Programación de Sistemas CCPG1008

Federico Domínguez, PhD.

Unidad 6 – Sesión 3: Concurrencia con hilos

Agenda

Definición hilos

POSIX Threads

Demos

Un **hilo** o ***thread*** es un flujo lógico de ejecución dentro de un proceso.

Un **flujo lógico de ejecución** (flujo lógico) es simplemente una secuencia de instrucciones que en conjunto forman un programa o parte de un programa. Típicamente, un proceso contiene un solo flujo lógico.

Sistemas operativos modernos permiten que un proceso pueda mantener más de un flujo lógico de manera concurrente mediante el concepto de **hilos**.

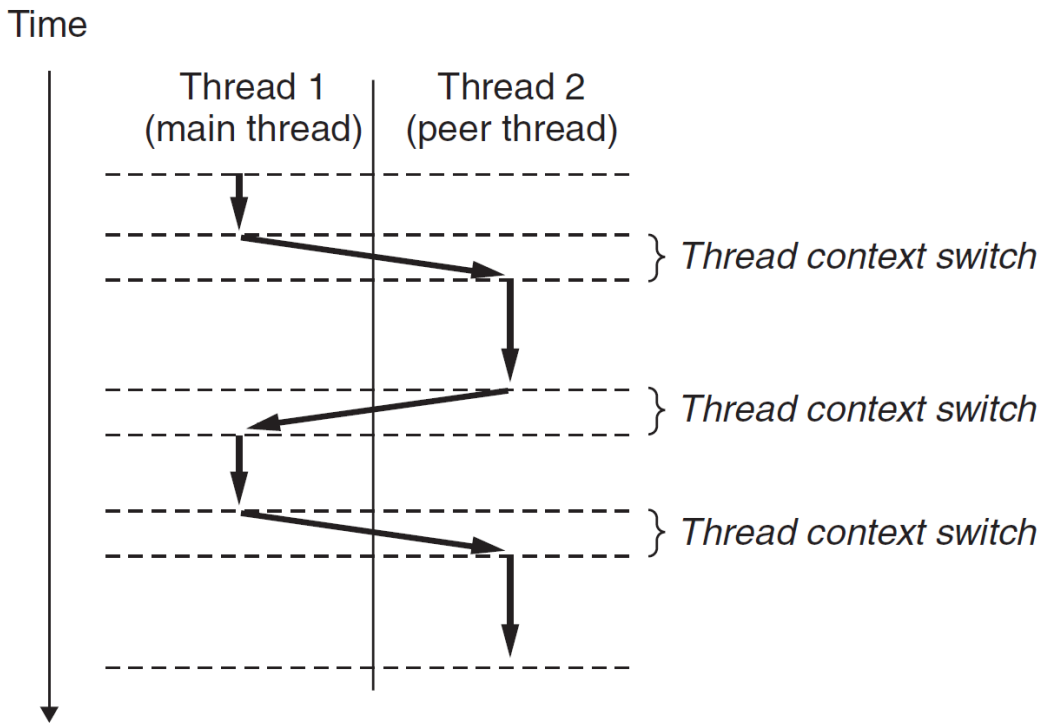
Un hilo es un flujo lógico dentro de un proceso con su propio:

- *Thread ID* (tid)
- Contexto de hilo (no es igual al contexto del proceso)
  - Stack
  - PC
  - y más (registros, puntero al stack, etc…)

Un hilo comparte el espacio de memoria virtual del proceso al cual pertenece.

Un proceso puede tener uno o varios hilos que se ejecutan de manera concurrente.

Un hilo, al igual que un proceso, es agendado por el kernel.



Entre hilos no existen jerarquías similares a padre – hijo, todos los hilos se ejecutan dentro de un proceso con igual “rango”.

Existe el hilo principal o *main thread* el cual se ejecuta primero en un proceso y crea el resto de hilos.

A parte de ser el primer proceso en ejecutarse, no existe diferencia entre el hilo principal y el resto de hilos. Todos los hilos pueden “esperarse” o “destruirse” entre si.

Hilos creados dentro de un mismo proceso son conocidos como *peer threads* y el conjunto de hilos creados por un proceso es un *thread pool*.

POSIX Threads (PThreads) es una interface estándar que implementa hilos en C.

PThreads define aproximadamente 60 funciones para manipular hilos (crear, eliminar, esperar, etc.)

```
#include <pthread.h>
typedef void *(func)(void *);

int pthread_create(pthread_t *tid, pthread_attr_t *attr,
                   func *f, void *arg);
```

Returns: 0 if OK, nonzero on error

```
#include <pthread.h>

pthread_t pthread_self(void);
```

Returns: thread ID of caller

code/conc/hello.c

```
1   #include "csapp.h"
2   void *thread(void *vargp);
3
4   int main()
5   {
6       pthread_t tid;
7       Pthread_create(&tid, NULL, thread, NULL);
8       Pthread_join(tid, NULL);
9       exit(0);
10  }
11
12  void *thread(void *vargp) /* Thread routine */
13  {
14      printf("Hello, world!\n");
15      return NULL;
16  }
```

code/conc/hello.c

Un *pthread* puede terminar de cuatro formas distintas.

1. Implícitamente cuando su función principal retorna.
2. Explícitamente cuando llama a *pthread_exit()*. Si es el hilo principal, *pthread_exit()* espera a todos los hilos y termina el proceso.
3. Algún hilo llama a *exit().* Esto termina todos los hilos en un proceso junto con el proceso.
4. Algún hilo llama a *pthread_cancel(pthread_t tid),* con el *tid* del hilo.

```
#include <pthread.h>

void pthread_exit(void *thread_return);
```

Demo threads

Revisar *share.c* en https://bitbucket.org/fexadom/ejemplo_threads

Similar a *wait()*, podemos esperar a que un hilo termine con *pthread_join()*.

- *pthread_join()* bloquea el hilo hasta que el otro hilo termina.
- A diferencia de *wait()*, *pthread_join()* solo puede esperar a un hilo específico.
- Similar a *wait()*, después de *pthread_join()*, el hilo esperado es recogido y todos sus recursos son eliminados.

```
#include <pthread.h>

int pthread_join(pthread_t tid, void **thread_return);
```

Returns: 0 if OK, nonzero on error

Un hilo, por defecto, debe ser recogido con *pthread_join(),* pero puede convertirse en un hilo "independiente" con ***pthread_detach().***

Un hilo puede ser:

- *joinable*: sus recursos son eliminados cuando es recogido por otro hilo con *pthread_join()*.
- *detach*: sus recursos son eliminados automáticamente por el kernel cuando el hilo termina.

Un hilo en estado *detach* es realmente independiente y no puede ser eliminado por otro hilo.

```
#include <pthread.h>

int pthread_detach(pthread_t tid);
```

Returns: 0 if OK, nonzero on error

Demo threads

Revisar *badcnt.c* en https://bitbucket.org/fexadom/ejemplo_threads

Referencias

Libro guía *Computer Systems: A programmers perspective*. Secciones 12.3

Lectura para la próxima semana

Libro guía *Computer Systems: A programmers perspective*. Secciones 12.4-5