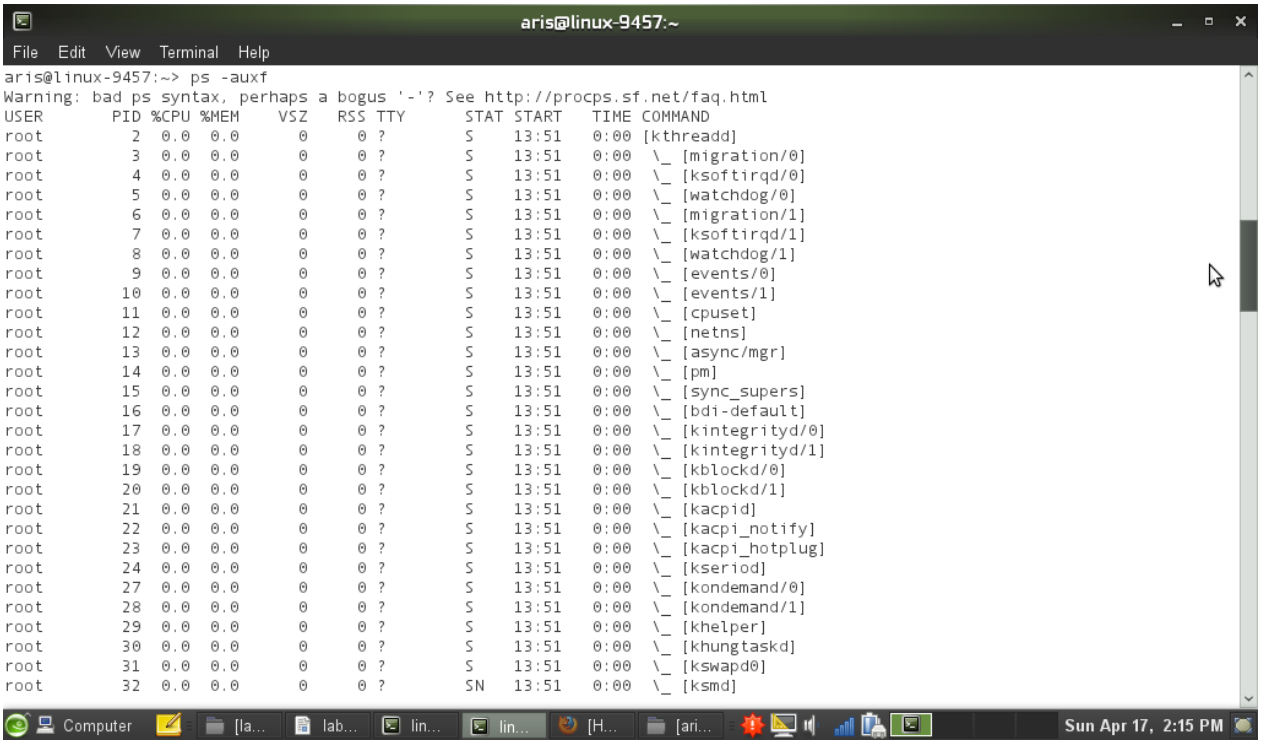
Universidad Tecnológica de Panamá
Sistemas Operativos I
Experiencia Práctica 4

Profesora Aris Castillo de Valencia

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.

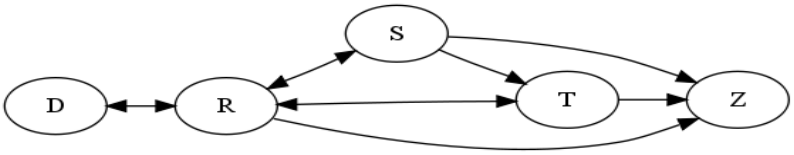


Qué significa cada una de las columnas de la salida del comando ps?
User – identifica al usuario que generó el proceso.
PID – indica el número que identifica al proceso o el Process ID.
%CPU – Indica el porcentaje de CPU que está consumiendo el proceso.

%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.

