# Programando con Hilos

Manuel Almeida V.

April 21, 2020

# Temario

# Introducción

- En los años noventa, con los procesadores de mayor capacidad, se puede implementar una nueva forma de aumentar las capacidades del cómputo.
- Para poder hacer multitareas, es necesario introducir el concepto de subprocesos e hilos.
- Esta nueva herramienta es la de poder generar más de un subproceso llamado thread o hilo.

# Introducción

- En los años noventa, con los procesadores de mayor capacidad, se puede implementar una nueva forma de aumentar las capacidades del cómputo.
- Para poder hacer multitareas, es necesario introducir el concepto de subprocesos e hilos.
- Esta nueva herramienta es la de poder generar más de un subproceso llamado thread o hilo.

# Introducción

- En los años noventa, con los procesadores de mayor capacidad, se puede implementar una nueva forma de aumentar las capacidades del cómputo.
- Para poder hacer multitareas, es necesario introducir el concepto de subprocesos e hilos.
- Esta nueva herramienta es la de poder generar más de un subproceso llamado thread o hilo.

# Introducción.

- En la definición de proceso, se puede incluir los elementos de control de recursos y su ambiente, los cuales heredan los hilos.

## Thread

- Un hilo es un subproceso o proceso ligero, pero que comparte el ambiente del proceso que lo genera.
- Los hilos dentro de un proceso pueden ser programados y ejecutados, independientemente.

# Introducción.

- En la definición de proceso, se puede incluir los elementos de control de recursos y su ambiente, los cuales heredan los hilos.

## Thread

- Un hilo es un subproceso o proceso ligero, pero que comparte el ambiente del proceso que lo genera.
- Los hilos dentro de un proceso pueden ser programados y ejecutados, independientemente.

# Introducción.

- Se puede trabajar con varios hilos

**Multithread**

Es la capacidad de un sistema operativo de mantener varios hilos en ejecución dentro de un mismo proceso.

# Introducción.

## Posix

La IEEE definió como el estándar para la programación de hilos a la API POSIX de UNIX.

# Beneficios de los hilos.

Los beneficios de programar con hilos se derivan de su rendimiento:

- Se tarda menos tiempo en crear un hilo que un proceso.
- Se tarda mucho menos tiempo en terminar un hilo que un proceso.
- La comunicación entre hilos es más eficiente que entre procesos.
- Podemos hacer un tipo de programación paralela con hilos.

# Beneficios de los hilos.

Los beneficios de programar con hilos se derivan de su rendimiento:

- Se tarda menos tiempo en crear un hilo que un proceso.
- Se tarda mucho menos tiempo en terminar un hilo que un proceso.
- La comunicación entre hilos es más eficiente que entre procesos.
- Podemos hacer un tipo de programación paralela con hilos.

# Beneficios de los hilos.

Los beneficios de programar con hilos se derivan de su rendimiento:

- Se tarda menos tiempo en crear un hilo que un proceso.
- Se tarda mucho menos tiempo en terminar un hilo que un proceso.
- La comunicación entre hilos es más eficiente que entre procesos.
- Podemos hacer un tipo de programación paralela con hilos.

# Beneficios de los hilos.

Los beneficios de programar con hilos se derivan de su rendimiento:

- Se tarda menos tiempo en crear un hilo que un proceso.
- Se tarda mucho menos tiempo en terminar un hilo que un proceso.
- La comunicación entre hilos es más eficiente que entre procesos.
- Podemos hacer un tipo de programación paralela con hilos.

# Programando hilos.

Vamos a la parte de programación de un hilo, requiere cuatro párametros, para lanzar un hilo se necesita:

- Declarar la librería pthread.h
- El primer parámetro, es el identificador del hilo.
- El segundo parámetro, como usaremos los atributos por default, será *NULL*.
- una estructura, para pasar parámetros a la función, puede omitirse si no se requiere.
- una función, la que ejecutará el hilo.

