Proyecto 1: Lotería de *Threads*

Lunes 12 de Marzo

I. Descripción

El propósito de este proyecto es crear y controlar nuestros propios *threads* a nivel de código de usuario (*i.e.*, sin intervención del kernel). Toda la programación debe realizarse en C sobre Linux, no se puede usar de manera directa o indirecta ninguna biblioteca que implemente *threads* (e.g., no se puede usar para nada `Pthreads`).

II. Los Datos

La primera parte de su programa leerá los parámetros de la ejecución. Aunque el despliegue final será de tipo gráfico, si les resulta conveniente los datos originales pueden ser leídos de un archivo, o de teclado de manera texto. Los datos necesarios son:

- Modo de operación (expropiativo o no expropiativo).
- Número de *threads* (mínimo 5, máximo definido por el despliegue más grande que puedan hacer).
- Cantidad de boletos para cada *thread* (número entero).
- Cantidad de trabajo para cada *thread*.
- Tamaño del *quantum* en milisegundos (para la versión expropiativa) o porcentaje del trabajo a realizar antes de ceder voluntariamente el procesador (para la versión no expropiativa). Por ejemplo, se podría indicar que el *quantum* es de 100 milisegundos, o que se quiere que se realice un 3 % del trabajo antes de ceder el CPU.

Con esa información, el programa creará los *threads* de forma *ad hoc* para este proyecto, y lanzará un pequeño *scheduler* que usará *Lottery Scheduling*[1] para seleccionar el siguiente *thread* a ejecutar (ya sea que el *scheduler* haya obtenido el control en forma expropiativa o no expropiativa).

III. Los *Threads*

Se creará una versión simplificada de *threads* para este proyecto. Estos se construirán con el uso adecuado de las funciones estándar de biblioteca de C `sigsetjmp()` y `siglongjmp()`. Consultando previamente con el profesor se podrían usar otras funciones equivalentes.

Cada *thread* hará algún cálculo predeterminado que sea muy intensivo en uso de CPU, y actualizará un despliegue gráfico sencillo que muestra el avance hecho en el trabajo y el resultado parcial obtenido. Al terminar, cada *thread* desplegará en la interfaz gráfica los resultados finales del cálculo realizado.

En el modo no expropiativo, los *threads* cederán voluntariamente el procesador después de realizar una fracción de su trabajo. En el modo expropiativo los *threads* no ceden el procesador, pero al agotarse el *quantum*, el *scheduler* seleccionará otro *thread* para que corra.

IV. Trabajo

Nótese que $\arctan(1) = \frac{\pi}{4}$, por lo que $4\arctan(1) = \pi$. Usando este hecho, nuestros *threads* calcularán el valor de π. La función $\arctan$ será aproximada usando una serie geométrica (no usar Taylor) programada de manera iterativa[2]. Entre más términos tenga dicha serie, más exacto será el valor de π.

La unidad de trabajo mínima serán 50 términos de esta serie. Así, si indicamos que un *thread* tendrá que hacer 4 unidades de trabajo, significa que debe calcular y acumular 200 términos de la serie, o si a otro *thread* se le asignan 100 unidades de trabajo, éste deberá calcular y acumular 5000 términos de esta serie.

En el caso expropiativo cada *thread* trabajará continuamente, sin notar que el *scheduler* (activado por el *timer*) les expropia el CPU cada cierto tiempo. En el caso no expropiativo, cada *thread* calculará el porcentaje de términos que se le haya indicado y, de no haber terminado con todo su trabajo asignado, cederá voluntariamente el procesador al *scheduler* para que otro *thread* sea escogido (el *scheduler* podría escogerlo de nuevo).

V. Despliegue

Durante su trabajo, cada *thread* actualizará una barra que indique el porcentaje de su trabajo que ya haya terminado. También cada *thread* mostrará el valor acumulado de la función requerida para este punto.

El *scheduler* indicará de alguna manera apropiada cuál *thread* está activo en cada momento. Usar GTK o algún equivalente gráfico.

VI. Fecha de Entrega

Demostraciones en clase el **Lunes 12 de Marzo del 2018**. También mandar por correo electrónico a `torresrojas.cursos@gmail.com` antes de la clase. Coloque todo lo necesario para compilar, documentar y ejecutar su proyecto en un directorio cuyo nombre esté formado por los apellidos de los miembros de cada grupo separados por guiones. Compacte este directorio en la forma de un `.tgz` llamado igual que el directorio (e.g., `torres-venegas-castro-smith.tgz`) y envíelo por correo. Identifique claramente su correo con el siguiente subject:

`[SOA] Proyecto Programado 1 - Apellido 1 - Apellido 2 - Apellido 3`

Buena Suerte.

[1] **C. A. Waldspurger, W. E. Weihl**, "*Lottery Scheduling: Flexible Proportional-Share Resource Management*", Operating Systems Design and Implementation (OSDI), 1994.

[2] Ya que el objetivo del ejercicio es calcular el valor de π, no use ninguna serie que requiera π como dato de entrada.