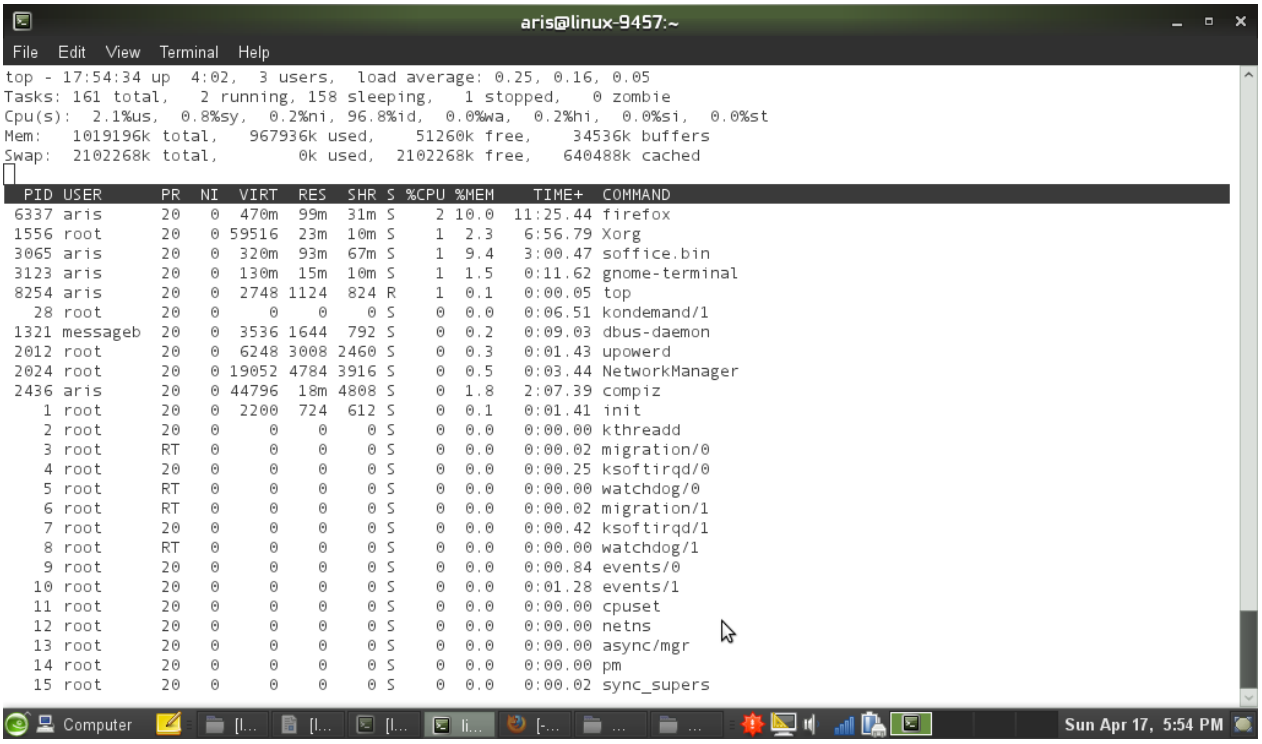
Qué significan los estados de los procesos en Linux?
El comando ps puede mostrar los procesos en Linux en uno de estados según se muestra en la figura siguiente.



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.

