Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

Administración de Sistemas Operativos

Taller 6:
Procesos

Ingeniería en Software y Tecnologías Emergentes
2023-2

Nombre del Estudiante
Reyes Udasco Richelle Nadine
Matrícula del Estudiante
1288433

Docente
M.I Alma Leticia Palacios Guerrero

Fecha de entrega: 25 de octubre de 2023.

Taller 6: Procesos

Introducción

Los sistemas operativos multitarea, como UNIX, permiten que varios procesos se ejecuten simultáneamente. Un proceso es una unidad de ejecución en un sistema operativo que representa un programa en ejecución. Cada proceso tiene su propio espacio de direcciones, su propio conjunto de recursos y su propio estado.

Los procesos en UNIX se presentan en diversas formas, desde los cruciales procesos del sistema que gestionan tareas fundamentales, los cuales son los primeras en ejecutarse, hasta los procesos de usuario que representan las acciones deseadas por quienes interactúan con el sistema. A través de actividades, se explorarán sus atributos y estados cambiantes, así como su relación con otros procesos.

La asignación de prioridades y el control sobre los recursos se pueden manipular a través del comando "nice," que otorga a los administradores y usuarios la posibilidad de controlar el comportamiento de los procesos.

El comando "ps" se convierte en la herramienta principal para poder observar el comportamiento y estado de los procesos en UNIX, ofreciendo una variedad de modalidades y opciones que permiten una monitorización detallada de las actividades en ejecución.

UNIX, como sistema operativo, también permite realizar múltiples tareas simultáneamente, gracias a la ejecución en segundo plano y primer plano, lo que amplía la productividad sin bloquear la línea de comandos.

La gestión de procesos culmina con la capacidad de terminarlos de manera controlada, liberando recursos valiosos. Y, en medio de todo esto, se pueden presentar las señales, las cuales son como mensajes que permiten la comunicación y el control entre procesos.

En este taller, se explorarán los conceptos básicos de los procesos en UNIX, incluyendo sus tipos, atributos, prioridades, estados y comandos asociados. La comprensión de los procesos es esencial para la gestión de tareas, la optimización del rendimiento del sistema y la solución de problemas en entornos UNIX.

Desarrollo

Como primer paso, se abrió una sesión de trabajo en el servidor (con la dirección proporcionada) y se ingresó el usuario y contraseña para tener acceso.

```
(base) macuser@Nadines-MacBook-Pro ~ % ssh richelle@148.231.130.237
richelle@148.231.130.237's password:
Welcome to Limesurvey, TurnKey GNU/Linux 17.1 (Debian 11/Bullseye)

  System information for Wed Oct 18 22:46:15 2023 (UTC+0000)

    System load:  0.00               Memory usage:  2.4%
    Processes:    135                Swap usage:    0.0%
    Usage of /:   2.7% of 212.43GB   IP address for eth0: 148.231.130.237

  TKLBAM (Backup and Migration):  NOT INITIALIZED

    To initialize TKLBAM, run the "tklbam-init" command to link this
    system to your TurnKey Hub account. For details see the man page or
    go to:

        https://www.turnkeylinux.org/tklbam


    For Advanced commandline config run:    confconsole

  For more info see: https://www.turnkeylinux.org/docs/confconsole

Linux limesurvey 5.10.0-25-amd64 #1 SMP Debian 5.10.191-1 (2023-08-16) x86_64
You have mail.
Last login: Wed Oct 18 22:40:33 2023 from 10.32.221.160
```

A continuación, se realizaron las siguientes actividades:

1. *Genere un listado completo de todos los procesos que están en el sistema y muestre la información completa de todos los que se empezaron a ejecutar el 7 de septiembre en una sola línea.*

   $ ps -fea | grep “Sep07”

```
$ ps -fea | grep "Sep07"
richelle  133223  132833  0 22:51 pts/24   00:00:00 grep Sep07
```

Como se pudo observar, no se presentaron muchos procesos, por lo que se decidió ejecutar otro comando, utilizando otra fecha. Por medio de este comando, se pueden observar más procesos en el sistema.

