Introducción a la librería pthreads en C

Programación de Alto Rendimiento

26 de junio de 2025

¿Qué son los pthreads?

- La biblioteca **pthreads** (POSIX Threads) es una **API estándar** para hilos en C/C++.
- Permite la creación de **múltiples flujos de ejecución concurrentes** dentro de un mismo proceso.
- thread = hilo.
- **Beneficios Clave**:
  - **Aprovechamiento de Multi-núcleos**: Muy efectiva en sistemas con múltiples procesadores o núcleos, mejorando el rendimiento a través del paralelismo.
  - **Menor Sobrecarga**: Requiere menos recursos que la creación de nuevos procesos (‘fork()‘) porque los hilos comparten el espacio de memoria del proceso padre.
  - **Repartición de Recursos**: Todos los hilos dentro de un proceso comparten el mismo espacio de direcciones de memoria.

Conceptos Básicos de Hilos

- Los hilos dentro del mismo proceso **comparten** los siguientes recursos:
  - **Instrucciones del proceso**.
  - La **mayoría de los datos**.
  - **Archivos abiertos** (descriptores).
  - **Señales y manejadores de señales**.
  - Directorio de trabajo actual.
  - ID de usuario y grupo.
- Cada hilo posee atributos **únicos**:
  - **ID de Hilo** (Thread ID).
  - Conjunto de registros y puntero de pila.
  - Pila para variables locales y direcciones de retorno.
  - Máscara de señal.
  - Prioridad.

Creación y Terminación de Hilos

- `pthread_create()`:
  - **Función**: Crea un nuevo hilo y comienza su ejecución, asignándole una función de inicio.
  - **Sintaxis**: `int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void *), void *arg);`.
  - **Parámetros Clave**:
    - `thread`: Puntero para almacenar el ID del nuevo hilo.
    - `start_routine`: Puntero a la función que ejecutará el hilo.
    - `arg`: Argumento (un puntero a 'void') pasado a la función del hilo.
- `pthread_exit()`:
  - **Función**: Termina explícitamente el hilo que la invoca.
  - No afecta la ejecución de otros hilos.
- `pthread_join()`:
  - **Función**: Hace que el hilo llamante **espere** a que un hilo especificado termine su ejecución.
  - **Importancia**: Crucial para sincronizar el hilo principal y los hilos secundarios, evitando que el programa principal finalice prematuramente.

Sincronización de Hilos: Mutexes

- **Mutexes** (Bloqueos de Exclusión Mutua):
  - **Propósito**: Previenen **condiciones de carrera** y aseguran la consistencia de los datos compartidos.
  - **Funcionamiento**: Garantizan **acceso exclusivo** a una región crítica"de memoria por un solo hilo a la vez.
  - **Funciones Comunes**:
    - `pthread_mutex_init()`: Inicializa un mutex.
    - `pthread_mutex_lock()`: Bloquea el mutex; si ya está bloqueado, el hilo llamante se suspende.
    - `pthread_mutex_unlock()`: Desbloquea el mutex.
    - `pthread_mutex_destroy()`: Libera los recursos de un mutex.
  - **Ejemplo de Riesgo**: La operación de incremento (`counter++`) sin protección por mutexes puede llevar a resultados incorrectos si varios hilos la ejecutan concurrentemente.
  - **Advertencia**: Un uso incorrecto (ej. no liberar un mutex) o un orden inadecuado al bloquear múltiples mutexes puede causar **deadlock** (interbloqueo).

Sincronización de Hilos: Variables de Condición

- **Variables de Condición** (`pthread_cond_t`):
  - **Propósito**: Permiten a los hilos suspender su ejecución y liberar el procesador hasta que se cumpla una **condición específica**.
  - **Asociación Obligatoria**: Siempre deben usarse **junto con un mutex** para evitar condiciones de carrera.
  - **Funciones Clave**:
    - `pthread_cond_init()`: Inicializa una variable de condición.
    - `pthread_cond_wait()`: Bloquea el hilo llamante (liberando el mutex asociado) hasta que la condición sea señalada.
    - `pthread_cond_signal()`: Despierta a **un hilo** que esté esperando en la variable de condición.
    - `pthread_cond_broadcast()`: Despierta a **todos los hilos** que estén esperando en la variable de condición.
    - `pthread_cond_destroy()`: Libera los recursos de una variable de condición.
  - **Consideración Importante**: La lógica de las condiciones (sentencias `if`/`while`) debe garantizar que la señal se ejecute si la espera se activa, para evitar deadlocks.

Compilación y Uso

- **Compilación**:
  - Para compilar programas C que utilizan la biblioteca pthreads, es necesario **enlazarla**.
  - Use la bandera `-pthread` o `-lpthread` con el compilador GCC/G++.
  - **Ejemplo**: `gcc mi_programa.c -o mi_programa -pthread`.
- **Ejecución**:
  - Simplemente ejecute el binario compilado: `./mi_programa`.
- **Consideraciones Importantes**:
  - **Código "Thread Safe"**: Las funciones llamadas por los hilos deben ser "thread safe" para manejar correctamente el acceso a variables compartidas.
  - **Depuración**: La depuración de programas multihilo puede ser compleja debido a la naturaleza no determinista de la ejecución concurrente. Herramientas como GDB o DDD pueden ser útiles.