestionar dichos recursos para que los procesos se ejecuten de forma simultánea (al menos de cara al usuario) y compatible.

Una posible definición de proceso es que es un programa que se encuentra en ejecución. Cada proceso, durante su ejecución, guarda información sobre su "contexto" que incluye, entre otras cosas, información sobre su proceso padre, los recursos del sistema que se están consumiendo (segmentos de memoria asignados), permisos de seguridad, etc.

### 3. PROCEDIMIENTO

#### 3.1 Iniciando un emulador de terminal

Permite ejecutar un comando Linux dentro de un ambiente gráfico (X).

- Emula una CONSOLA DE TEXTO.
- El tipo de emulador de terminal depende de la distribución.

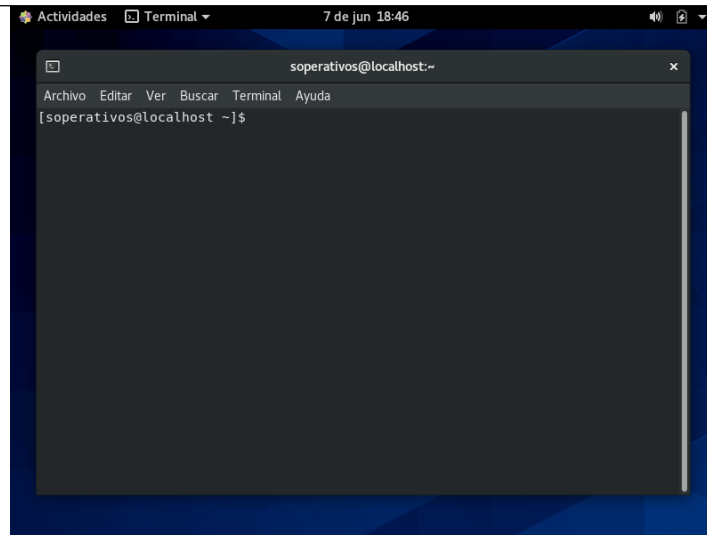


Figura 1. Terminal de comandos dentro del ambiente gráfico CentOS

### 3.2 Prompt de Comandos

El prompt de comandos indica que el sistema está listo para aceptar comandos.

- Puede ser personalizado.
- El prompt por defecto depende de la distribución.

#### Ejemplos:

**Prompt para el usuario soperativos:**

```
[soperativos@localhost ~]$
```

**Prompt para el superusuario root:**

```
[root@localhost ~] #
```

- El símbolo dólar (\$) regularmente significa “logueado como usuario regular”.
- El símbolo numeral (#) usualmente significa “logueado como usuario root”.

### 3.3 Sintaxis de comandos Linux

Los comandos en Linux tienen el siguiente formato:

**\$ comando(s) opcion(es) argumento(s)**

#### Ejemplos:

```
$ ls
```

```
$ ls -l
```

\$ ls /dev

\$ ls -l /dev

### Ejemplos de formato de comandos

#### CORRECTO

#### INCORRECTO

##### 1. Separación

\$ mail -f test  
\$ who -u

\$ mail - f test  
\$ who-u

##### 2. Orden

\$ mail -s test1 tux1  
\$ who -u

\$ mail test1 tux1 -s  
\$ -u who

##### 3. Opciones múltiples

\$ who -m -u  
\$ who -mu

\$ who -m-u  
\$ who -m u

Figura 2. Formato de comandos para la terminal de CentOS

### 3.4 Algunos comandos básicos

Comando	Descripción	Ejemplos
at [-lr] hora [fecha]	Ejecuta un comando más tarde	at 6pm Friday miscript
cal [[mes] año]	Muestra un calendario del mes/año	cal 1 2025
date [mmddhhmm] [+form]	Muestra la hora y la fecha	date
echo string	Escribe mensaje en la salida estándar	echo "Holamundo"
finger usuario	Muestra información general sobre un usuario en la red	finger druiz@pcl1.lsi.us.es
id	Número id de un usuario	id usuario
kill [-señal] PID	Enviar una señal a un proceso (dependiendo de la señal, a veces lo finalizará)	kill 1234
man comando	Ayuda del comando especificado	man gcc
passwd	Cambia la contraseña.	passwd
ps [axiu]	Muestra información sobre los procesos que se están ejecutando en el sistema	ps -ux, ps -ef
who / rwho	Muestra información de los usuarios conectados al sistema	who