$ ps -fea | grep “Oct01”

```
$ ps -fea | grep "Oct01"
root           1       0  0 Oct01 ?        00:03:15 /sbin/init
root           2       0  0 Oct01 ?        00:00:00 [kthreadd]
root           3       2  0 Oct01 ?        00:00:00 [rcu_gp]
root           4       2  0 Oct01 ?        00:00:00 [rcu_par_gp]
root           6       2  0 Oct01 ?        00:00:00 [kworker/0:0H-events_highpri]
root           8       2  0 Oct01 ?        00:00:00 [mm_percpu_wq]
root           9       2  0 Oct01 ?        00:00:00 [rcu_tasks_rude_]
root          10       2  0 Oct01 ?        00:00:00 [rcu_tasks_trace]
root          11       2  0 Oct01 ?        00:00:00 [ksoftirqd/0]
root          12       2  0 Oct01 ?        00:00:47 [rcu_sched]
root          13       2  0 Oct01 ?        00:00:06 [migration/0]
root          15       2  0 Oct01 ?        00:00:00 [cpuhp/0]
root          16       2  0 Oct01 ?        00:00:00 [cpuhp/1]
root          17       2  0 Oct01 ?        00:00:18 [migration/1]
root          18       2  0 Oct01 ?        00:00:00 [ksoftirqd/1]
root          20       2  0 Oct01 ?        00:00:00 [kworker/1:0H-kblockd]
root          21       2  0 Oct01 ?        00:00:00 [cpuhp/2]
root          22       2  0 Oct01 ?        00:00:15 [migration/2]
root          23       2  0 Oct01 ?        00:00:00 [ksoftirqd/2]
root          25       2  0 Oct01 ?        00:00:00 [kworker/2:0H-events_highpri]
root          26       2  0 Oct01 ?        00:00:00 [cpuhp/3]
root          27       2  0 Oct01 ?        00:00:07 [migration/3]
root          28       2  0 Oct01 ?        00:00:00 [ksoftirqd/3]
root          30       2  0 Oct01 ?        00:00:00 [kworker/3:0H-kblockd]
root          35       2  0 Oct01 ?        00:00:00 [kdevtmpfs]
root          36       2  0 Oct01 ?        00:00:00 [netns]
root          37       2  0 Oct01 ?        00:00:00 [kauditd]
root          38       2  0 Oct01 ?        00:00:00 [khungtaskd]
root          39       2  0 Oct01 ?        00:00:00 [oom_reaper]
root          40       2  0 Oct01 ?        00:00:00 [writeback]
root          41       2  0 Oct01 ?        00:00:58 [kcompactd0]
root          42       2  0 Oct01 ?        00:00:00 [ksmd]
root          43       2  0 Oct01 ?        00:00:16 [khugepaged]
root          62       2  0 Oct01 ?        00:00:00 [kintegrityd]
root          63       2  0 Oct01 ?        00:00:00 [kblockd]
root          64       2  0 Oct01 ?        00:00:00 [blkcg_punt_bio]
root          65       2  0 Oct01 ?        00:00:00 [edac-poller]
root          66       2  0 Oct01 ?        00:00:00 [devfreq_wq]
root          68       2  0 Oct01 ?        00:00:03 [kworker/2:1H-kblockd]
```

2. *¿Qué están haciendo los procesos que actualmente está ejecutando maestro. (Comando)*

   `$ ps -u alma`

```
$ ps -u alma
    PID TTY          TIME CMD
```

Como no se mostró ningún proceso, se decidió implementar el mismo comando para otro usuario del mismo maestro.

```
$ ps -u lety
    PID TTY          TIME CMD
  68202 pts/0    00:00:00 sh
  70919 pts/0    00:00:00 bash
```

3. *Genere un listado con el número de proceso, número del proceso padre, comando en ejecución y prioridad de tres de sus compañeros.*

   `$ ps -u alain,galindo,emmanuel -o pid,ppid,cmd,pri`

```
$ ps -u alain,galindo,emmanuel -o pid,ppid,cmd,pri
    PID    PPID CMD                         PRI
 106283  106264 sshd: alain@pts/1             19
 106284  106283 -sh                           19
 106674  106284 top                           19
 106957  106284 vi prac7                      19
 106959  106284 vi                            19
 107050  106284 cat                           17
 107052  106284 cat                           19
 107063  106284 vi                            19
 107160  107141 sshd: galindo@pts/2           19
 107161  107160 -sh                           19
 107239  107161 vim                           19
 107495  107476 sshd: emmanuel@pts/5          19
 107496  107495 -sh                           19
```

4. *Explique la diferencia entre las opciones de ps e,f,l y j*

Primeramente, para observar cómo se comportan estas modalidades del comando ps, se ejecutaron y se compararon los resultados mostrados en pantalla.

