Proyecto 1: The Scheduler

Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Profesor: Ernesto Rivera Alvarado

I. Introducción

El propósito de este proyecto es crear y gestionar nuestros propios procesos a nivel de código de usuario (sin intervención del kernel). Los estudiantes deberán investigar e implementar el algoritmo *lottery scheduling* según se explica en el paper P029 *"Lottery Scheduling: Flexible Proportional-Share Resource Management"* de Waldspurger. Además, escogerán para implementar alguno de los algoritmos que se le presentan a continuación: FCFS, SJF, RR, PS, PS con RR, MQS, MFQS con diferentes parámetros según se explican en el libro *Operating System Concepts* de Silberschatz.

II. Funcionamiento

Su programa leerá de un archivo de texto los parámetros de ejecución, los cuales contienen la siguiente información:

- Algoritmo de calendarización.
- Parámetros del algoritmo de calendarización (ejemplo MFQS).
- Modo de operación: Expropiativo y no expropiativo (para los que apliquen).
- Número de procesos: Mínimo 5, máximo 25.
- Tiempo de llegada del proceso.
- Cantidad de trabajo para cada proceso.
- Parámetros específicos de cada proceso según el algoritmo de calendarización (ejemplo cantidad de boletos para *lottery scheduling*).
- Tamaño del *quantum* (para la versión expropiativa), o cantidad de trabajo a realizar antes de ceder voluntariamente el procesador (versión no expropiativa).

Con esta información, el programa creará los procesos para este proyecto e iniciará una interfaz gráfica que mostrará el avance de los procesos según el algoritmo de calendarización y modo seleccionado.

III. Los procesos

Se hará una versión simplificada de procesos para este proyecto. Los mismos se harán con el uso adecuado de las funciones estándar de la biblioteca de C `sigsetjmp()` y `siglongjmp()` (el uso de otras bibliotecas para esta clase de funciones deben consultársele al profesor con al menos dos semanas de antelación a la presentación del proyecto). Cada proceso hará un cálculo predeterminado que será muy intensivo a nivel de CPU y actualizará su avance en el despliegue gráfico. Durante la ejecución, cada proceso se desplegará en la interfaz gráfica con el avance de los resultados del cálculo realizado.

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

IV. Trabajo

Cada proceso calculará el valor de $2arcsin(1)$. Durante la ejecución y finalización, el proceso desplegará el valor encontrado con la mayor precisión posible. Debe utilizar la Serie de Taylor correspondiente para aproximar esta función (el punto es aproximar el valor de PI, por lo que no utilice ninguna serie que pida PI como el dato de entrada). La unidad de trabajo mínima será de 50 términos de esta serie. Esto quiere decir de que si a un proceso le indicamos que tiene que hacer 5 unidades de trabajo, deberá realizar 250 términos de la serie, o si a otro proceso se le asignan 100 unidades de trabajo, éste deberá calcular y acumular 5000 términos de esta serie. En el caso del *scheduling* expropiativo, cada proceso trabajará continuamente sin notar que el *scheduler* (activado por el *timer*) le expropia el CPU cada cierto tiempo. En el caso no expropiativo, cada proceso 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 proceso sea escogido.

V. Despliegue

Durante la ejecución de un proceso, éste actualizará una barra con el porcentaje del trabajo que ya haya terminado. También, cada proceso mostrará el valor que lleve acumulado de PI. Por último, la interfaz mostrará cual proceso está activo en un momento dado.

VI. Requisitos indispensables

Cualquier incumplimiento de los siguientes requisitos vuelven al proyecto "no revisable" y se asigna automáticamente una nota de cero.

- Todo el código debe de estar escrito C sobre Linux.
- Debe utilizar GTK como interfaz gráfica.
- No debe presentarse *segmentation fault* bajo ninguna circunstancia.
- No puede usar de manera directa o indirecta una biblioteca que involucre *threads* (por ejemplo **pthreads**).
- El estudiante no proporciona el equipo necesario para revisar el proyecto.

VII. Evaluación

El requisito mínimo para que el proyecto sea revisado es que tenga la implementación de *lottery scheduling* completa y correcta con todo lo solicitado funcional (lectura del archivo de texto, interfaz gráfica). Esto corresponderá al 80% de su nota. El algoritmo adicional que haya escogido para implementar sumará el 20%.

VIII. Entrega

Las demostraciones se harán en **semana 5** en las horas de clase.

La carpeta comprimida `.zip` de su proyecto debe contener únicamente los archivos fuentes `.c`, un archivo de texto `readme.md` que describa con detalle el procedimiento para ejecutar y echar a andar el proyecto, además del archivo `Makefile` que compila y genera el ejecutable del código fuente. El nombre de la carpeta comprimida serán los apellidos de los integrantes del grupo de trabajo (por ejemplo `rivera-alvarado-alfaro.zip`) y este debe de ser subido por cada uno de los integrantes al Tec-Digital el día de la revisión antes de las 6 am. La revisión se hará con el archivo que hayan subido al Tec-Digital, y por ninguna razón se habilitará la entrega posterior a esa fecha y hora.