Tabla 1. Comandos básicos (parte 1)

Comando	Descripción	Ejemplos
cat $f_1 \dots f_n$	Concatena y muestra los archivos	cat /etc/passwd
cd [dir]	Cambia de directorio	cd /tmp
chmod permisos fich	Cambia los permisos de un archivo	chmod +x miscript
chown usuario:grupo fich	Cambia el dueño un archivo	chown nobody miscript
cp $f_1 \dots f_n$ dir	Copia archivos	cp foo foo.backup
diff [-e] $f_1 f_2$	Encuentra diferencia entre archivos	diff foo.c newfoo.c
du [-sabr] $f_1 \dots f_n$	Devuelve el tamaño del directorio	du -s /home/
file f	Muestra el tipo de un archivo	file a.out
find dir test acción	Encuentra archivos.	find . -name ".bak" -print
grep expr $f_1 \dots f_n$	Busca patrones en archivos	grep druiz /etc/passwd
head -n f	Muestra las $n$ primeras líneas de un archivo	head prog1.c
mkdir dir	Crea un directorio.	mkdir temp
mv $f_1 \dots f_n$ dir	Mueve un archivo(s) a un directorio	mv a.out prog1
mv $f_1 f_2$	Renombra un archivo.	mv *.c prog_dir
less / more fich(s)	Visualiza página a página un archivo. (less acepta comandos vi)	more /less muy_largo.c
ln [-s] f acceso	Crea un acceso directo a un archivo	ln -s /users/mike/.profile .
ls	Lista el contenido del directorio	ls -l /usr/bin
pwd	Muestra la ruta del directorio actual	pwd
rm f	Borra un fichero.	rm foo.c
rm -r dir	Borra un todo un directorio	rm -rf prog_dir
rmdir dir	Borra un directorio vacío	rmdir prog_dir
tail -n fich	Muestra las $n$ últimas líneas de un archivo	tail prog1.c
vi fich	Edita un archivo.	vi .profile

Tabla 2. Comandos básicos (parte 2)

### 3.5 Sistema de ayuda

Linux dispone de un completo sistema de ayuda. Podemos obtener ayuda sobre cualquier comando o sobre cualquier aspecto del sistema mediante el comando *man*, cuyo formato es:

**man comando**

Dónde:

**comando** es el comando del cual se solicita información.

Ejemplo:

**man ls**

### 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*.

2. Muestre en pantalla el calendario con el comando *cal*. Luego muestre el calendario de septiembre de 1983.
3. Verifique qué usuarios se encuentran dentro del sistema con el comando *w*.
4. Limpie la pantalla de su terminal con el comando *clear*.
5. Muestre el contenido de las variables de entorno de la tabla debajo con el comando *echo* (respeta el uso de mayúsculas y minúsculas).

### Ejemplo:

#### **echo \$DISPLAY**

Variable	Descripción
DISPLAY	Dirección IP a donde se envían los gráficos de los clientes X.
HOME	Directorio personal.
HOSTNAME	Nombre de la máquina.
MAIL	Archivo de correo.
PATH	Lista de directorios donde buscar los programas.
PS1	Prompt.
SHELL	Intérprete de comandos por defecto.
TERM	Tipo de terminal.
USER	Nombre del usuario.

Tabla 3. Variables de entorno y su descripción

6. Muestre un mensaje de texto en la terminal de un usuario con el comando *write*.
7. Con el comando *wall*, coloque un mensaje en todas las pantallas de los usuarios logueados. Abrir varias terminales virtuales con Ctrl + Alt + Fn para comprobar la presentación del mensaje. Para salir del comando usar "Ctrl +D".
8. Verificar el historial de comandos con *history*.
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).
10. Verificar el resultado obtenido con la ejecución de los siguientes comandos. Explicar su uso.

\$ !-1

\$ !!

\$ !7

### 3.7 Herramientas para visualización de ficheros

- **cat** → De *catenate*, permite leer datos de ficheros y mostrar sus contenidos. Es la forma más sencilla. Cat es uno de los comandos más utilizados en Linux.
- **more** → Permite leer el contenido de un fichero o comando y visualizarlo por páginas (para avanzar pulsar la barra espaciadora).