$ ps -e

```
$ ps -e
    PID TTY          TIME CMD
      1 ?        00:03:02 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-events_highpri
      8 ?        00:00:00 mm_percpu_wq
      9 ?        00:00:00 rcu_tasks_rude_
     10 ?        00:00:00 rcu_tasks_trace
     11 ?        00:00:00 ksoftirqd/0
     12 ?        00:00:36 rcu_sched
     13 ?        00:00:05 migration/0
     15 ?        00:00:00 cpuhp/0
     16 ?        00:00:00 cpuhp/1
     17 ?        00:00:13 migration/1
     18 ?        00:00:00 ksoftirqd/1
     20 ?        00:00:00 kworker/1:0H-kblockd
     21 ?        00:00:00 cpuhp/2
     22 ?        00:00:11 migration/2
     23 ?        00:00:00 ksoftirqd/2
     25 ?        00:00:00 kworker/2:0H-events_highpri
     26 ?        00:00:00 cpuhp/3
     27 ?        00:00:05 migration/3
     28 ?        00:00:00 ksoftirqd/3
```

$ ps -f

```
$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
richelle  102132  102131  0 18:50 pts/1    00:00:00 -sh
richelle  102145  102132  0 18:53 pts/1    00:00:00 ps -f
```

$ ps -l

```
$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1001  102132  102131  0  80   0 -   620 -      pts/1    00:00:00 sh
0 R  1001  102146  102132  0  80   0 -  1669 -      pts/1    00:00:00 ps
```

$ ps -j

```
$ ps -j
    PID    PGID     SID TTY          TIME CMD
 102132  102132  102132 pts/1    00:00:00 sh
 102147  102147  102132 pts/1    00:00:00 ps
```

A partir de estas ejecuciones se puede observar que las opciones e, f, l y j del comando ps se pueden utilizar para obtener información detallada sobre los procesos que se están ejecutando en un sistema UNIX, y las diferencias radican en los procesos que muestran y lo detallado que describe a cada uno, es decir, a los atributos que muestran y el formato que utilizan.

En este caso, la opción -e proporciona un vista general de todos los procesos activos, incluyendo los del sistema y de los usuarios, mientras que las opciones -f, -l y -j se pueden diferenciar por el formato en que muestran los procesos.

La opción -f proporciona una vista detallada de los procesos en ejecución (con los estados UID, PID, PPID, C, STIME, TTY, TIME y CMD); la opción -l muestra una vista más detallada en formato largo de los procesos (además de los estados de la opción anterior, muestran F, S, PRI, NI, ADDR, SZ y WCHAN); por último, la opción -j muestra la información comenzando por el PID (incluyen también el PGID, SID, TTY, TIME y CMD)

5. *Explique la diferencia entre las opciones de ps a y u*

De la misma manera que en la instrucción anterior, para observar el comportamiento de estas modalidades del comando ps, se ejecutaron y se compararon los resultados.

$ ps -a

```
$ ps -a
    PID TTY          TIME CMD
    672 tty8     00:00:00 confconsole
   1377 tty8     00:00:00 dialog
  68201 pts/0    00:00:00 su
  68202 pts/0    00:00:00 sh
  70919 pts/0    00:00:00 bash
 102148 pts/1    00:00:00 ps
```

$ ps -u

```
$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
richelle  102132  0.0  0.0   2480  1660 pts/1    Ss   18:50   0:00 -sh
richelle  102149  0.0  0.0   6756  2924 pts/1    R+   18:54   0:00 ps -u
```

A partir de esto, se puede observar que la diferencia entre las opciones -a y -u está en los usuarios de los que se muestran los procesos. Mientras que -a muestra los procesos de todos los usuarios en el sistema, la opción -u muestra los procesos de usuario específicos (si no se indica un parámetro, se muestran los procesos del usuario activo que ejecuta el comando).

6. *Explique qué es lo que hace la opción de ps t y u*

Nuevamente, con el fin de observar el comportamiento de estas modalidades del comando ps, se ejecutaron en el Terminal.

$ ps -t

```
$ ps -t
    PID TTY      STAT   TIME COMMAND
 102132 pts/1    Ss     0:00 -sh
 102150 pts/1    R+     0:00 ps -t
```

La opción -t muestra los procesos que se asocian a una terminal y es útil para identificar los procesos que están utilizando la interfaz de usuario.

