Proyecto 1 de Sistemas Operativos. Implementación de una shell

Cecilia Hernández

August 29, 2019

Fecha inicio: Jueves, 29 de Agosto, 2019.
Fecha entrega: Jueves, 12 de Septiembre, 2019 (a mediodia).

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 señales, tuberías y algunos efectos con colores.

2. Metodología: Trabajo en grupo de 2 alumnos.

3. Descripción

   Desarrollo de un intérprete de comandos simple en Linux (shell). La shell a implementar será similar a las disponibles actualmente en Linux. El desarrollo constará de dos partes de implementación y una de prueba y análisis.

   Primera Parte

   (a) La shell debe proporcionar un prompt, lo que identifica el modo de espera de comandos de la shell.

   (b) Debe leer un comando desde teclado y parsear la entrada para identificar el comando y sus argumentos (debe soportar al menos 3 argumentos).

   (c) 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.

   (d) Si se presiona “enter“ sin previo comando, la shell simplemente imprime nuevamente el prompt.

   (e) Debe proporcionar la salida de la ejecución del comando en distinto color.

   (f) Debe almacenar la fecha y hora del inicio de la sesión de la shell y todos los comandos ejecutados junto con las fechas y horas cuando estos comandos fueron ejecutados en dicha sesión. Esta información debe ser reportada en caso sea solicitada con un comando especial denominado *info*.

      - Buscar si un comando fue ejecutado en la sesión y si es así desplegar la fecha y hora en que fue ejecutado.
      - Desplegar por pantalla los comandos ejecutados en la sesión como una lista.
      - Elegir un comando de la lista y volver a ejecutarlo.
      - Además debe escribir esta información en un archivo log de la sesión denominado *.mishell.log*.

La correcta implementación de esta etapa lo hace merecedor de un 50% de la nota.

A continuación se presenta un ejemplo de ejecución.

```
shellABC$> ps -la
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R  1000  5991  3808  0  80   0 -  3660 -      pts/6    00:00:00 ps
0 S  1000 18243 25660  2  80   0 - 601233 se_sys pts/1   00:42:42 firefox
0 S  1000 18303 18243  1  80   0 - 525750 se_sys pts/1   00:23:13 Web Content
0 S  1000 18342 18243  0  80   0 - 437472 se_sys pts/1   00:00:35 WebExtensions
0 S  1000 18668 18243  1  80   0 - 589258 se_sys pts/1   00:31:44 Web Content
0 S  1000 19707 18243  1  80   0 - 489629 se_sys pts/1   00:22:03 Web Content
0 S  1000 31387 18243  1  80   0 - 480918 se_sys pts/1   00:08:48 Web Content
```

Su shell debe considerar comandos que contengan pipes, es decir, del tipo

ps -la | *grep PRI*, para ello debe utilizar las llamadas a sistema pipe(), dup() o dup2(), and close(). Su shell debe soportar múltiples pipes en un comando dado, por ejemplo: *ls -l* | *grep file* | *wc -l*

Segunda Parte

La segunda parte de su shell debe usar señales usando la llamado a sistema "sigaction" para permitir la comunicación con su shell desde el exterior de su shell.

La idea es que su shell tenga un comando especial llamado *miocio arg1 arg2* que tiene dos argumentos, el primero es un tiempo en millisegundos y el segundo es una opción de seguir o terminar. Por ejemplo *miocio 100ms si*. Cuando la shell reconozca este comando la shell debe manejar la señal SIGUSR1 y comenzar a dormir por el tiempo especificado. Luego, un proceso externo a la shell la puede despertar usando el comando *kill -SIGUSR1 pidshell*. Tal señal debe ser capturada por su shell y reportar el tiempo que lleva durmiendo, y si $arg2 = si$ se despierta y queda activa para continuar escuchando por comandos; y si $arg2 = no$ la shell continua durmiendo por el tiempo restante dado al especificado en $arg1$. Si el tiempo dado en $arg1$ expira la shell debe terminar.

Si la shell es interrumpida con un control C esta debe terminar.

La correcta implementación de esta etapa lo hace merecedor de un 50% de la nota.

Llamadas a sistema y funciones

Su proyecto incluye el uso de las llamadas fork(), variantes exec(), variantes wait(), dup2(), pipe(), close(), sigaction(), nanosleep(). Además puede utilizar funciones de C/C++ para hacer el parsing.

Para manejar colores mirar el siguiente sitio:

`https://www.cyberciti.biz/faq/bash-shell-change-the-color-of-my-shell-prompt-under-linux-or-unix/`

Evaluación

La evaluación final consiste en una nota por el funcionamiento (50%), una por interrogación del grupo (40%) y un informe (10%). El informe debe contener introducción, desarrollo y conclusiones. El desarrollo debe indicar como fueron utilizadas las llamadas a sistema, las estructuras de datos utilizadas en el desarrollo. Esta parte corresponde al 10% de la nota final.