- **less** → Permite leer el contenido de un fichero línea a línea, no se puede manipular ni editar el texto. Con este comando se puede subir y/o bajar por el texto.
- **file** → Determina el tipo de un archivo e imprime en pantalla el resultado.

### 3.8 Herramientas para edición de ficheros

Para editar un fichero el usuario debe poseer permisos de escritura.

**vi** → Es el editor por defecto en todos los sistemas Linux. Existen tres modos de operación en vi:

- **Modo comando:** Este es el modo en el que se encuentra el editor cada vez que se inicia. Las teclas ejecutan acciones (comandos) que permiten mover el cursor, ejecutar comandos de edición de texto, salir de vi, guardar cambios, etc.
- **Modo edición:** Este es el modo que se usa para insertar el texto. Existen varios comandos que se pueden utilizar para ingresar a este modo.
- **Modo línea o ex:** Se escriben comandos en la última línea al final de la pantalla.

### 3.9 Otros editores

Una distribución Linux típica incluye un gran número de editores.

Editores en modo texto:

- pico
- vim
- Emacs

Editores en modo gráfico:

- kedit, kwrite
- gedit

Editores hexadecimales: permiten modificar archivos que no son de texto.

- Khexedit

### 3.10 INICIO DE VI:

Abrir un archivo específico para su edición (\$vi NombreArchivo):

**\$vi**

**\$vi suNombre**

Nota: Si el archivo no existe, lo crea. Presione la tecla **i** para poder introducir texto, y escriba algunas líneas. Utilice **ESC** para regresar al modo comando

#### MODOS DE VI:

El editor vi, al igual que UNIX, diferencia entre mayúsculas y minúsculas. Después de abrir un fichero con el comando vi, digite los siguientes comandos

(digite las letras) y observe su funcionamiento:

h  
j  
k  
l  
^  
\$  
1G  
G  
<up arrow>  
<down arrow>  
<left arrow>  
<right arrow>  
ZZ

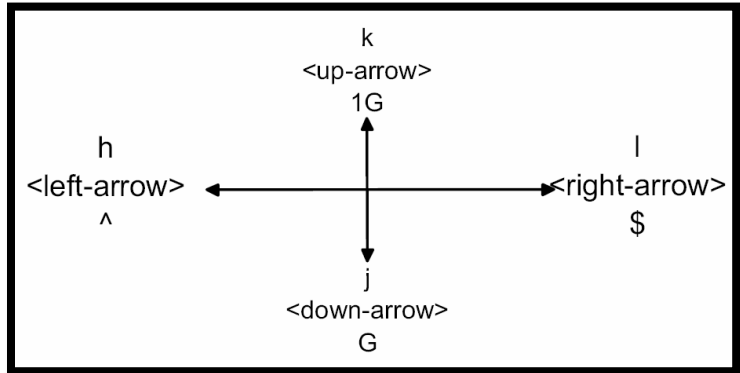


Figura 3. Resumen de los comandos para navegar dentro de vi

Resumen:

<b>&lt;left arrow&gt;</b> o <b>h</b>	un caracter a la izquierda
<b>&lt;right arrow&gt;</b> o <b>l</b>	un caracter a la derecha
<b>^</b>	moverse al inicio de la línea
<b>\$</b>	moverse al final de la línea
<b>&lt;up arrow&gt;</b> o <b>k</b>	una línea arriba
<b>&lt;down arrow&gt;</b> o <b>j</b>	una línea abajo
<b>1G</b>	ir a la primera línea
<b>G</b>	ir a la última línea

Nota: Para usar estos comandos asegurarse de estar en modo comando.

