Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Arquitectura de Computadores II

Grupo 1

Taller 1: Hilos

Estudiante:

André Herrera Chacón 2015183074

Profesor:

Ing. Luis Barboza Artavia

Febrero,2020

Hilos en GNU Linux

POSIX Thread Libraries (pthreads)

Biblioteca estándar para el manejo de hilos en C/C++. Esta permite crear nuevos procesos concurrentes, es efectiva en sistemas con multiprocesadores o multinúcleo.Lo anterior aumenta la velocidad en procesamiento paralelo y distribuido. Los hilos son creados definiendo una función con argumentos para ser procesados.

Entre algunas funciones de la biblioteca se encuentran:

- Creación del hilo, terminación, sincronización, calendarización, manejo de datos e interacción en el proceso.
- No se mantiene una lista de hilos creados.
- Los hilos del mismo proceso comparten:
    - Instrucciones.
    - Datos.
    - Manejadores de señales.
    - Directorio actual.
    - Usuario y id grupal.
- Los hilos tienen un único:
    - ID.
    - Registros, Stack pointer.
    - Stack de variables locales, direcciones de retorno.
- Retorna 0 si el resultado es OK

Ejemplos de bibliotecas POSIX

- **Provenzano threads (PT)**: Permite llamadas al sistema“Thread-blocking” y “thread-safe”. Soporta funcionalidades básicas, sincronización de primitivas, atributos de hilos e información específica de cada hilo.
- **FSU_Pthreads (FSUT)**:Creada para los sistemas operativos: Solaris 2.x, SCO UNIX, FreeBSD, Linux and DOS. Maneja la prioridad por calendarización y las funcionalidades básicas de POSIX.
- **PC threads (PCT)**: Esta es una biblioteca POSIX a nivel que incluye selección, lectura y escritura sin bloqueo. Esta biblioteca tiene parámetros configurables en tiempo de ejecución, como el intervalo de interrupción del reloj, la política predeterminada de programación de subprocesos, el tamaño predeterminado de la pila de subprocesos y el intervalo de sondeo de entradas y salidas.
- **CLthreads (CLT)**: Threads es una biblioteca compatible con POSIX a nivel de kernel sobre Linux. Threads utiliza la llamada al sistema de clonación para aprovechar al máximo los sistemas multiprocesador.

- **LinuxThreads (LT)**: Proporciona subprocesos a nivel de núcleo: se crean utilizando la llamada al sistema de clonación, y toda la programación se realiza en el núcleo. Este enfoque puede aprovechar al máximo los sistemas multiprocesador. También da como resultado una biblioteca de subprocesos más simple y robusta, especialmente para bloquear llamadas al sistema.

Solaris Threads

Esta biblioteca tiene similitudes con POSIX ya que sus funcionalidades principales son idénticas. Lo anterior permite crear aplicaciones que mezclen ambos manejadores de hilos sin generar muchos problemas. Al igual que el anterior únicamente se debe agregar a biblioteca y el include respectivo (-lthread, <thread.h) en el programa.

Las funcionalidades únicas de este programa son:

- Suspender la ejecución del hilo.
- Reanudar el hilo suspendido.
- Especificar el nivel de concurrencia del hilo.
- Obtener el nivel de concurrencia.

Mutex en Multiprogramación

Mutex significa “exclusión mutua”, en general, estas variables, se encargan de la sincronización entre procesos y la protección de los datos compartidos cuando se producen lecturas y escrituras al mismo tiempo.

Estas variables se comportan como un bloqueo para resguardar el acceso a un recurso. Un ejemplo de esto es cuando por medio de la biblioteca Pthreads un hilo intenta acceder a un dato, sólo este puede bloquear una variable mutex por el tiempo dado por el sistema. Ningún otro subproceso puede poseer ese mutex hasta que el subproceso propietario desbloquee ese mutex. Los hilos deben "turnarse" para acceder a los datos protegidos.

Los mutexes se pueden usar para prevenir condiciones de "carrera", esta condición se da cuando dos o más procesos intentan tener acceso a un recurso del sistema al mismo tiempo.

Condición de carrera

Esta condición, como se mencionó anteriormente, se da cuando múltiples procesos tratan de acceder a un mismo recurso. El resultado de esto depende de quién se “apropie” del recurso primero y el momento en que lo haga. Esto se da, normalmente, por que la ejecución de tareas en un sistema es incontrolable y por más que se eviten estas situaciones siempre varios procesos van a ocupar un mismo recurso.

En general, se deben sincronizar los procesos-hilos para evitar el problema anterior, obteniendo lo que se le conoce como exclusión mutua en la región crítica del programa donde se comparten recursos.

Algunas formas de solucionar la condición de carrera son:

- Deshabilitar interrupciones: Deshabilitar las interrupciones después de entrar a la región crítica y luego las habilita justo después que sale.
- Variables de candado: Se tiene una variable compartida, cuando un proceso tiene que entrar a a región crítica evalúa el candado.
- Alternancia estricta: Los procesos toman turnos para acceder a la región crítica.
- Peterson: Es una combinación de variables de candado y alternancia estricta. En este se establece de quién es el turno y se modifica la variable de candado o bandera.
- Semáforos: Se utiliza para controlar el acceso a un recurso por parte de un número finito de instancias.
    - Down: Comprueba si el valor es mayor que cero.Si es mayor que cero, disminuye el valor del semáforo y continúa el proceso.Si el valor escero,el proceso es dormido sin completar la operación down por el momento.
    - Up: Incrementa el valor del semáforo. Si uno o más procesos estaban inactivos en ese semáforo, sin poder completar una operación down anterior ,el sistema selecciona uno al alzar.

Respuesta 2

El comportamiento que se ve en el ejercicio demuestra que el hilo A accede al contador primero y en segundo lugar el hilo B. Lo anterior se explica con la exclusión mutua ya que al entrar de primero se apropia del recurso, lo libera, entra B, ejecuta la operación y libera el recurso, todo esto hasta terminar las iteraciones correspondientes.

Bibliografía

Ippolito, G(2004). POSIX thread (pthread) libraries. Recuperado de: https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html

Marshall, D(1999). Threads: Basic Theory and Libraries. Recuperado de: https://users.cs.cf.ac.uk/Dave.Marshall/C/node29.html

National Laboratory.POSIX Threads Programming. Recuperado de: https://computing.llnl.gov/tutorials/pthreads/#Mutexes

Beal, V(s.f). mutex - mutual exclusion object .Recuperado de: https://www.webopedia.com/TERM/M/mutex.html

internetlab(2013).La condición de carrera. Recuperado de: https://www.internetlab.es/post/2548/condicion-de-carrera/

Leitón, J(2019). Notas de clase: Procesos e Hilos en un Sistema Operativo.