$ ps -u

```
$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
richelle  102132  0.0  0.0   2480  1660 pts/1    Ss   18:50   0:00 -sh
richelle  102151  0.0  0.0   6756  2868 pts/1    R+   18:54   0:00 ps -u
```

La opción -u lista los procesos que pertenecen a los usuarios indicados como parámetros (si no se indica, se muestran los del usuario actual).

7. *Si tiene dos sesiones de ssh abiertas con el mismo user name ¿qué procesos muestra al ejecutar ps?*

En primer lugar, se abrió otra sesión de ssh con el mismo username para observar lo que ocurría. Posteriormente se ejecuta el siguiente comando para mostrar que sí se encuentran ambas sesiones activas:

$ who

```
$ who
root     pts/0        Oct 12 03:22 (148.231.169.208)
richelle pts/1        Oct 18 22:40 (10.32.221.160)
richelle pts/2        Oct 18 22:46 (10.32.221.160)
```

Al ejecutar ps, solamente se muestran los procesos propios de la sesión actual, es decir, solamente a la sesión que se inició.

$ ps

```
$ ps
    PID TTY          TIME CMD
 103038 pts/1    00:00:00 sh
 103081 pts/1    00:00:00 ps
```

Si se quiere observar todos los procesos del usuario, de ambas sesiones activas, se puede utilizar el comando siguiente:

$ ps -u richelle

```
$ ps -u richelle
    PID TTY          TIME CMD
 103037 ?        00:00:00 sshd
 103038 pts/1    00:00:00 sh
 103069 ?        00:00:00 sshd
 103070 pts/2    00:00:00 sh
 103084 pts/1    00:00:00 ps
```

8. *¿Qué opción de ps debería de usar para ver todos los procesos de un usuario?*

$ ps -u richelle

```
$ ps -u richelle
    PID TTY          TIME CMD
 103037 ?        00:00:00 sshd
 103038 pts/1    00:00:00 sh
 103069 ?        00:00:00 sshd
 103070 pts/2    00:00:00 sh
 103084 pts/1    00:00:00 ps
```

$ ps -u alain

```
$ ps -u alain
    PID TTY          TIME CMD
 106283 ?        00:00:00 sshd
 106284 pts/1    00:00:00 sh
 106674 pts/1    00:00:03 top
 106957 pts/1    00:00:00 vi
 106959 pts/1    00:00:00 vi
 107050 pts/1    00:00:00 cat
 107052 pts/1    00:00:00 cat
 107063 pts/1    00:00:00 vi
```

9. *¿Cómo identifico a los procesos que el usuario está ejecutando en cada terminal?*

Como primera opción, se puede ejecutar el comando ps con el usuario para observar el nombre del terminal de control para el proceso por medio del estado **TTY**. Cada sesión se muestra con un valor diferente.

```
$ ps -u richelle
```

```
$ ps –u richelle
    PID TTY          TIME CMD
 103037 ?        00:00:00 sshd
 103038 pts/1    00:00:00 sh
 103069 ?        00:00:00 sshd
 103070 pts/2    00:00:00 sh
 103084 pts/1    00:00:00 ps
```

Como segunda opción, se puede utilizar el comando siguiente para indicar específicamente el nombre del usuario y número de sesión de la terminal para listar los procesos a esa terminal. Si se omiten los parámetros, se muestran los procesos del terminal de la sesión actual.

```
$ ps –t
    PID TTY      STAT   TIME COMMAND
 102132 pts/1    Ss     0:00 –sh
 102150 pts/1    R+     0:00 ps –t
```

10. *¿Cuál es el significado de TODAS las columnas de formato que maneja ps –o? (Sólo las que no están explicadas en este material).*

Comando	Significado
user	Nombre del usuario que inició el proceso.
ruser	Nombre real del usuario que inició el proceso.
group	Nombre del grupo al que pertenece el proceso.
rgroup	Nombre real del grupo al que pertenece el proceso.
ruid	Identificador de usuario real del usuario que inició el proceso.
gid	Identificador de grupo del grupo al que pertenece el proceso.
rgid	Identificador de grupo real del grupo al que pertenece el proceso.

