Programación multiproceso - Programación multihilo

Aclaración: En este examen se evalúan contenidos específicos de programación concurrente y paralela. Cada ejercicio **debe** ser hecho con las técnicas requeridas, en caso contrario el ejercicio será puntuado con **0 puntos**.

Ejemplo: Escribe un programa que cree un proceso hijo con fork, el hijo escribirá “Hola mundo” y el padre esperará a que finalice el proceso hijo.
Entrega: El alumno entrega un programa que escribe “Hola mundo” (Similar al que se hace el primer día de clase de programación).
Resultado: **0 puntos**.

Penalizaciones:
Cada línea de código con mala sangría o sangría inconsistente restará **0.2 punto**.
Cada número mágico en el código restará **0.2 puntos**. Los valores constante deben ser CONSTANTES (escritas con nombre mayúsculas)

Ejercicios **2,5 puntos** cada uno.

1. Tabla de multiplicar. fork y for
2. Ordenador. fork y pipes
3. Analizador procesos. Java y comandos
4. Carrera. Java threads

Ejercicio 1

Crea un programa que cree 10 procesos hijos en un bucle **for**. Cada proceso hijo escribirá por pantalla la tabla de multiplicar de su número (el primer hijo del 1, el segundo del 2, etc. desde el 1 al 10). El programa principal debe crear los hijos y esperar a que finalicen en un bucle **for**. Los hijos escriben la tabla con la función **write**.

Creación del los hijos. 0.5 puntos

Cada hijo sabe qué número debe escribir. 0.5 puntos

Cada hijo escribe la tabla de su número. 1 puntos

El padre espera a todos sus hijos. 0.5 puntos

Ejercicio 2

Crea un programa que cree dos pipes, después haga fork. El padre enviará 3 números aleatorios al proceso hijo a través de uno de los pipes. El proceso hijo los ordenará y los escribirá en un pipe de vuelta al padre ordenados. El padre muestra la información y espera a que finalice el hijo.

Envío y recepción de información padre-hijo 1 punto

Ordenar números 0.75 puntos

Recibir información hijo-padre, mostrar y esperar finalización 0.75 puntos

Ejercicio 3

Realiza un programa Java que reciba como parámetro el nombre de un usuario. El programa Java ejecutará un “ps aux” (comando linux), leerá la información del comando, mostrará la cabecera y todos los procesos de ese usuario que consumen más del 0.0 de memoria.

Ejemplo:

java Examen.Memoria jorge

Salida:

USER PID %CPU **%MEM** VSZ RSS TTY STAT START TIME COMMAND

jorge 13288 0.0 **0.5** 1184771532 91308 ? Sl 11:08 0:00 /opt/google/chrome/chrome --type=renderer --enable-crashpad --crashpad-handler-pid=6901

jorge 13302 0.0 **0.4** 1184730548 64572 ? Sl 11:08 0:00 /opt/google/chrome/chrome --type=renderer --enable-crashpad –crashpad-handler-pid=6901

Ejecución y lectura de la salida de ps. 0.5 puntos

Pintar cabecera. 0.5 puntos

Filtrado de información. 1.5 puntos

Ejercicio 4

Crea una clase Corredor que implemente Runnable. El corredor tiene un dorsal y debe recorrer 1000 metros. Su forma de correr es que avanza un número de metros aleatorio entre 50 y 100, luego espera de forma aleatoria entre 50 y 100 milisegundos. Cada vez que avanza escribe “Dorsal: 13 Avanzo: 57 metros” (los metros cambiarán con cada avance). Cuando llega a 1000 metros escribe “He cruzado la meta y soy el dorsal 13”. Tras llegar a meta el Thread finaliza.

Para que la carrera sea justa los Threads antes de comenzar a correr esperan a la señal de salida en un objeto. El Thread principal crea 10 corredores cada uno con su dorsal, una vez creados notifica que pueden salir a correr y espera a los 10 threads finalicen. La creación y la espera deben realizarse con un bucle.

Thread corredor con dorsal. 1 punto

Sincronización de salida. 1 punto

Thread principal. 0.5 puntos.

Nombre:

Apellido: