Proyecto 1

Lotería de Threads

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**). El proyecto debe correr correctamente en nuestros laboratorios.

Los Datos

La primer 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 leidos 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 sin complicar el despliegue).
- Cantidad de boletos para cada thread (número entero).
- Cantidad de trabajo para cada *thread.*
- Tamaño del *quantum* (para la versión expropiativa) o factor del trabajo a realizar antes de ceder voluntariamente el procesador (para la versión no expropiativa).

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

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 `setjmp`() y `longjmp`(). Cada thread hará algún cálculo predeterminado que sea muy intensivo en uso de CPU, y actualizará un despliegue gráfico sencillo. 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.

Trabajo

Necesitamos calcular el valor de **2** *arccos*(**0**). Cuando un *thread* termine, desplegará el valor encontrado con la mayor precisión posible. Se usará la Serie de Taylor correspondiente para la aproximacióna esta función (el objetivo del ejercicio es calcular el valor de PI, no use ninguna serie que implique usar PI como dato de entrada). 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 de *scheduling* expropiativo cada thread trabajará continuamente, sin notar que el *scheduler* les expropia el CPU cada cierto tiempo. En el otro caso cada calculará cada vez tantos términos como su cantidad de boletos multiplicado por el factor indicado en los parámetros de entrada.

Despliegue

Durante su trabajo, cada *thread* actualizará una barra que indique el % de su trabajo que ya haya terminado. El scheduler indicará también cual *thread* está activo en cada momento. Usar GTK o algún equivalente gráfico.

Fecha de entrega: Demostraciones el **Lunes 12 de Marzo**. Enviar además por e-mail un .tgz conteniendo todos los programas fuentes y archivos pertinentes a `torresrojas@gmail.com` antes de las demostraciones. Identifique claramente a los miembros de cada equipo y ponga el subject indicado en clases.