### MODOS EDICIÓN:

Para cambiar de modo comando a modo edición, dentro de algún fichero digite:  
**I** (i mayúscula, o tecla INSERT):

Tomar en cuenta que se muestra -- **INSERT** -- o -- **INSERTAR** -- en modo Edición:

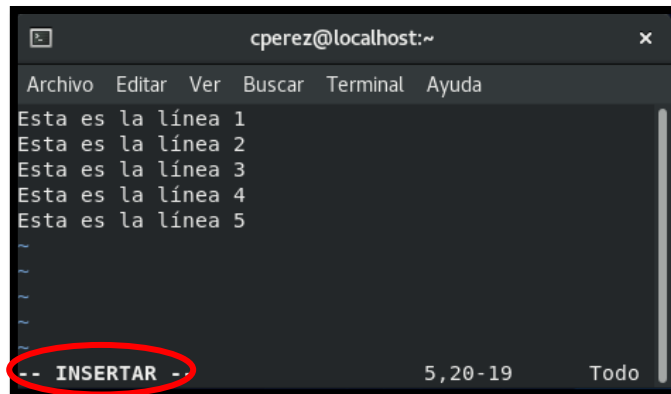


Figura 4. Texto de ayuda para saber si se está dentro del Modo Edición

Luego pruebe el funcionamiento de los siguientes comandos:

- ***i***
- ***A***
- ***a***
- ***O***
- ***o***

Para insertar texto al inicio de la línea	<b><i>I</i></b> ( <b><i>letra i mayúscula</i></b> )
Para insertar texto antes del cursor	<b><i>i</i></b>
Para agregar texto después del cursor	<b><i>a</i></b>
Para agregar texto al final de la línea	<b><i>A</i></b>
Para regresar al modo de comando	<b>&lt;ESC&gt;</b>

Dentro del Modo Comando, compruebe los siguientes comandos.

- ***x***
- ***dd***
- ***2dd***
- ***ndd*** (***siendo n un número entero cualquiera***)
- ***D***
- ***dw***

Es importante destacar que todo lo que se borra queda almacenado en un buffer (área temporal de memoria), de modo que, si se borró algo por error, puede volver a escribirse (si se hace antes de realizar otros cambios, es decir, inmediatamente luego de eliminar el texto por error). Esto se hace simplemente ejecutando el comando ***p***.

**Cortar y pegar:** Esto implica mover partes del archivo de un lugar a otro del mismo. Para esto se debe:

- Cortar el texto que se desea mover utilizando alguno de los comandos usados para borrar texto (**dd**, **2dd**, etc).
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando **p**.

**Copiar y pegar:** Esta operación difiere de la anterior. En este caso lo que se hace es repetir partes del texto en otro lugar del archivo. Para esto se debe:

- Utilizar el comando **yy**, cuya función es copiar la línea donde se encuentra situado el cursor. **8yy** es una variante que copia múltiples líneas, en este caso 8.
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando **p**.

**Deshacer cambios:** Se puede deshacer el último cambio realizado, utilizando el comando **u**.

Para cambiar de Modo Edición a Modo Comando digitar la tecla **Esc**.

#### MODULO LÍNEA O EX:

Para ingresar al Modo Línea desde el Modo Comando, se debe utilizar alguna de las siguientes teclas:

- **:**
- **/**
- **?**

Para volver al Modo Comando desde el Modo Línea, se debe digitar la tecla **ENTER** (al finalizar el comando) o la tecla **ESC** (que interrumpe el comando).

Dentro de algún fichero abierto con el comando **vi**, digite:

- **/textoABuscar**
- **?textoABuscar**
- **n**
- **:q**
- **:q!**
- **:w**
- **:w!**
- **:w archivo1**
- **:wq**
- **:x**

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

### 3.13 Verificar los procesos que se encuentran ejecutando:

\$ ps

```

  PID TTY          TIME CMD
 3109 tty3        00:00:00 bash
 3129 tty3        00:00:00 ps