# Programando hilos.

Vamos a la parte de programación de un hilo, requiere cuatro párametros, para lanzar un hilo se necesita:

- Declarar la librería pthread.h
- El primer parámetro, es el identificador del hilo.
- El segundo parámetro, como usaremos los atributos por default, será *NULL*.
- una estructura, para pasar parámetros a la función, puede omitirse si no se requiere.
- una función, la que ejecutará el hilo.

# Programando hilos.

Vamos a la parte de programación de un hilo, requiere cuatro párametros, para lanzar un hilo se necesita:

- Declarar la librería pthread.h
- El primer parámetro, es el identificador del hilo.
- El segundo parámetro, como usaremos los atributos por default, será *NULL*.
- una estructura, para pasar parámetros a la función, puede omitirse si no se requiere.
- una función, la que ejecutará el hilo.

# Programando hilos.

Vamos a la parte de programación de un hilo, requiere cuatro párametros, para lanzar un hilo se necesita:

- Declarar la librería pthread.h
- El primer parámetro, es el identificador del hilo.
- El segundo parámetro, como usaremos los atributos por default, será *NULL*.
- una estructura, para pasar parámetros a la función, puede omitirse si no se requiere.
- una función, la que ejecutará el hilo.

# Programando hilos.

Vamos a la parte de programación de un hilo, requiere cuatro párametros, para lanzar un hilo se necesita:

- Declarar la librería pthread.h
- El primer parámetro, es el identificador del hilo.
- El segundo parámetro, como usaremos los atributos por default, será *NULL*.
- una estructura, para pasar parámetros a la función, puede omitirse si no se requiere.
- una función, la que ejecutará el hilo.

# Programando hilos.

- En nuestro primer programa, mostraremos cómo crear un hilo, y cómo interactua con el proceso que lo genero, también llamado hilo master.
- Usando las funciones y definiciones de POSIX threads, en la librería "pthread.h".
- Declaramos el identificador *th*1, como tipo $pthread_t$.
- Usamos *pthread_create*(), para crear e iniciar la ejecución del hilo.
- Los parámetros son: (&*th*1, *NULL*, &*print_xs*, *NULL*), el cuarto parámetro es *NULL* pues no es necesario.

# Programando hilos.

- En nuestro primer programa, mostraremos cómo crear un hilo, y cómo interactua con el proceso que lo genero, también llamado hilo master.
- Usando las funciones y definiciones de POSIX threads, en la librería "pthread.h".
- Declaramos el identificador *th*1, como tipo $pthread_t$.
- Usamos *pthread_create*(), para crear e iniciar la ejecución del hilo.
- Los parámetros son: (&*th*1, *NULL*, &*print_xs*, *NULL*), el cuarto parámetro es *NULL* pues no es necesario.

# Programando hilos.

- En nuestro primer programa, mostraremos cómo crear un hilo, y cómo interactua con el proceso que lo genero, también llamado hilo master.
- Usando las funciones y definiciones de POSIX threads, en la librería "pthread.h".
- Declaramos el identificador *th*1, como tipo $pthread_t$.
- Usamos *pthread_create*(), para crear e iniciar la ejecución del hilo.
- Los parámetros son: (&*th*1, *NULL*, &*print_xs*, *NULL*), el cuarto parámetro es *NULL* pues no es necesario.

# Programando hilos.

- En nuestro primer programa, mostraremos cómo crear un hilo, y cómo interactua con el proceso que lo genero, también llamado hilo master.
- Usando las funciones y definiciones de POSIX threads, en la librería "pthread.h".
- Declaramos el identificador *th*1, como tipo $pthread_t$.
- Usamos *pthread_create*(), para crear e iniciar la ejecución del hilo.
- Los parámetros son: (&*th*1, *NULL*, &*print_xs*, *NULL*), el cuarto parámetro es *NULL* pues no es necesario.

