Universidad Simón Bolívar
Departamento de Computación y T.I.
Taller de Sistemas de Operación I

Tarea 1 (10 pts)

Objetivos

El objetivo principal de la tarea es que el estudiante adquiera destrezas de programación en el lenguaje C. Haremos énfasis en los siguientes elementos: estructuras de datos dinámicas, manejo de apuntadores, definición de tipos de datos y operaciones de E/S.

Enunciado de la Tarea

Se desea que uds. desarrollen en C la estructura de datos y las funciones asociadas para la implementación de un planificador de procesos o "scheduler"

1. Descripción General

1.1 La Estructura de datos del planificador

El planificador a implementar se basa en un esquema de prioridades. En el mismo, cada proceso tiene una prioridad asignada entre los números del 0 al 5, siendo los procesos de mayor prioridad aquellos que tienen asignado el valor 0 y los de menor prioridad los que tienen asignado el valor 5. La estructura de datos del planificador tendrá entonces un conjunto de 6 colas de procesos en orden de prioridad descendente: Q0, Q1, ...Q5, donde prioridad[Qi] > prioridad[Qj] para i < j.
Cuando el planificador requiere seleccionar un proceso para su ejecución, comenzará por los procesos de mayor prioridad (Q0). Si hay uno o más procesos con esta prioridad, se selecciona el primero de la cola. Si la cola Q0 está vacía, se examina la cola Q1 y así sucesivamente.

1.2 Rutinas que manejan la Estructura de Colas

Aunque son muchas las operaciones que se deben proveer para realizar la planificación de procesos, en esta tarea se implementarán las funciones que se describen a continuación:

Insertar Proceso: esta rutina permite insertar un proceso en la cola de prioridades que le corresponda
.
Eliminar Proceso: esta rutina permite eliminar un proceso, que ha culminado su ejecución, de la cola de prioridades donde se encuentre.

Eliminar Proceso en Ejecución: esta rutina permite eliminar de la estructura de datos al último proceso que se escogió para ejecución.

Próximo Proceso a Planificar: esta rutina devuelve el próximo proceso a planificar, es decir el próximo proceso que se ejecutará.

Cambiar Estado de Proceso: esta rutina cambia el estado de un proceso por el nuevo estado que recibe como parámetro. Por ahora se manejan sólo dos estados: *Listo* (L) y *EnEjecución* (E).

Construir estructura de procesos: En un sistema operativo en normal funcionamiento, las colas Q0, ... Q5, se van actualizando a medida que inicia la ejecución de un nuevo proceso. En nuestro caso, el estado inicial de las colas se obtendré a partir de un archivo de texto cuyo formato se describe en la sección 2.1

Imprimir estructura de Procesos: esta rutina imprime por la salida estándar información sobre los procesos que se encuentran en cada una de las colas del planificador.

2. Detalles de Implementación

2.1 Las rutinas

La descripción detallada de las rutinas y sus parámetros se presenta a continuación:

Insertar Proceso: esta rutina permite insertar un proceso en alguna de las colas de prioridades. Los procesos se insertarán al final de la cola que les corresponda. El prototipo de la rutina es el siguiente:
void InsertarProceso(EstrucSched s, Proceso* p, short prioridad)*

donde:
s es un apuntador a la estructura de colas del *planificador*,
p es un apuntador a la estructura de datos que almacena información perteneciente al proceso que se va a insertar
prioridad es un número del 0 al 5 y corresponde a la prioridad del proceso.

Los datos que se almacenarán para cada uno de los procesos son:

- PID: Identificador del proceso
- Estado: Estado del proceso. Se manejarán sólo los estados *Listo* (L) y *EnEjecución (E)*.
- Time: tiempo acumulado de ejecución del proceso.
- Comando: nombre del comando

Eliminar Proceso: esta rutina permite eliminar un proceso de la estructura de colas. El planificador la invocará cuando un determinado proceso ha terminado su ejecución. El prototipo de la rutina es el siguiente:

*void ElimProceso(EstrucSched *s, long pid, short prio)*

donde:
s es un apuntador a la estructura de colas del *planificador*,
pid es el PID del proceso a eliminar y
prio su prioridad.

Eliminar Proceso en Ejecución: esta rutina permite eliminar de la estructura de colas, al último proceso que se escogió para ejecución, es decir al único que tiene asignado en su estado el valor *EnEjecución.*La rutina recibe como parámetro la dirección de la estructura de colas.