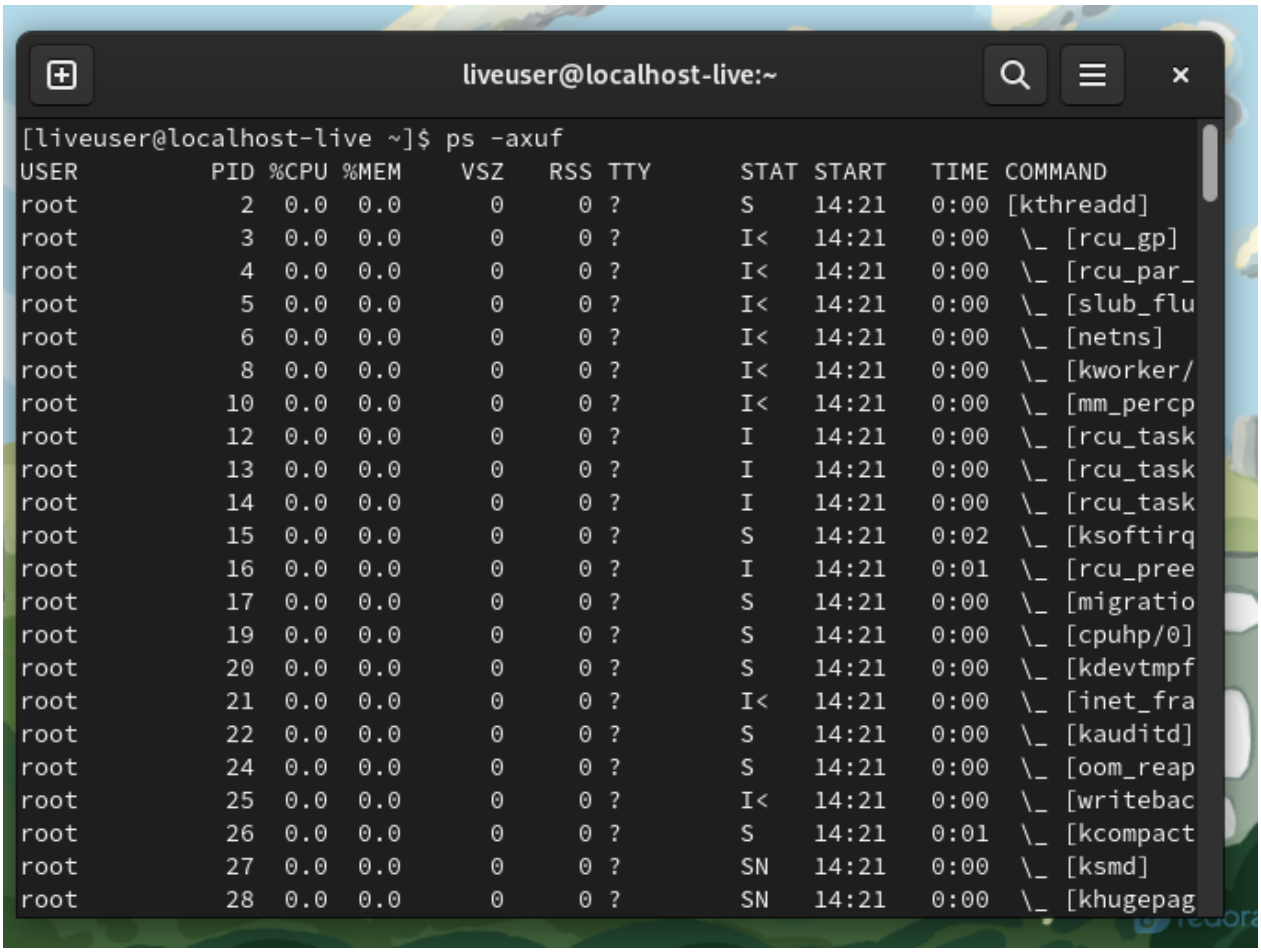


El comando **top** muestra más información que ps. Veamos lo que significan los distintos elementos mostrados:

- 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

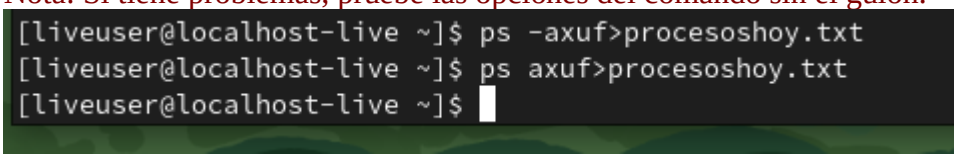
1. Abrir la terminal de consola y escriba el comando **ps –axuf**. Describa la salida.



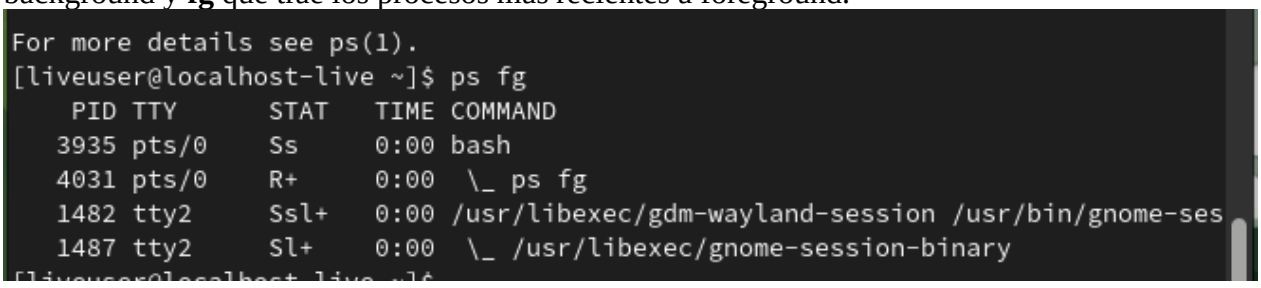
2. Ahora abra otra terminal de texto. Nuevamente escriba el comando y redireccione a un archivo llamado procesoshoy.txt. Este archivo contendrá la salida del comando ps. Después de ejecutar el comando, revise el archivo.

   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.

