PRIMER PROYECTO DE SISTEMAS OPERATIVOS (25057)
ENTREGA: 17/05

1. **Procesos e hilos**

El objetivo es familiarizarse con los llamados al sistema relativos a la creación y administración de procesos y hilos usando la librería POSIX u otra para creación de procesos e hilos en Linux usando C.

A. **Una de las herramientas más básicas para la creación de procesos es fork (bifurcación). Esta herramienta permite que un proceso pida al sistema operativo que se cree una copia exacta de él mismo, para que luego cada uno de los procesos (el padre y su hijo) sigan el curso normal de ejecución. Si bien puede parecer de poca utilidad tener dos procesos que realicen lo mismo, el uso de identificadores que permiten diferenciar entre el proceso padre y su hijo da a este mecanismo un gran potencial.**

a) Estudiar los llamados al sistema fork(), getpid(), getppid() y kill().
b) Si un proceso queda huérfano, ¿Qué proceso lo adopta y cuál es su PID?
c) Escribir un programa en C o C++ llamado replicaMuerte.c El proceso crea un hijo que cada 5 segundos comprueba si su padre está vivo. En caso que sí, se replica a sí mismo. En caso que no, se suicida. El primer padre por su parte queda esperando que se presione una tecla tras lo cual mata a su primer hijo y termina.
AYUDA: El resultado debe presentar un crecimiento exponencial en la cantidad de procesos al comienzo y tras presionar un botón o tecla, un decrecimiento exponencial.

B. **Una herramienta más ligera respecto a la creación de procesos es el thread (hilo). El hilo se distingue de los procesos por ser más liviano y compartir parte de la memoria, por lo cual se requiere más cuidado al respecto.**

a) Estudiar los llamados al sistema pthread create(), pthread exit() y pthread join().
b) Escribir un programa multihilo en C llamado promedioDehilos.c El padre crea un arreglo aleatorio de 100 enteros (entre 0 y 1000) y da acceso a fragmentos de 25 elementos a 4 hilos que promedian el subarreglo y retornan este valor. Al terminar todos los hilos, el padre despliega por pantalla la media de los elementos del arreglo promediando el retorno de las 4 hilos.
c) Solucione el punto anterior, utilizando semáforos de la siguiente manera: cada vez que un hilo tenga el resultado, escribirá en una variable compartida el resultado y terminará. El padre leerá este resultado y lo desplegará en pantalla. El resultado es que el padre despliega el resultado cada vez que el hijo termina.

2. **Interrupciones**

Implementar un simulador de interrupciones que reciba una petición de interrupción de un dispositivo e interrumpa el flujo de la ejecución del programa. Para esto se debe implementar:

a. Simulador de interrupciones
b. CPU: una aplicación que actúa como si fuera el CPU que va ejecutando una secuencia de instrucciones

c. Driver / Dispositivo: una aplicación que maneja un recurso

Dinámica

1. CPU va escribiendo en un archivo las instrucciones que ejecuta, que son
    a. i = 1
    b. While (true)
        i. Instrucción i (escribir i: hora)
        ii. Loop de 10 segundos
        iii. i++
2. El simulador de interrupciones está ejecutándose infinitamente
    a. while (true)
3. El driver / dispositivo comienza a ejecutarse y en un momento dado, le envía una interrupción al simulador de interrupciones y este interrumpe el flujo del CPU y luego de 30 segundos el CPU retoma su flujo normal.
4. Esto significa que durante 30 segundos el CPU no va a hacer nada y esto se debe notar en el archivo de salida.

EVALUACIÓN

- **1er Proyecto**
    - **Evaluación Grupal (Informe final 5% + promedio de las notas de evaluación individual 5%) Total: 10%**
    - **Corrida y defensa (Evaluación individual) Total: 10%**
- **ENTREGABLES**
    - **Informe sintético que tiene párrafos sobre**
        - **creación de procesos hijos y muerte de los padres**
        - **creación de hilos y ejecución sin semáforo, ejecución con semáforos**
        - **comunicación entre procesos**
    - **Código fuente comentado**
    - **Proyecto ejecutándose**