Proyecto 1

Sistemas Operativos

Segundo Semestre 2022, Prof. Cecilia Hernández

Fecha Inicio: Miércoles 31 de Agosto. 2022.

Fecha Entrega: Viernes 23 de Septiembre 2022 (23:59 hrs).

Trabajo en grupo: Integrado con 3 estudiantes. Una entrega por grupo.

Entrega: Archivo comprimido con readme.txt software e informe.

1. Objetivos
   - Introducir a los estudiantes en el manejo de procesos concurrentes en Unix, creación, ejecución y terminación usando llamadas a sistemas fork(), exec() y wait(). Además el uso de otras llamadas a sistema como signals y comunicación entre procesos usando pipes.
2. Metodología: Trabajo en grupo de 3 estudiantes.
3. Descripción

   Desarrollo de un intérprete de comandos simple en Linux (shell). La shell a implementar será similar a las disponibles actualmente en Linux. A continuación se detalla lo requerido en su implementación.

   a) Parte 1 (3.0 puntos.)
      1) La shell debe proporcionar un prompt, lo que identifica el modo de espera de comandos de la shell.
      2) Debe leer un comando desde teclado y parsear la entrada para identificar el comando y sus argumentos (debe soportar un número indeterminado de argumentos).
      3) Debe ejecutar el comando ingresado en un proceso concurrente, para lo cual debe usar el llamado a sistema fork() y algunas de las variantes de exec(). Los comandos a soportar son ejecutados en foreground, es decir, la shell ejecuta y espera por el término de su ejecución antes de imprimir el promtp para esperar por el siguiente comando.
      4) Si se presiona "enter" sin previo comando, la shell simplemente imprime nuevamente el prompt.
      5) Su shell debe soportar comandos que se comunican mediante pipes, es decir debe soportar comandos del tipo mishell:$ ps -aux | sort -nr -k 4 | head -10.
      6) Su shell además debe soportar el comando exit para terminar.
      7) Debe poder continuar si es que un comando ingresado no existe, proporcionando el error correspondiente.

b) Segunda parte (3.0 puntos)

1) Su shell debe crear un comando que permita a un proceso comunicarse con otro proceso usando señales. En particular, debe usar las señales SIGUSR1 y SIGUSR2. Para ello puede inspeccionar con el comando *man* la llamada a sistema signal() o bien la llamada a sistema sigaction() y kill().
2) Su shell debe reconocer el Control-C y continuar preguntar si desea continuar o no ejecución de shell.
3) Debe agregar otro comando que le permita reportar el uso de recursos en términos de tiempo de ejecución de usuario y sistema; y además de uso de memoria utilizada mediante la medida de maximum resident set. Su comando debe poder comenzar el registro de uso de recursos para los comandos a ejecutar en su shell y almacernarla en un archivo log. Y además debe soportar la detención del registro. Para su implementación se pide usar la función *getrusage*. Para ver como funciona use el comando *man* para determinar como usar la función. A continuación se muestra un ejemplo de como usar este comando.

```
mishell:$ usorecursos start recursos.log
mishell:$ ls -l
mishell:$ ./miprog
mishell:$ usorecursos stop
mishell:$ cat recursos.log
comando tuser  tsys   maxrss
ls -l   1ms    0.5ms  10KB
miprog  10s    2ms    249KB
```