```

Figura 7. Ejecución del comando ps

\$ ps aux

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.4	0.1	19356	1536	?	Ss	06:40	0:04	/sbin/init
root	2	0.0	0.0	0	0	?	S	06:40	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	06:40	0:00	[migration/0]
root	4	0.0	0.0	0	0	?	S	06:40	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S	06:40	0:00	[stopper/0]
root	6	0.0	0.0	0	0	?	S	06:40	0:00	[watchdog/0]
root	7	0.1	0.0	0	0	?	S	06:40	0:01	[events/0]
root	8	0.0	0.0	0	0	?	S	06:40	0:00	[events/0]
root	9	0.0	0.0	0	0	?	S	06:40	0:00	[events_long/0]
root	10	0.0	0.0	0	0	?	S	06:40	0:00	[events_power_ef]
root	11	0.0	0.0	0	0	?	S	06:40	0:00	[cgrouper]
root	12	0.0	0.0	0	0	?	S	06:40	0:00	[khelper]
root	13	0.0	0.0	0	0	?	S	06:40	0:00	[netns]
root	14	0.0	0.0	0	0	?	S	06:40	0:00	[async/mgr]
root	15	0.0	0.0	0	0	?	S	06:40	0:00	[pm]
root	16	0.0	0.0	0	0	?	S	06:40	0:00	[sync_supers]
root	17	0.0	0.0	0	0	?	S	06:40	0:00	[bdi-default]
root	18	0.0	0.0	0	0	?	S	06:40	0:00	[kintegrityd/0]
root	19	0.0	0.0	0	0	?	S	06:40	0:00	[kblockd/0]
root	20	0.0	0.0	0	0	?	S	06:40	0:00	[kacpid]
root	21	0.0	0.0	0	0	?	S	06:40	0:00	[kacpi_notify]
root	22	0.0	0.0	0	0	?	S	06:40	0:00	[kacpi_hotplug]
root	23	0.0	0.0	0	0	?	S	06:40	0:00	[ata_aux]
root	24	0.0	0.0	0	0	?	S	06:40	0:00	[ata_sff/0]
root	25	0.0	0.0	0	0	?	S	06:40	0:00	[ksuspend_usbd]
root	26	0.0	0.0	0	0	?	S	06:40	0:00	[khubd]
root	27	0.0	0.0	0	0	?	S	06:40	0:00	[kseriod]
root	28	0.0	0.0	0	0	?	S	06:40	0:00	[md/0]
root	29	0.0	0.0	0	0	?	S	06:40	0:00	[md_misc/0]
root	30	0.0	0.0	0	0	?	S	06:40	0:00	[linkwatch]

Figura 8. Ejecución del comando ps aux

\$ ps l

F	UID	PID	PPID	PRI	NI	VSZ	RSS	WCHAN	STAT	TTY	TIME	COMMAND
0	501	5427	5419	20	0	108336	1820	wait	Ss	pts/0	0:00	bash
0	501	5965	5427	20	0	108092	1568	signal	T	pts/0	0:00	nano
0	501	5990	5419	20	0	108336	1808	n_tty_	Ss+	pts/1	0:00	bash
0	501	6524	5427	20	0	108128	996	-	R+	pts/0	0:00	ps l

Figura 9. Ejecución del comando `ps l`

Cada una de las columnas que aparecen al ejecutar el comando anterior tiene su propio significado:

**F:** Son los indicadores asociados al proceso

**UID:** Identificación del usuario propietario del proceso

**PID:** ID de un proceso

**PPID:** ID proceso padre

**PRI:** Prioridad del proceso

**NI:** Valor de bondad. Mientras más elevado, menor es la prioridad.

**VSZ:** Tamaño de la memoria virtual del proceso en KB.

**RSS:** Tamaño de la memoria física usada en KB.

**WCHAN:** Para los procesos que esperan o están dormidos, enumera el evento que espera.

**TTY:** Terminal virtual

**TIME:** tiempo de ejecución

**CMD/COMMAND:** Comando

**STAT:** Estado del proceso. Un proceso puede tener cualquiera de los siguientes estados:

- R – Ready/Ejecutando.
- D – Interrumpido
- S – Suspendido
- s – Es el proceso líder de la sesión
- T – Detenido
- Z – Zombie
- N – Tiene una prioridad menor que lo normal.
- < – Tiene una prioridad mayor que lo normal.
- L – Tiene páginas bloqueadas en memoria
- l – Es multi-hilo.
- + – Está en el grupo de procesos en foreground

3.14 En GNU/Linux se puede crear una estructura jerárquica de procesos (relación padre-hijo).

```
$ ps -o ppid,pid,cmd | grep ps
```

```
[operativos@localhost ~]$ ps -o ppid,pid,cmd | grep ps
4618    4841  ps  -o ppid,pid,cmd
4618    4842  grep --color=auto ps
```

