Tarea 1

Sistemas Operativos

Segundo Semestre 2025, Prof. Cecilia Hernández y Juan Felipe González

Fecha Inicio: Viernes 29 de agosto, 2025.

Fecha Entrega: Viernes 26 de septiembre, 2025 (23:59 hrs).

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: Integrado con 3 o 4 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. Debe entregar un informe en pdf con la descripción de lo desarrollado, el cual debe incluir el link a un repositorio donde se aloja su proyecto. El repositorio debe incluir un Readme que describa como compilar y ejecutar comandos.

   I Parte 1. Su shell debe: (3.0 puntos.) (KPI 1.1, 1,2)
   1) Proporcionar un prompt, lo que identifica el modo de espera de comandos de la shell. Identifique si necesita alguna llamada a sistema para cumplir con este requerimiento. (KPI 1.1)
   2) Leer un comando desde teclado y parsear la entrada para identificar el comando y sus argumentos (debe soportar un número indeterminado de argumentos). Identifique si necesita alguna llamada a sistema para cumplir con este requerimiento. (KPI 1.1)
   3) Soportar el comando exit para terminar. Identifique si necesita alguna llamada a sistema para cumplir con este requerimiento. (KPI 1.1)
   4) Poder continuar si es que un comando ingresado no existe, proporcionando el error correspondiente. Identifique si necesita alguna llamada a sistema para cumplir con este requerimiento. (KPI 1.1)
   5) Si se presiona “enter” sin previo comando, la shell simplemente imprime nuevamente el prompt. Identifique si necesita alguna llamada a sistema para cumplir con este requerimiento. (KPI 1.1)
   6) 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. (KPI 1.2)
   7) Soportar comandos que se comunican mediante pipes, es decir debe soportar comandos del tipo mishell:$ ps -aux | sort -nr -k 4 | head -20. (KPI 1.2)

II Segunda parte (3.0 puntos) (KPI 1.3, 2.1, 6.3, 6.4)
Su shell debe implementar un comando personalizado llamado *miprof*, el cual permite ejecutar cualquier comando o programa y capturar la información respecto al tiempo de ejecución en tiempos de usuario, sistema y real mas información acerca del peak de memoria máxima residente (Maximum Resident Set). Además debe opcionalmente almacenar en forma persistente los comandos y resultados obtenidos. El comando consiste en lo siguiente:

1) miprof [ejec|ejecsave archivo] comando args: Solo ejecuta y despliega por pantalla o ejecuta y guarda el comando ejecutado por miprof en archivo proporcionado. Si ejecuta miprof con el mismo nombre de archivo y otro comando, el comando ejecutado y sus resultados deben agregarse al final del archivo. (KPI 1.3, 2.1)
2) Use miprof con el comando sort para diseñar e interpretar los resultados usando distintos largos de archivos de texto. (KPI 6.3, 6.4)
3) miprof ejecutar maxtiempo comando args: Solo ejecuta el miprof sin guardar en forma persistente la salida. La ejecución del comando tiene un tiempo máximo de ejecución, en caso no se cumpla este tiempo la shell debe terminar el proceso asociado al comando. (KPI 1.3, 2.1)

4. Ayuda: Lista de llamadas a sistema que puede necesitar para su implementación. Para ver la forma en que se usa cada una de ellas puede usar el comando man en la consola. Por ejemplo, *$man fork*.

- Procesos y ejecución
  - `fork`
  - `variantes de exec`
  - `_exit`
  - `setpgid`
- Tuberías y descriptores
  - `pipe`
  - `pipe2`
  - `dup2`
  - `dup3`
  - `close`
  - `fcntl`
- Archivos y redirecciones
  - `open`
  - `creat`
- Espera, estado y uso de recursos
  - `waitpid`
  - `wait4`
  - `getrusage`
- Tiempo / timers
  - `clock_gettime`
- Señales, timeout y control de procesos
  - `sigaction`
  - `sigprocmask`
  - `kill`
  - `setitimer`
  - `alarm`

Información de KPIs

1. KPI 1.1 Identificar los componentes de un problema complejo y los mecanismos relevantes.
2. KPI 1.2 Formular y expresar problemas complejos dentro del campo de la ingeniería informática utilizando herramientas matemáticas o computacionales.
3. KPI 1.3 Resolver o identificar soluciones para problemas complejos aplicando herramientas de ingeniería.
4. KPI 2.1 Concebir, diseñar, implementar u operar sistemas o procesos organizacionales relevantes para la sociedad desde la ingeniería informática y disciplinas afines, para crear soluciones combinando las herramientas de la ciencia y la tecnología.
5. KPI 6.3 Diseñar y realizar experimentos o pruebas de calidad de software.
6. KPI 6.4 Analizar e interpretar resultados.