

# Taller 1 - Arquitectura de Computadores 2

Josafat Vargas Gamboa, 2013030892

Febrero, 2020

## 1. Investigación

### 1.1. Investigue posibles métodos (bibliotecas, apis, etc) para el uso de hilos bajo el sistema operativo GNU Linux.

#### 1.1.1. POSIX

Las bibliotecas POSIX para hilos son un API basado en estándares (IEEE 1003.1c) para C y C++. Pthreads nace porque, históricamente, los vendedores de hardware implementaban threads de maneras muy diferentes y esto hacía la portabilidad de programas con hilos muy difícil para los programadores.

#### 1.1.2. MPI

MPI es una biblioteca de rutinas que puede ser utilizada para crear programas paralelos en C. Por limitaciones del lenguaje, que no soporta paralelismo, los vendedores crean una variedad de extensiones para que los usuarios puedan crear aplicaciones paralelas. De manera similar a POSIX, esto reduce la portabilidad e implica volver a entrenar a los programadores para cada plataforma. MPI se crea para reducir estos problemas.

MPI también soporta hardware heterogéneo, lo que le permite iniciar un programa que se ejecuta en múltiples sistemas de computación para resolver el mismo problema.

#### 1.1.3. GNU Portable Threads

Pth es una biblioteca muy portable basada en POSIX/ANSI-C para plataformas de UNIX que provee calendarización sin preferencias y basada en prioridades para múltiples hilos en ejecución dentro de aplicaciones guiadas por eventos. Todos los hilos corren en el mismo espacio de direcciones de la aplicación de servidor pero cada una tiene su propio PC, Stack, máscara de señal y variable de error.

La calendarización de los hilos está hecha de forma cooperativa y cabe notar que Pth está hecho para proveer máxima portabilidad y no las características más elegantes.

#### 1.1.4. `longjmp`

ANSI-C provee funciones para guardar y recuperar el contexto de ejecución para un buen manejo de errores. `setjmp` guarda el contexto actual de ejecución y `longjmp` devuelve un contexto previamente guardado. El único desafío nuevo para implementar fibras con estas funciones es que no proveen una forma de crear un nuevo stack. Simplemente recuperan un stack frame previo. Para crear un stack se puede utilizar `signalstack` para alocar un stack alternativo para el manejador de señales.

#### 1.1.5. `libfiber`

Es una biblioteca que maneja fibras a nivel de usuario con soporte para sistemas multinúcleo. Similar a los hilos de bajo peso con bloque de IO y rápido cambio de contexto, similar a las desarrolladas en Erlang o Go.

Las fibras son similares a los hilos pero usualmente solo se interrumpen cuando terminan por lo que comienzan y terminan en lugares bien definidos y hacen que la integridad de datos sea un problema menor. Las fibras también suelen usarse a nivel de usuario por lo que no necesitan llevar a cabo cambios de contexto o cambios al estado del CPU.

### 1.2. ¿Qué es el concepto de mutex en multiprogramación y qué busca hacer?

Los mutexes son rutinas que manejan la sincronización. Mutex es una abreviación para exclusión mutua. Estos proveen funciones para crear, destruir, bloquear y desbloquear regiones de memoria o recursos que pueden crear condiciones de carrera entre hilos y están suplementados por funciones de atributos que asignan o modifican atributos asociados con lo mutex.

### 1.3. ¿Qué sucede cuando dos hilos quieren utilizar el mismo recurso? ¿Cómo se manejan estos casos?

La mayoría de las funciones en computación se llevan a cabo en varios pasos, si dos hilos están intentando acceder a la misma sección de memoria y esta no se bloquea uno de los hilos puede tomar información desactualizada o eliminar las modificaciones creadas por otro hilo de ejecución. Por esta razón se definen dos tipos de bibliotecas, seguras e inseguras de acuerdo a si tienen el manejo de hilos. Si no se sabe o si no lo tienen se debe evitar el uso de hilos para evitar la corrupción de los datos.