pgid	Identificador del grupo de procesos.
sid	Identificador de sesión.
taskid	Identificador de tarea
opri	Prioridad original del proceso.
pcpu	Porcentaje de tiempo de CPU que el proceso ha utilizado.
pmem	Porcentaje de memoria física que el proceso ha utilizado.
vsz	Tamaño virtual del proceso en bytes.
rss	Tamaño de la memoria real del proceso en bytes.
osz	Tamaño de la memoria de pila en bytes.
nice	Prioridad nice del proceso.
class	Clase del proceso.
etime	Tiempo total que el proceso ha estado en ejecución en modo usuario.
stime	Tiempo total que el proceso ha estado en ejecución en modo kernel.
f	Bandera de estado del proceso.
c	Tipo de CPU en el que se está ejecutando el proceso.
lwp	Número de hilos ligeros del proceso.
nlwp	Número máximo de hilos ligeros del proceso.
psr	Registro de estado del proceso.
fname	Nombre del archivo ejecutable que se está ejecutando.
args	Argumentos del comando que se está ejecutando.
projid	Identificador del proyecto al que pertenece el proceso.
project	Nombre del proyecto al que pertenece el proceso.
pset	Identificador del conjunto de procesos al que pertenece el proceso.

11. *Ejecute dos comandos en background (los que quiera).*

$ vi&

```
$ vi&
```

$ cat >new&

```
$ cat >new&
```

12. *Ejecute el comando cat >lista, ¿Qué prioridad tiene asignada?*

$ cat >lista

```
$ cat >lista
```

Con sesión 2 se ejecuta:

$ ps -u richelle -o pid,ppid,cmd,pri

```
$ ps -u richelle -o pid,ppid,cmd,pri
    PID    PPID CMD                         PRI
 132829  132808 sshd: richelle@pts/24         19
 132833  132829 -sh                           19
 133560  133521 sshd: richelle@pts/26         19
 133561  133560 -sh                           19
 133718  132833 vi                           19
 133719  132833 cat                          19
 133741  132833 cat                          19
 133782  133561 ps -u richelle -o pid,ppid,   19
```

13. *Mate el proceso anterior.*

Con sesión 2 se ejecuta:

$ kill -KILL 133741

```
$ kill -KILL 133741
```

En la sesión 1 se muestra:

```
$ cat >lista
Killed
```

14. *Vuelva a ejecutar cat >lista pero con menor prioridad.*

$ nice -10 cat >lista

```
$ nice -10 cat >lista
```

15. *¿Qué prioridad le fue asignada?*

`$ ps -u richelle -o pid,ppid,cmd,pri`

```
$ ps -u richelle -o pid,ppid,cmd,pri
    PID    PPID CMD                         PRI
 132829  132808 sshd: richelle@pts/24         19
 132833  132829 -sh                          19
 133560  133521 sshd: richelle@pts/26         19
 133561  133560 -sh                          19
 133718  132833 vi                           19
 133719  132833 cat                          19
 133910  132833 cat                           9
 133912  133561 ps -u richelle -o pid,ppid,  19
```

Se asignó la prioridad: 9.

16. *Una vez más ejecute cat>lista, pero ahora en el background.*

`$ cat >lista&`

```
$ cat>lista&
```

17. *¿Cuál es su prioridad ahora?*

`$ ps -u richelle -o pid,ppid,cmd,pri`

```
$ ps -u richelle -o pid,ppid,cmd,pri
    PID    PPID CMD                         PRI
 132829  132808 sshd: richelle@pts/24         19
 132833  132829 -sh                          19
 133560  133521 sshd: richelle@pts/26         19
 133561  133560 -sh                          19
 133718  132833 vi                           19
 133719  132833 cat                          19
 133910  132833 cat                           9
 133924  132833 cat                          19
 133926  133561 ps -u richelle -o pid,ppid,  19
```

Se asigna la misma prioridad: 19.

18. *Verifique que el comando en background esté en la lista de procesos.*

`$ ps -u richelle -o pid,ppid,cmd,pri`

```
$ ps -u richelle -o pid,ppid,cmd,pri
    PID    PPID CMD                         PRI
 132829  132808 sshd: richelle@pts/24         19
 132833  132829 -sh                          19
 133560  133521 sshd: richelle@pts/26         19
 133561  133560 -sh                          19
 133718  132833 vi                           19
 133719  132833 cat                          19
 133910  132833 cat                           9
 133924  132833 cat                          19
 133926  133561 ps -u richelle -o pid,ppid,  19
```

Se puede observar que sí se encuentra en la lista de procesos.