Figura 10. Ejecución del comando `ps -o ppid,pid,cmd | grep ps`

```
[operativos@localhost ~]$ echo $$
2963
[operativos@localhost ~]$ bash
[operativos@localhost ~]$ echo $$
3507
[operativos@localhost ~]$ ps -l
F S    UID      PID     PPID  C  PRI   NI  ADDR  SZ  WCHAN   TTY          TIME CMD
0 S    1000      2963    2958  0   80    0   -    6165  -      pts/0        00:00:00 bash
0 S    1000      3507    2963   1   80    0   -    6167  -      pts/0        00:00:00 bash
0 R    1000      3551    3507   0   80    0   -   11351  -      pts/0        00:00:00 ps
```

Figura 11. Verificación de los procesos activos mediante el comando `ps -l`

3.15 Procesos en segundo plano

```
[cloudera@quickstart ~]$ ps -fe|more
UID      PID  PPID  C  STIME TTY          TIME CMD
root         1     0  0  12:39 ?          00:00:02 /sbin/init
root         2     0  0  12:39 ?          00:00:00 [kthreadd]
root         3     2  0  12:39 ?          00:00:00 [migration/0]
root         4     2  0  12:39 ?          00:00:00 [ksoftirqd/0]
root         5     2  0  12:39 ?          00:00:00 [migration/0]
root         6     2  0  12:39 ?          00:00:00 [watchdog/0]
root         7     2  0  12:39 ?          00:00:01 [events/0]
root         8     2  0  12:39 ?          00:00:00 [cgroup]
root         9     2  0  12:39 ?          00:00:00 [khelper]
root        10     2  0  12:39 ?          00:00:00 [netns]
root        11     2  0  12:39 ?          00:00:00 [async/mgr]
root        12     2  0  12:39 ?          00:00:00 [pm]
```

Figura 12. Ejecución del comando `ps -fe|more`

3.16 Mediante el comando ***pstree*** 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?

**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.**

**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.

**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>*.**

**3.20 Los procesos pueden ser controlados en dos formas:**

1. Desde el *shell* que lo inició usando su *job number*.
2. Desde cualquier lugar en el sistema, usando su PID.

Señales enviadas a procesos:

- *Foreground*: Un proceso que recibe la entrada del teclado que es escrita en una terminal.
- *Background*: Un proceso que no recibe ninguna entrada del teclado.

### Procesos Foreground

Los procesos foreground son invocados simplemente ejecutando un comando en la línea de comandos.

## # find / -name README

### Procesos de Background

Los procesos de background se invocan colocando un & al final de la línea de comandos.

## # find / -name README &

- |                       |   |
|-----------------------|---|
| <b>&lt;Ctrl-z&gt;</b> | Suspende la tarea <i>foreground</i>                       |
| <b>jobs</b>           | Lista los jobs en <i>background</i> o suspendidos         |
| <b>fg</b>             | devuelve la tarea suspendida al <i>foreground</i>         |
| <b>bg</b>             | <u>devuelve</u> las tarea suspendida al <i>background</i> |
- Los comandos **fg**, **bg** y **kill** pueden ser usados con un número de job. Para matar al job 3: **kill %3**
  - Si se especifica el parámetro **nohup** los procesos no mueren a pesar de que el usuario se desloguee.

Figura 13. Resumen de los comandos para el control de procesos

Ejecutar:

**\$ sleep 100&**

**\$ nohup sleep 100&**

Copie los PID y, en una nueva terminal, ejecute:

**\$ ps -fe |grep sleep**

### 3.21 Señales Kill

Muchas señales pueden ser enviadas a un proceso:

- Usando interrupciones de teclado (procesos *foreground*).
- Usando el comando **kill -signal PID**.
- Ejecutar **kill -l**

Las señales más importantes:

Nombre de la señal	Número	Significado y uso
<b>HUP</b>	1	Hang up. Esta señal es enviada automáticamente cuando se desloguea o el módem es desconectado. Es utilizado por procesos “demonios” para hacer que el archivo de configuración vuelva a leerse.
<b>INT</b>	2	Interrupt. Dejar de correr. Esta señal es enviada cuando se presiona Ctrl+c.

<b>KILL</b>	9	Kill. Detener el proceso inmediatamente y sin ninguna condición. Esta es una señal enviada en casos drásticos, ya que el proceso no puede ignorarla. Es una señal de “detención de emergencia”.
<b>TERM</b>	15	Terminate. Terminar el proceso lo más ameno posible. Esta señal se utiliza para preguntar al proceso por un cierre tranquilo.
<b>TSTP</b>	20	Stop executing. Termina de ejecutar y está listo para continuar. Esta señal es enviada cuando se presiona Ctrl+z.
<b>CONT</b>	18	Continue execution. Continuar la ejecución. Esta señal es enviada para empezar de nuevo un proceso que fue detenido por SIGTSP o SIGSTOP. (La <i>Shell</i> envía esta señal cuando se utiliza los comandos <i>fg</i> o <i>bg</i> después de detener un proceso con Ctrl+z).

Tabla 4. Señales más importantes enviadas a un proceso

Si un proceso se ejecuta de manera adecuada, basta terminarlo con la señal **SIGTERM** (15).

Luego de que detenga todo lo que está haciendo, ejecute:

**\$ kill -15 PID**

Si un proceso se cuelga y no puede responder, debe usarse el comando más fuerte para detenerlo:

**\$ kill -9 PID**

**\$ kill -KILL PID**

En ocasiones la señal HUP puede ser utilizada para re-leer determinados archivos de configuración.

**\$ kill -1 `cat /var/run/rsyslog.pid`**

Ejecute el comando: **yes que tal > /dev/null**

Abra otra terminal y determine el PID del comando yes.

Termine el proceso yes.

Verifique que sucedió y el mensaje que apareció en la primera terminal.

Ejecute el comando: **yes que tal > /dev/null**

Abra otra terminal y determine el PID del comando yes.

Mate el proceso yes.

Verifique que sucedió y el mensaje que apareció en la primera terminal.

Ejecute el comando: **yes que tal > /dev/null &**

Ejecute el comando **jobs**.

¿En qué estado se encuentra el proceso yes?

¿Qué porcentaje de CPU y memoria consume el proceso yes?

Ejecute ***fg job\_ID***, y explicar qué sucede.  
Termine el proceso yes con la señal SIGTERM.

Nota: el ***job\_ID*** se refiere al número que aparece entre corchetes al ejecutar el comando ***jobs***.

Ejecute el comando: ***yes que tal > /dev/null &***

Ejecute ***kill %job\_ID***.

Ejecute el comando ***jobs***.

¿En qué estado se encuentra el proceso yes?

### 3.22 Cree el fichero **bucle** con el código:

```
#!/bin/bash  
yes que tal > /dev/null  
exec bucle
```

Nota: Puede crear el fichero en el “Editor de textos” de CentOS o utilizar uno de los editores presentados en clases anteriores.

Asigne al fichero bucle permisos de ejecución. (Si no funciona, pruebe a asignar todos los permisos al fichero)

Ejecutar con: ***./bucle***

En otra terminal ejecute ***top***

Verifique los datos del proceso yes, consumo CPU, memoria, prioridad.

Digite ***r***

Coloque el número 10 y presione Enter dos veces

Verifique nuevamente los datos del proceso yes, consumo CPU, memoria, prioridad.

### 3.23 Cambio de prioridades a los procesos

El comando ***nice*** es usado para iniciar un proceso con una prioridad definida por el usuario.

***nice [-n <value>] <original command>***

value: entre -20 y +19

```
$ nice -n 10 my_program &
[1] 4862
$ ps l
F  UID  PID  PRI    NI  VSZ  ...  COMMAND
0   500 4372   9     0  4860  ...   -bash
0   500 4862  15    10  3612  ...   my_program
0   500 4863  21     0  1556  ...   ps l
```

Figura 14. Ejemplo de uso del comando nice

El comando renice

El comando **renice** es usado para cambiar la prioridad de un proceso que está corriendo.

**renice <new priority> <PID>**

```
$ renice 15 4862
4862: old priority 10, new priority 15
$ ps l
F  UID  PID  PRI    NI  VSZ  ...  COMMAND
0   500 4372   9     0  4860  ...   -bash
0   500 4862  22    15  3612  ...   my_program
0   500 4868  21     0  1556  ...   ps l
```

Figura 15. Ejemplo de uso del comando renice

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

**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.**

#### 4. INFORME

Realizar un informe sobre las preguntas en rojo.