Proyecto 2: Simulador de Planificador de Procesos con C++

Planificador de Procesos

Un planificador de procesos es una tarea del Sistema Operativo que se encarga de decidir cuál será el siguiente proceso en ejecutarse de una lista de procesos y le asigna un tiempo para ejecutarse.

Los procesos tienen su propio contexto, es decir, su propio conjunto de instrucciones y su propia sección de datos. Para este trabajo solamente se le pedirá que simule la ejecución de instrucciones, no deberá simular todo el contexto.

Cada proceso tiene la siguiente estructura:

proceso *[nombre del proceso] [prioridad del proceso]*

instrucciones

fin proceso

La primera línea debe indicar la palabra clave **proceso**, seguido del **nombre del proceso**, separados por espacios. Del mismo modo, como tercer elemento de la línea, debe aparecer una **prioridad** para el proceso, el cual será algún número entre 0 y 10 (este dato le será útil para el algoritmo por Prioridad).

En el bloque de instrucciones pueden haber dos tipos de instrucción:

* **Instrucciones normales**: Se ejecutan de manera "inmediata" una tras otra, en el periodo de tiempo que el procesador le asignó. Cada instrucción normal se ejecuta en una fracción de tiempo constante.
* **Operaciones de entrada/salida**: Representan un tiempo de retardo para el proceso. Por ejemplo, para mostrar datos se debe acceder al archivo de salida estándar (STDOUT) y esto representa un esfuerzo extra para el proceso, por lo que no es "inmediato". De forma similar, cuando se leen datos del usuario (STDIN), el proceso entra en un estado de "suspensión" hasta que se terminen de cargar los datos, por lo que tampoco es inmediato. Cada instrucción de entrada/salida se ejecuta en una fracción de tiempo variable. En el simulador, las operaciones de entrada/salida aparecerán por pares, dando a entender que la primera aparición es el inicio de la operación y la segunda es el fin de la operación.

Finalmente, para reconocer el final del proceso se deben colocar las dos palabras clave **fin proceso**.

Estos datos estarán almacenados en un archivo de texto el cual tendrá un nombre específico. En un solo archivo puede haber más de un proceso, la diferenciación entre dónde comienza y termina uno la hacen las palabras clave.

Un ejemplo del contenido de un archivo de procesos para la simulación sería así (se le adjuntan algunas muestras para que pruebe su programa):

proceso *programa1 0*

instruccion 1

instruccion 2

instruccion 1

e/s

e/s

instruccion 2

instruccion 3

instruccion 4

fin proceso

proceso *programa2 9*

instruccion 5

instruccion 1

instruccion 5

e/s

e/s

instruccion 1

instruccion 5

e/s

e/s

instruccion 7

instruccion 8

fin proceso

proceso *programaN 1*

instruccion 1

instruccion 3

instruccion 1

e/s

e/s

instruccion 5

instruccion 1

e/s

e/s

instruccion 5

e/s

e/s

instruccion 2

instruccion 5

instruccion 3

fin proceso

Su programa deberá cargar el contenido del archivo en estructuras de datos que representen los procesos, de modo que puedan ser "ejecutados". Todo esto deberá realizarlo mediante Programación Orientada a Objetos.

Los procesos estarán almacenados en lo que se denomina una **cola de procesos**. Cada proceso tendrá uno de los siguientes estados:

* **Listo**: El proceso ha sido cargado exitosamente en la cola de procesos y está listo para que el planificador le asigne la CPU y entre en ejecución.
* **En ejecución**: El proceso se encuentra ejecutando instrucciones actualmente. Debe ejecutar las instrucciones pendientes en el momento que el planificador le asigne el periodo de tiempo para uso de CPU.

En este estado se identifican otros subestados:

Activo: El proceso justo en ese momento le fue asignado el CPU por parte del planificador y se encuentra ejecutando antes que se le acabe el "quantum" (tiempo de CPU).

Cortado: El quantum se acaba y si el proceso tiene pendiente algo que ejecutar, es decir, no ha finalizado, vuelve a la cola de procesos hasta que se le vuelva asignar el CPU.

* **Bloqueado**: El proceso pasa del estado de "En ejecución" a "Bloqueado" al encontrar una una operación de **entrada/salida**. En el simulador, hasta que no se encuentre la siguiente operación de entrada/salida el proceso se mantendrá bloqueado. Después de encontrarla el proceso volverá a pasar al estado de "En ejecución" o "Finalizado".