19. *Verifique que el comando en background esté en la lista de tareas (jobs).*

$ jobs

```
$ jobs
[4] + Stopped (tty input)        cat 1>lista
[3] - Stopped                    nice -1 cat 1>lista
[2]   Stopped (tty input)        cat 1>new
[1]   Stopped (tty output)       vi
```

20. *Pase una de las tareas al foreground (use el número de tarea)*

$ fg %2

```
$ fg %2
cat 1>new
```

$ fg %4

```
$ fg %4
cat 1>lista
```

21. *Pase la otra tarea al foreground, pero ahora use el número de PID.*

$ fg 133924

```
$ fg 133924
-sh: 21: fg: No such job: 133924
```

Se puede observar que no es posible, a pesar de ejecutarlo varias veces y utilizando la sintaxis correcta.

22. *Envíe otro comando al background.*

$ vi&

```
$ vi&
```

23. *Finalice este proceso.*

Para finalizar un proceso, se necesita el PID. Para obtener este identificador, se ejecuta el comando siguiente en la sesión 2:

$ ps -u

```
$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
richelle  132833  0.0  0.0   2480  1756 pts/24   Ss+  22:47   0:00 -sh
richelle  133561  0.0  0.0   2480  1668 pts/26   Ss   22:54   0:00 -sh
richelle  133718  0.0  0.0   4936  3368 pts/24   T    22:56   0:00 vi
richelle  133719  0.0  0.0   2528   568 pts/24   T    22:56   0:00 cat
richelle  133910  0.0  0.0   2528   508 pts/24   TN   22:59   0:00 cat
richelle  133924  0.0  0.0   2528   568 pts/24   T    23:00   0:00 cat
richelle  134129  0.0  0.0   4936  3444 pts/24   T    23:04   0:00 vi
richelle  134140  0.0  0.0   4936  3344 pts/24   T    23:04   0:00 vi
richelle  134141  0.0  0.0   4936  3368 pts/24   T    23:04   0:00 vi
richelle  134167  0.0  0.0   6756  2940 pts/26   R+   23:04   0:00 ps -u
```

Para finalizar un proceso, se ejecuta el siguiente comando:

`$ kill -KILL 134141`

```
$ kill -KILL 134141
```

En la sesión 1 se muestra:

```
$ vi&
[5] + Stopped (tty output)       vi
```

Conclusiones

A través de este Taller dedicado a la gestión de procesos en UNIX, se adquirió una mejor comprensión sobre uno de los elementos fundamentales del sistema operativo. Por medio de ejercicios prácticos donde se implementaron diferentes comandos, se lograron identificar las utilidades de los diferentes conceptos que se involucran en el tema.

Comenzando con la generación de listados de los diferentes procesos del sistema, descubrimos cómo el comando ps nos puede brindar información detallada sobre ellos, de tal manera que se pudimos buscar procesos e identificar los atributos de cada uno. Además de esto, la ejecución de sus diferentes opciones logramos distinguir entre las diferentes funcionalidades y saber qué seleccionar cuando se necesite extraer cierta información de los procesos.

Por otra parte, exploramos la jerarquía de proceso y logramos ejecutar procesos en primer y segundo plano, lo cual nos permitió comprender más la estructura y las relaciones que existen entre los procesos en el sistema.

Esta gestión nos permitió comprender cómo tener más control sobre los procesos que se ejecutan en el sistema, proporcionándonos la confianza necesaria para tomar decisiones informadas ante diversas situaciones que puedan requerir de estos comandos.

Referencias

1. Oracle Corporation. (2010). *Guía avanzada del usuario. Capítulo 5: Contraseñas, procesos y almacenamiento en disco.* Documentación de Oracle. https://docs.oracle.com/cd/E19620-01/805-7644/6j76klop8/index.html
2. Palacios, A. L. y Pérez, F. (s/f). *MANUAL DE PRÁCTICAS TALLER DE SISTEMA OPERATIVO UNIX.* Universidad Autónoma de Baja California. [Documento PDF]
3. Robbins, A. (2005). Unix in a Nutshell. *"O'Reilly Media, Inc."*. http://www.ceri.memphis.edu/people/rsmalley/ESCI7205_misc_files/OReilly.Unix.in.a.Nutshell.pdf