Laboratorio 3 (y Lección 2):
Procesos, Hilos Posix, Condición de Carrera

Objetivos

- Crear y ejecutar procesos con uno o más hilos de ejecución.
- Observar similitudes y diferencias entre procesos e hilos.
- Conocer los **thread**, la función **fork()** y el concepto de Condición de Carrera.

Duración: 1 sesión de teoría y 1 sesión de laboratorio

Al trabajar en Linux, pueden crear proyectos de Eclipse para cada programa a continuación, o pueden compilar los programas directamente desde la consola. Si trabajan en Windows, usarán Cygwin.

Primera Parte: Múltiples Procesos, Múltiples Hilos

1. Compilen los programas **L3_Hello.cpp** y **L3_World.cpp**. Asegúrense de nombrar los ejecutables **L3_Hello** y **L3_World**, respectivamente. En general, el nombre del ejecutable no tiene que ser igual al nombre del archivo fuente. Sin embargo, esta vez nos servirá tener esos nombres (inciso 3 y otros más).
2. Ejecutarán los dos programas en una misma consola. Esencialmente, deberán mandar el primer proceso al "trasfondo". Para ello, usarán el operador & (como al invocar Eclipse desde la terminal). Ejecuten el primer programa así:
   ./L3_Hello &
   Eso permitirá seguir usando la consola, por lo que podrán ejecutar el segundo programa:
   ./L3_World
   Noten que el primer programa imprime los mensajes a la terminal al mismo tiempo que ustedes ingresan el comando para el segundo programa. Tengan cuidado de ingresar correctamente.

3. Abran una segunda consola (o ventana de Cygwin). Mientras los programas anteriores estén ejecutándose, ingresen el comando: **ps ax**
   A continuación, ingresen el comando: **ps ax | grep L3_**
   Tomen una captura de pantalla de la consola para cada caso. Describan las diferencias que observen. ¿Qué hace el operador | (*pipe*)? ¿Qué hace el comando grep?

4. Encuentren el identificador de cada uno de los procesos (PID) correspondientes a sus programas. Para cada proceso, utilicen el comando: **kill PID**, donde PID es un número (el identificador correspondiente). ¿Qué ocurre?

5. A continuación, estudien el programa **L3_Hilos_Ej1.cpp**.
   Atiendan las explicaciones del catedrático sobre este programa y el concepto de hilos.

6. Intenten compilar el programa en la consola directamente, siguiendo la sintaxis descrita en prácticas anteriores. En Linux, deberían observar un problema al compilar (en Cygwin no se verá el problema) ¿Cuál es el problema?
   Este programa utiliza la librería **thread**, la cual permite el uso de hilos. Al utilizar esta librería no es necesario indicarla explícitamente al invocar el compilador.
   Una vez compilado, corran el programa en una de las consolas.

7. Ahora estudien, compilen y ejecuten el programa **L3_Hilos_Ej2.cpp** en una de las consolas.
   ¿Cómo se compara lo observado aquí con lo observado al ejecutar los programas de los incisos 1 y 2?

8. En la otra consola, ingresen el comando **ps ax | grep L3_ .** ¿Qué observan ahora? Tomen una captura de pantalla de la consola.

9. En Eclipse, también se debe indicar el uso de librerías como pthread. En Linux, creen un nuevo proyecto y copien el código de **L3_Hilos_Ej2.cpp.** Intenten compilar el programa. ¿Algún problema?
   Corran el programa creado en Eclipse en una de las consolas que usaron antes (Linux)

Segunda Parte: Función fork(), Contexto entre Procesos y entre Hilos

Como tarea, ustedes investigaron la función **fork()** y el concepto de Condición de Carrera. En esta parte del laboratorio, ustedes correrán algunos programas que usan/ejemplifican la función y el concepto.

1. Estudien el código **L3_fork_Ej1.cpp**. Brevemente mencionen las similitudes que encuentren con los programas de la Primera Parte.
   El instructor explicará el funcionamiento de la función fork().

2. En una terminal, compilen y ejecuten el programa (nómbrenlo **L3_fork_Ej1**). Sólo deben invocar el programa una vez:
   ./L3_fork_Ej1
   ¿Cómo se compara la salida de este programa con la de los programas de la Primera Parte?

3. En la otra terminal, ingresen el comando **ps ax | grep L3_** . ¿Qué observan ahora? ¿Cuántos procesos L3_fork_Ej1 distintos observan? Tomen una captura de pantalla.

A continuación, compararán los programas **L3_fork_contexto** y **L3_pthread_contexto**.

4. Estudien ambos códigos. Anoten las similitudes y diferencias entre ambos.

5. Compilen los programas. Nuevamente, por conveniencia, nombren los ejecutables **L3_fork_contexto** y **L3_pthread_contexto**, respectivamente.
6. Ejecuten ambos programas, en consolas distintas. ¿Observan alguna diferencia en lo desplegado por ambos programas? Expliquen las diferencias, si las hay.

7. En una tercera consola, ingresen **ps ax | grep L3_** ¿Qué observan ahora? Tomen una captura de pantalla.

8. Según lo investigado en la tarea 1 y lo observado en la práctica, ¿Qué entienden por Condición de Carrera? ¿Qué problemas puede haber cuando múltiples procesos/hilos pueden acceder/modificar un recurso compartido?

Tercera Parte: Más Ejemplos y Ejercicios

1. Estudien, compilen y ejecuten el programa **L3_varios_forks**. Antes que termine la ejecución, corran **ps ax | grep L3_** en otra consola. ¿Hace sentido lo que observan? Expliquen claramente. Tomen una captura de pantalla.

2. Modifiquen el programa **L3_Hilos_Ej2.cpp** para que tenga cuatro hilos en lugar de dos. Para ello, deberán definir más variables tipo **std::thread**, y llamar a la función ***nombre de variable* = std::thread();** más veces en el main. Recuerden que el main es el primer hilo. El segundo hilo debe quedar igual que en **L3_Hilos_Ej2.cpp**.
    a. El tercer hilo debe ejecutar el mismo código **My_Thread**, pero debe desplegar un mensaje distinto. Es decir, al llamar a la función **std::thread()**, el tercer argumento debe ser el mismo que para el segundo hilo. Pero el cuarto argumento debe ser un string distinto.
    b. El cuarto hilo debe ejecutar un código distinto a los hilos 2 y 3. Para ello, deben definir una función adicional. La estructura de esta función será similar a **My_Thread**. Pueden nombrar a la función como ustedes deseen (siempre y cuando no sea una palabra reservada del lenguaje C++). Al llamar a **std::thread()** en el main, el tercer argumento debe tener el nombre de su función nueva.
    Este cuarto hilo no recibirá ningún argumento. Por lo tanto, el último argumento de **std::thread()** en el main debe ser **NULL**.
    Lo único que debe hacer este hilo es "dormir" por 5 segundos, luego desplegar un mensaje cualquiera (por ejemplo, "ELECTRONICA DIGITAL 3, LAB 3, su nombre"), y luego salir (**std::thread::detach()**).

Material de Apoyo

1. Páginas del manual (*man page*) de la función **fork**.