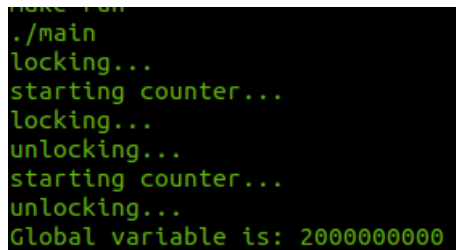
Existen varios modelos para programas con manejo de hilos:

- Manager/Worker: Existe un solo hilo, el manager, que asigna trabajo a los otros hilos (los trabajadores). Típicamente el manager maneja todas las entradas y asigna el trabajo con base en esto. La cantidad de hilos trabajadores se puede definir de manera dinámica o estática.

- Pipeline: Una tarea se divide en una serie de sub-operaciones, cada una es manejada en serie pero de forma concurrente por un thread diferente, similar al pipe de un procesador o una línea de producción de automóviles.
- Peer: Similar al modelos de Manager/Worker pero una vez que el thread inicial crea los otros threads, este participa en el trabajo.

## 2. Problema 2

### 2.1. Explique el inicio y fin de la ejecución de cada hilo



```

./main
locking...
starting counter...
locking...
unlocking...
starting counter...
unlocking...
Global variable is: 2000000000

```

Figura 1: Screenshot del output de la ejecución

En la figura 1 se puede ver como el contador inicia, en primer instancia después de que se bloquea el recurso y cuando el segundo thread intenta utilizarlo, bloquea el recurso pero el contador no comienza hasta después de que el primer hilo haya desbloqueado el recurso. Note que el print “locking...” ocurre antes de que se bloquee el mutex y “starting counter...” cuando ya el mutex está bloqueado. por otro lado “unlocking...” ocurre cuando ya el recurso está liberado.

## 3. Instrucciones de compilación

Esta sección también se encuentra en el README.

Abra una consola en esta dirección (la dirección del README) y para compilar el programa ejecute:

```
$ make
```

Para correr el programa ejecute:

```
$ make run
```

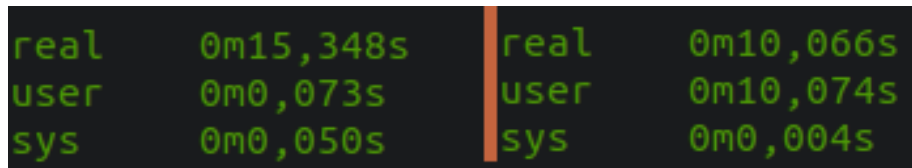
Para llevar el programa al estado original (solo código fuente) ejecute:

```
$ make reset
```

### 3.1. Notas de la solución 1

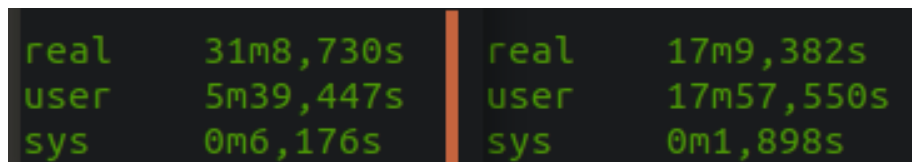
No se recomienda subir la cantidad de repeticiones a más de 100 000. Recuerde que la creación de cada elemento en los vectores tarda 10ms en el peor caso,

por lo que el programa tarda, como mínimo,  $10\text{ms} \times \text{repeticiones}$ . Con 100 000 repeticiones esto es aproximadamente 15 minutos, con un millón de repeticiones es 3 horas. El código fuente base cuenta con 1000 repeticiones, aproximadamente 10s.



real	0m15,348s	real	0m10,066s
user	0m0,073s	user	0m10,074s
sys	0m0,050s	sys	0m0,004s

Figura 2: Análisis de tiempo con 1000 repeticiones



real	31m8,730s	real	17m9,382s
user	5m39,447s	user	17m57,550s
sys	0m6,176s	sys	0m1,898s

Figura 3: Análisis de tiempo con 100 000 repeticiones

Las figuras 2 y 3 muestran el tiempo que tardó ejecutándose el programa. Adicionalmente estas imágenes tienen, en la columna izquierda el tiempo de ejecución sin hilos

### 3.2. Notas de la sección 2

Se utiliza un contador por su simplicidad y se crea con un `int` en vez de un `unsigned int` por lo que sobrepasar 2,147,483,647 causaría un overflow. Para la prueba se utilizan 1 000 000 000 repeticiones que es suficiente para ver cuando como uno de los threads termina mucho antes que el otro.