Universidad Católica San Pablo | Departamento de Ciencia de la Computación

Sistemas Operativos 2021-02 (CCOMP7-2)
Práctica 02a: uso de `pthread`

Yván Jesús Túpac Valdivia
Universidad Católica San Pablo, Arequipa – Perú

24 de setiembre de 2021

---

1 Objetivos

- Familiarizarse con el uso de la biblioteca `pthread` (POSIX thread), que es un API para *threads* del Standard IEEE POSIX (1995) en C/C++. Permite manipular *threads* en ambientes multiprocesador. Para ambientes uniprocesador se puede aprovecha mejor las esperas I/O.

2 Material a utilizar

- Ambiente de programación C++ (Cualquier Sistema Operativo)

3 Tarea de Laboratorio

3.1 Crear y destruir `threads`

1. Cuando un programa *multithreads* inicia, él mismo es una *thread* completa ejecutando la función `main()` con un propio ID. Si queremos crear una nueva *thread* dentro de `main()`, se puede usar la función `pthread_create()` como muestra el siguiente código:

```
#include <stdio.h>       /* standard I/O routines                 */
#include <pthread.h>     /* pthread functions and data structures */

/* function to be executed by the new thread */
void* PrintHello(void* data)
{
    int my_data = (int)data;            /* data received by thread */

    printf("Hello from new thread - got %d\n", my_data);
    pthread_exit(NULL);                 /* terminate the thread */
}

/* like any C program, program's execution begins in main */
int main(int argc, char* argv[])
{
    int        rc;              /* return value                          */
    pthread_t  thread_id;    /* thread's ID (just an integer)          */
    int        t = 11;       /* data passed to the new thread          */

    /* create a new thread that will execute 'PrintHello' */
    rc = pthread_create(&thread_id, NULL, PrintHello, (void*)t);
    if(rc)                   /* could not create thread */
    {
        printf("\n ERROR: return code from pthread_create is %d \n", rc);
        return(1);
```

```
    }
    printf("\n Created new thread (%d) ... \n", thread_id);

    pthread_exit(NULL);          /* terminate the thread */
}
```

Revisar el código anterior, que aunque no está ejecutando nada útil, nos ayuda a entender cómo los *threads* funcionan. Haga una ejecución paso a paso y observe cómo el programa se ejecuta.

2. En `main` se declaró una variable `thread_id` de tipo `pthread_t`, la cual es un entero que identifica un `thread` en el sistema.

3. La funcion `pthread_create(&thread_id, NULL, PrintHello, (void*)t);` recibe 4 argumentos: un puntero al ID, atributos del *thread* (NULL indicaría atributos *default*), la función a ejecutar, que puede pasar un bloque de datos arbitrario al nuevo *thread*, mediante el 4to argumento (NULL si no se desea pasar datos). Se retorna 0 si inicia OK o un valor diferente si da error al iniciar. Fuente en

   `https://www.dropbox.com/s/0rco9klibgicz6x/newThread.c?dl=1`

4. Compilar y ejecutar este código, reporte la salida de consola.

3.2 Tomar el ID

- Un *thread* puede obtener su ID propio mediante `pthread_self()` que devuelve un ID:

```
pthread_t tid;
tid = pthread_self();
```

- Modificar el código inicial para que se imprima el ID de ambos *threads* (main y el acabado de crear). Reporte lo impreso en consola, que deberá ser similar a:

```
I am thread 1. Created new thread (?) ...
Hello from new thread 4 - got 11
```

- Modifique el código para que el *thread* `main` pase su identificador al nuevo *thread*. Compile y ejecute, se espera ubna salida similar a:

```
I am thread 1. Created new thread (?) ...
Hello from new thread 4 - got 1
```

3.3 Finalización de `threads`

Existen varias maneras de finalizar un *thread*. La más segura es llamar a `pthread_exit`.

- Modificar el programa incluyendo una llamada a la función `sleep` (pausar el proceso por x milisegundos), para 1 milisegundo, dentro de la función `PrintHello`, justo antes de la impresión que ya existe ahí. Tome en cuenta que antes de C++11, esta función (sleep, Sleep) depende de la plataforma (Windows o Linux/UNIX), verifique la documentación de C++ para su uso ya que es posible que necesite incluir `<Windows.h>` o `<unistd.h>`.

- Además quite o comente en el programa main() la llamada `pthread_exit` que finaliza los *threads*. COmpile, ejecute y reporte lo arrojado emn consola explicando qué está ocurriendo.

- Ahora descomente la llamada `pthread_exit` en main y comente la que está dentro de `PrintHello`, ejecute, reporte lo observado en cosola y explique lo ocurrido.

4 Sobre la entrega

Entregar (vía Moodle) de acuerdo al plazo indicado:

- Un informe en PDF con las respuestas encontradas y las explicaciones solicitadas.
- El código con las últimas modificaciones