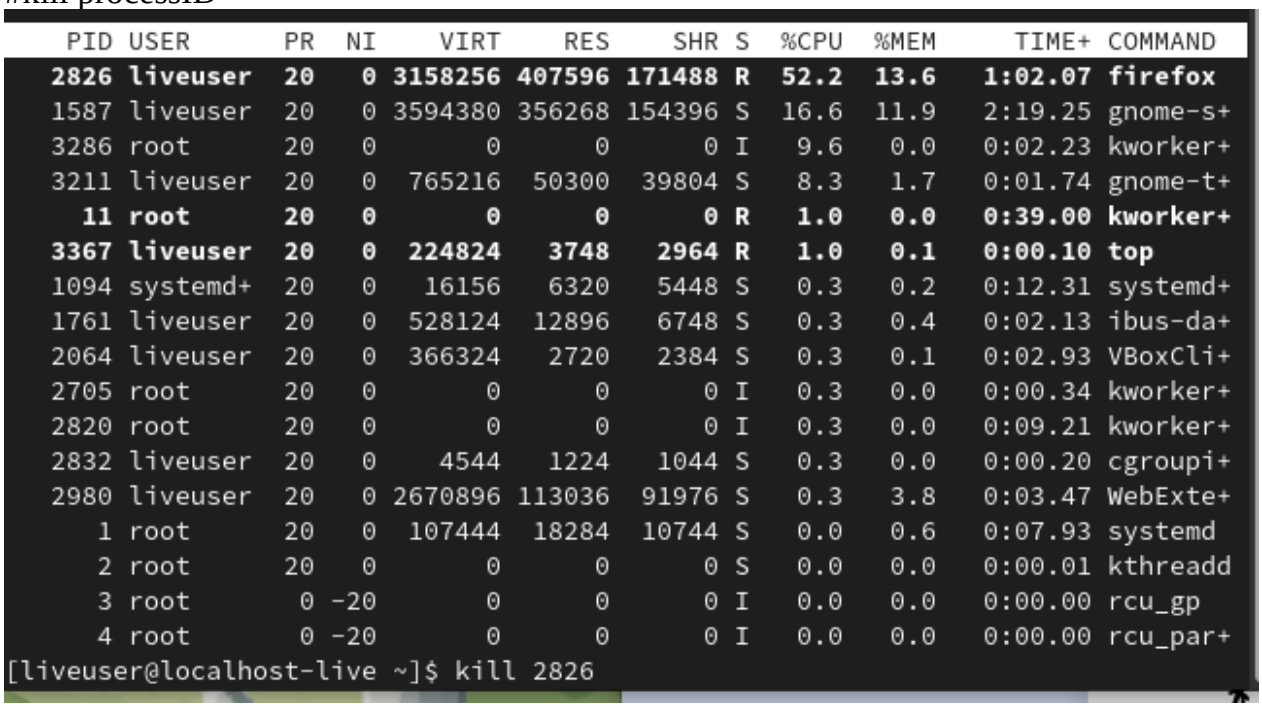


Cómo puedo saber solamente la cantidad de memoria disponible?

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

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int 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());

return 0;
}

~
~
```

Paso 2: Compile el programa: gcc - o *programaejecutable programa.c*

```
[liveuser@localhost-live ~]$ gcc archivo.c -o archivoejecutable
```

Paso 3: Ejecutar el programa: ./programa

```
[liveuser@localhost-live ~]$ ./archivoejecutable
El padre del proceso 3926 es 3645
El padre del proceso 3927 es 1473
El padre del proceso 3929 es 1473
El padre del proceso 3930 es 1473
El padre del proceso 3933 es 1473
El padre del proceso 3934 es 1473
El padre del proceso 3935 es 1473
El padre del proceso 3936 es 1473
El padre del proceso 3937 es 1473
El padre del proceso 3938 es 1473
El padre del proceso 3940 es 3938
[liveuser@localhost-live ~]$
```

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 manterner 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.
2. 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.

1 Compilar múltiples archivos fuente	gcc archivo1.c archivo2.c -o programa
2 Especificar ubicación de las bibliotecas	gcc archivo.c -o programa -L/ruta/de/las/bibliotecas -lname_biblioteca
3 Habilitar optimizaciones	gcc archivo.c -o programa -O2
4 Incluir archivos de encabezado adicionales	gcc archivo.c -o programa -I/ruta/del/directorio
5 Generar información de depuración	gcc archivo.c -o programa -g

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.
4. En qué situaciones específicas considera que serían útiles los comandos utilizados?

Para abrir diferentes tipos de programas escritos en algún lenguaje, modificarlos y ejecutarlos.

5. Qué dificultades encontró durante el desarrollo del laboratorio?

Simplemente dedicarle un tiempo para entenderlo, leyendo la guía y buscando en internet, no creo que haya sido difícil.

6. Qué mejoraría de esta experiencia de laboratorio?

Me gusta como está planteado, quizás agregar algún ejemplo mas gráfico, mas imágenes etc.

Referencias:

1. Kernighan, B. y Pike, R. El Entorno de programación Unix. Prentice Hall.
2. Linux Shortcuts and Commands: http://www.unixguide.net/linux/linuxshortcuts.shtml
3. TTY Desmystified: http://www.linusakesson.net/programming/tty/index.php
4. 20 Linux System Monitoring tools every sysadmin should know: http://www.cyberciti.biz/tips/top-linux-monitoring-tools.html