# Programando hilos.

- En nuestro primer programa, mostraremos cómo crear un hilo, y cómo interactua con el proceso que lo genero, también llamado hilo master.
- Usando las funciones y definiciones de POSIX threads, en la librería "pthread.h".
- Declaramos el identificador *th*1, como tipo $pthread_t$.
- Usamos *pthread_create*(), para crear e iniciar la ejecución del hilo.
- Los parámetros son: (&*th*1, *NULL*, &*print_xs*, *NULL*), el cuarto parámetro es *NULL* pues no es necesario.

# Programando hilos.

crea_thread.cpp

```
#include <pthread.h>
#include <cstdio>
#include <iostream>

using namespace std;
//compilar con g++ -o crth.x crea_thread.cpp -lpthread
/* La funcion es prints_xs imprime x's  el parametro por
void* print_xs (void* unused)
{

while (1)
cout<<"x";
return NULL;
}
```

# Programando hilos.

```
/* The main program.  */
int main ()
{
//declaramos thread_id tipo pthread_t
pthread_t th1;
/* Crea un nuevo thread si es el numero 1
 El nuevo  thread llama a la function print_xs  */
pthread_create (&th1, NULL, &print_xs, NULL);
/* si no Print o's continuously to stderr.  */
while (1)
cout<<"o";
return 0;
    pthread_exit(NULL);
}
```

# Programando hilos.

- Se compila con:$g++ -oarch.xnombreprog.cpp -lpthread$

# Programando hilos.

- Observe la pantalla y observe qué está sucediendo.
  La función prints_xs imprime xs el parámetro es unused, por otro lado todo tiempo imprimirá os.
- Las xs las imprime el hilo, pero las os las imprime el proceso con el que lanzamos el hilo, vea cómo se intercalan xs y os.
- En este programa no hemos usado ninguna estructura, en el siguiente programa usaremos la estructura.

# Programando hilos.

- Observe la pantalla y observe qué está sucediendo.
  La función prints_xs imprime xs el parámetro es unused, por otro lado todo tiempo imprimirá os.
- Las xs las imprime el hilo, pero las os las imprime el proceso con el que lanzamos el hilo, vea cómo se intercalan xs y os.
- En este programa no hemos usado ninguna estructura, en el siguiente programa usaremos la estructura.

# Programando hilos.

- Observe la pantalla y observe qué está sucediendo.
  La función prints_xs imprime xs el parámetro es unused, por otro lado todo tiempo imprimirá os.
- Las xs las imprime el hilo, pero las os las imprime el proceso con el que lanzamos el hilo, vea cómo se intercalan xs y os.
- En este programa no hemos usado ninguna estructura, en el siguiente programa usaremos la estructura.

# Programando hilos.

- En este segundo programa, crearemos tres hilos, cada uno imprimirá un caracter un cierto número de veces.
- La función usa dos parámetros, un caracter y un número.
- En este programa hemos usado una estructura, por medio de la cual, pasamos a la función el caracter a imprimir y el número de veces que lo imprimirá.

# Programando hilos.

- En este segundo programa, crearemos tres hilos, cada uno imprimirá un caracter un cierto número de veces.
- La función usa dos parámetros, un caracter y un número.
- En este programa hemos usado una estructura, por medio de la cual, pasamos a la función el caracter a imprimir y el número de veces que lo imprimirá.

# Programando hilos.

- En este segundo programa, crearemos tres hilos, cada uno imprimirá un caracter un cierto número de veces.
- La función usa dos parámetros, un caracter y un número.
- En este programa hemos usado una estructura, por medio de la cual, pasamos a la función el caracter a imprimir y el número de veces que lo imprimirá.