*void ElimProcesoE(EstrucSched *s)*

Próximo Proceso a Planificar: esta rutina devuelve el próximo proceso a planificar. Como se mencionó anteriormente, cuando se va a seleccionar un proceso para su ejecución, se comienza por los procesos de mayor prioridad (Q0). Si hay uno o más procesos se selecciona el primero de la cola que esté listo para ejecutarse; supondremos que todos los procesos están inicialmente listos para ejecución. Si Q0 está vacía, se examina Q1 y así sucesivamente. El proceso escogido se coloca al final de su cola y se le cambia el estado de *Listo* a *EnEjecucion*. La función recibe como parámetro un apuntador a la estructura de colas y devuelve un apuntador al proceso escogido. Si no encuentra ningún proceso la rutina devuelve NULL. El prototipo de la función es el siguiente:

*Proceso *ProxProceso(EstrucSched *s);*

Cambiar Estado de Proceso:

*void CambiarEstado(EstrucSched *s, Proceso* p, Estado newestado)*
esta función asigna el estado almacenado en la variable *newestado*, al proceso *p* que recibe como parámetro. Aunque los posibles valores de *newestado* son *Listo* y *EnEjecución,* esta rutina será invocada normalmente por el planificador para indicar que un proceso dejó de ejecutarse y vuelve al estado *Listo*. Supondremos que los parámetros que recibe la rutina son correctos. Note que al igual que las rutinas anteriores, *CambiarEstado* también recibe un apuntador a la estructura de colas.

Construir estructura de procesos:

*EstrucSched *Construye(char *filename)*

Esta rutina recibe el nombre del archivo de prueba (*filename*) donde se encuentran procesos ficticios, y con esta información construye en la memoria la estructura de colas del planificador; una vez construída la estructuctura de colas, retorna un apuntador a la misma. El archivo de datos *filename* es un archivo de texto que contiene, en cada
línea, información sobre un proceso determinado, de la siguiente forma:

PID Estado Prioridad Time Comando
Donde,

- PID: es un entero que corresponde al PID del proceso.
- Estado: en este archivo se utilizará el carácter L (Listo) para indicar que el proceso está Listo para su ejecución. En este archivo todos los procesos se encontrarán en ese estado.
- Prioridad: es un número del 0 al 5
- Time: tiempo acumulado de ejecución, es un número real.
- Comando: nombre del comando

Ejemplo del archivo:

```
9344 L 2 10.4 netscape
```

```
1345 L 1 5.6 javac
6777 L 3 22.6 Kghostview
```
Supondremos que la información del archivo está correcta.

Imprimir estructura de Procesos: imprime por la salida estándar el estado actual de la estructura de colas, es decir, para cada cola, imprime información sobre los procesos que en ella se encuentran. Recibe como parámetros un apuntador a la estructura de colas.

*void Imprime(EstrucSched *s)*

2.2 El programa

Ud. deberá desarrollar un programa que permita probar las funciones por uds. desarrolladas. El programa se denominará **pscheduler** y recibirá como parámetros el nombre de dos archivos, en el primero (*datos*) se encuentran los datos de prueba (los procesos ficticios) y en el segundo (*archsalida*) se escribirá el contenido de la estructura de colas después que ha finalizado el programa. Desde la línea de comandos el programa se invoca de la siguiente forma:

$ **pscheduler** *datos archsalida*

Una vez que el programa ha construido la estructura de datos, debe ofrecer opciones al usuario para insertar procesos, eliminar procesos, cambiar estado del proceso, imprimir el contenido de la estructura de datos y salir del programa. Si el usuario desea insertar un proceso en la estructura de datos deberá indicar una prioridad del 0 al 5, un PID, un tiempo de ejecución ficticio y el nombre del comando. Para eliminar un proceso se debe suministrar su identificador o PID y su prioridad, a menos que se desee eliminar el proceso que se está ejecutando actualmente, en cuyo caso no se necesitan parámetros del usuario. Sólo se permitirá cambiar el estado de un proceso que está en ejecución (para que retorne de nuevo al estado *Listo*). Cuando el usuario indique que desea salirse del programa se debe escribir el archivo *archisalida* con información actualizada sobre los procesos que se encuentran en la estructura de colas. Recuerde hacer un manejo limpio de la memoria dinámica.
El formato del archivo de salida es libre, siempre que se entienda qué procesos pertenecen a cada cola y cuáles son sus características.

3. Observaciones Adicionales

- Ud. deberá respetar el nombre que se le dio al programa principal, así como los nombres de las rutinas, y el número y tipo de parámetros que cada una de ellas recibe o retorna. **Una violación de nombres o parámetros será penalizada.**
- **Debe respetar el formato del archivo de entrada** ya que este archivo lo suministrará su profesor. Puede suponer que el contenido del archivo *datos* es correcto.

Revise las llamadas al sistema *fprintf, fscanf* para la lectura y escritura de archivos de texto.

4. Entrega

La entrega del proyecto está pautada para el jueves 2 de febrero (23:55) usando un recurso de Moodle que le permitirá la entrega de su proyecto. Su entrega debe incluir:

- Un archivo README donde explique brevemente cómo están organizados los archivos de su programa y la estructura de datos utilizada
- El código fuente (archivos .c y .h)
- Un makefile que permita producir el ejecutable a partir de los archivos fuente.