Los subestados "**Activo**" y "**Cortado**" también aplican en este estado, ya que un proceso puede iniciar una operación de E/S y justo en ese momento se le acaba el quantum, por lo que la siguiente operación de E/S se ejecutaría en el siguiente quantum.

* **Finalizado**: El proceso terminó de ejecutar todas sus intrucciones y ya no necesita que el planificador le asigne tiempo de CPU, por lo tanto puede salir ya de la cola de procesos.

Su programa deberá utilizar los siguientes algoritmos para elegir el orden de ejecución de los algoritmos:

* **Planificación por turno rotatorio:** Llamado ***Round Robin***, es un algoritmo donde se determina el mismo tiempo para la ejecución de todos los procesos. Si un proceso no puede ejecutarse por completo en el tiempo asignado su ejecución será después de la ejecución de todos los procesos que se ejecuten con el tiempo asignado. Este algoritmo se fundamenta en FCFS y ordena la cola de procesos circularmente cuando se hallan en estado de listos.
* **Planificación por prioridad:** Esta planificación se caracteriza porque a cada proceso se le asigna una prioridad y se continúan con un criterio determinado. Los procesos serán atendidos de acuerdo con la prioridad determinada.

El programa inicia leyendo el archivo del proceso. Debe mostrar un menú donde se pueda elegir el nombre del archivo a cargar.

Después de cargado el archivo, se le solicitará cuál de los dos algoritmos desea ejecutar, si *Round Robin* o Prioridad. Al hacer esto su programa iniciará la simulación.

Cualquiera de los algoritmos deberá ejecutar todos los procesos hasta que todos estén en estado Finalizado.

En el ciclo de ejecución, su programa deberá asignar un tiempo de ejecución (quantum) para que los procesos ejecuten sus instrucciones. Los datos de los tiempos puede quemarlos en el código. Por

ejemplo, puede usar los siguientes tiempos:

* **Tiempo de CPU (quantum)**: 5 ciclos (simulados con 5 segundos)
* **Tiempo de ejecución de una instrucción**: 1 ciclo (1 segundo).
* **Tiempo de espera de una E/S**: 1.5 ciclos (1.5 segundos). Al ejecutarse por pares, las operaciones de entrada/salida suman 3 ciclos. El tiempo debería ser variable, pero al ser una simulación podemos usar un valor fijo un poco mayor al de una instrucción normal.

Según esos tiempos, considere lo siguiente:

Si un programa tiene 5 o menos instrucciones normales, en un único tiempo de quantum se ejecutaría por completo.

Si un programa tiene 10 instrucciones normales, necesitaría dos tiempos de quantum para ejecutarse, por lo que, el planificador le asigna una primera vez el tiempo de CPU para ejecutarse, luego lo "corta" y lo retoma hasta que le vuelva a tocar.

Si un programa tiene 4 operaciones de entrada/salida seguidas, es decir, dos pares de e/s, eso sumarían 6 ciclos, por lo que en un quantum se ejecutaría una parte y luego habría que esperar otro quantum para continuar la ejecución.

Utilice la función sleep o Sleep para simular los tiempos necesarios en su simulación.

Su programa deberá mostrar en pantalla las líneas del archivo conforme las vaya ejecutando.

Evaluación:
- **Funcionalidad (50 puntos):** Cada funcionalidad correctamente implementada y sin errores.
- **Calidad del Código (45 puntos)**: Estructura clara, comentarios apropiados y manejo de errores. Utiliza lo visto en clase, no utiliza bibliotecas del lenguaje ni externas. El proyecto tiene la estructura de carpetas vista en clase.
- **Documentación (5 puntos):** Claridad y precisión en las instrucciones de uso.

Los estudiantes pueden trabajar en grupos de **máximo 3 integrantes** y presentar sus proyectos en la fecha indicada en clase. Deberán comprimir su carpeta con la aplicación y subirla a Mediación Virtual.

Uso de **list**, **vector, map** y/o colecciones o estructuras de datos **no vistas en clase** reciben **nota de cero**.

Trabajos copiados (todos los involucrados) o trabajos hechos con herramientas de **inteligencia artificial** serán calificados con **nota de cero**.

Prepárense para presentar el trabajo en los laboratorios de la ECCI.