# Programando hilos.

- Declaramos librerías y la estructura.

```
/* compilar con g++   -o   -lpthread*/
#include <pthread.h>
#include <cstdio>
#include <iostream>
using namespace std;
/*estructura de paso de datos a la funcion*/
struct char_print_parms
{
/* The character to print.  */
char caracter;
/* The number of times to print it.  */
int count;
};
```

# Programando hilos.

- Definimos la función a ejecutar por los hilos.

```
void* char_print (void* parameters)
{
/* Convierte el pointer al tipo correcto.  */
struct char_print_parms* p = (struct char_print_parms*) paramet
int i;
for (i = 0; i < p->count; ++i)
cout<< p->caracter;
return NULL;
}
```

# Programando hilos.

- Declaramos los identificadores de los hilos así como las estructuras para cada hilo.

```
int main ()
{
/*Declara los hilos*/
pthread_t thread1_id;
pthread_t thread2_id;
pthread_t thread3_id;
/* declara las tipos de estructuras*/
struct char_print_parms thread1_args;
struct char_print_parms thread2_args;
struct char_print_parms thread3_args;
/* Create a new thread to print 10,000 'x's.  */
```

# Programando hilos.

- Asignamos valores a los parámetros y creamos para cada hilo.

```
/* Create a new thread to print 10,000 'x's.  */
thread1_args.caracter = 'x';
thread1_args.count = 10000;
pthread_create (&thread1_id, NULL, &char_print, &thread1_args);
/* Create a new thread to print 10,000 o's.  */
thread2_args.caracter = 'o';
thread2_args.count = 10000;
pthread_create (&thread2_id, NULL, &char_print, &thread2_args);
/* Create a new thread to print 10,000 y's.  */
thread3_args.caracter = 'y';
thread3_args.count = 10000;
pthread_create (&thread3_id, NULL, &char_print, &thread3_args);
```

# Programando hilos.

- Hacemos join, y salimos.

```
/* Se asegura que el primer hilo ha terminado.  */
pthread_join (thread1_id, NULL);
/* Se asegura que el segundo hilo ha terminado..  */
pthread_join (thread2_id, NULL);
/* Se asegura que el tercer hilo ha terminado..  */
pthread_join (thread3_id, NULL);
/* Now we can safely return.  */
   pthread_exit(NULL);
return 0;
}
```

# Temario

# Práctica.

Vamos a la primera parte de práctica.

- Codifique los programas mostrados.
- El en siguiente paso, use arreglos de hilos y de estructuras, para el segundo programa.
- Por medio de un for, asigne valores a los parámetros para cada hilo.
- Compile y observe los resultados.

# Práctica.

Vamos a la primera parte de práctica.

- Codifique los programas mostrados.
- El en siguiente paso, use arreglos de hilos y de estructuras, para el segundo programa.
- Por medio de un for, asigne valores a los parámetros para cada hilo.
- Compile y observe los resultados.

# Práctica.

Vamos a la primera parte de práctica.

- Codifique los programas mostrados.
- El en siguiente paso, use arreglos de hilos y de estructuras, para el segundo programa.
- Por medio de un for, asigne valores a los parámetros para cada hilo.
- Compile y observe los resultados.

# Práctica.

Vamos a la primera parte de práctica.

- Codifique los programas mostrados.
- El en siguiente paso, use arreglos de hilos y de estructuras, para el segundo programa.
- Por medio de un for, asigne valores a los parámetros para cada hilo.
- Compile y observe los resultados.

# Práctica.

- La segunda parte, con las mismas características igual al anterior, haga un programa con cuatro hilos:
  - Un hilo, lea de un archivo e imprima lo que va leyendo.
  - el segundo, calcule una suma del 1 al 5000, y de el resultado.
  - El tercero, imprima en la pantalla 25,000 xs
  - el cuarto, que escriba el resultado del segundo punto, en un archivo.

# Bibliografía y Referencias



Mark Mitchell, Jeffrey Oldham, and Alex Samuel ”Advanced Linux Programming",USA 2001, New Riders Publishing


S. Kleiman, D. Shah, B. Smaalders. "Programming with THREADS", Sun Soft Press USA 1996 .


W. Stallings. "Sistemas Operativos ". Pearson 4a Ed México 2001.