AYUDANTÍA T1

Germán Leandro Contreras Sagredo
Ricardo Esteban Schilling Broussaingaray

IIC2333 [2019-1] - Sistemas Operativos y Redes

INTRODUCCIÓN

INTRODUCCIÓN

Los objetivos de esta ayudantía son:

- Entender el funcionamiento del algoritmo de scheduling a implementar en la tarea 1.
- Resolver las dudas que surjan durante la explicación de lo pedido.

SCHEDULER

SCHEDULER

¿Qué es un *scheduler*?

- Encargado de decidir qué proceso poner a ejecución en la CPU. ¿Cuál es su objetivo principal?
- Puede ser una selección arbitraria (no recomendable... ¿por qué?) o basada en criterios.

SCHEDULER

¿Qué criterios utilizar? Dependerá del tipo de *scheduler*:

- *Preemptive* (Expropiativo): Hace uso de interrupciones para decidir cuándo cesar la ejecución de un proceso.
- *Non-Preemptive* (No Expropiativo): Esperan a que el proceso termine su ejecución (ya sea voluntariamente, por I/O o por término).

PRIORITY SCHEDULING

PRIORITY SCHEDULING

¿Cómo funciona?

1. Atiende a los procesos que tengan mayor prioridad dentro de una cola.
2. Para el caso *preemptive*, deja que cada proceso ejecute un máximo de tiempo antes de interrumpirlos (que lo entenderemos como **bloqueo**) para darle el paso al siguiente. A este tiempo le llamamos *quantum*.
3. Cuando es *non-preemptive*, dejamos que el proceso sea el que decida en qué momento salir de la CPU.
4. Puede causar *starvation* de procesos con baja prioridad. ¿Qué significa esto?

PRIORITY SCHEDULING

Además de las características antes mencionadas, en la simulación consideraremos lo siguiente con respecto a los procesos:

- Tienen periodos en los que hacen uso de la CPU (CPU-burst o ráfaga) y otros en los que esperan un ingreso por I/O (I/O-burst).
- En la secuencia de input, los tiempos A_i reflejan una ráfaga y los tiempos B_j reflejan una espera de input (que simularemos como una simple espera).

PRIORITY SCHEDULING

Además de las características antes mencionadas, en la simulación consideraremos lo siguiente con respecto a los procesos:

- Poseen estados. Estos son: `READY`, `RUNNING`, `WAITING` y `FINISHED`.
- Los procesos ubicados en la cola se encuentran en estado `READY` y, al ingresar a la CPU para ejecutar, pasan a estar en estado `RUNNING`.
- Pasan a estado `WAITING` una vez que terminan una ráfaga y posteriormente viene un tiempo de espera. **No ingresan a la cola hasta haber terminado este intervalo de tiempo.**
- Pasan a estado `FINISHED` al terminar de ejecutar la última ráfaga.

ESTADÍSTICAS

ESTADÍSTICAS

Durante la simulación, **debe** obtener una estadística general para cada proceso.

Veremos en qué consiste cada una de estas.

ESTADÍSTICAS

Los siguientes datos tienen relación con los accesos de los procesos a la CPU:

- Número de usos de la CPU: Corresponde a la cantidad de veces que el proceso ingresa a la CPU para ejecutar, es decir, la cantidad de veces que pasa a estado `RUNNING`.
- Número de bloqueos: O bien el número de interrupciones. Corresponde a la cantidad de veces que el scheduler decide sacar al proceso de la CPU por el consumo del quantum. ¿Qué valor tiene esta estadística en la versión no expropiativa?

ESTADÍSTICAS

A continuación, se presentan distintas métricas de tiempo que también debe calcular a lo largo de la simulación:

- *Turnaround time*: Tiempo total que le toma al proceso terminar su ejecución, es decir, el tiempo que le toma pasar a estado `FINISHED` desde que ingresa a la cola por primera vez.
- *Response time*: Tiempo que le toma al proceso ser atendido por primera vez una vez que ingresa a la cola en estado `READY`.
- *Waiting time*: Tiempo total que el proceso estuvo en espera, ya sea en la cola esperando a ser atendido, o bien en estado `WAITING`.

ESTADÍSTICAS

Tomemos como ejemplo un proceso que pasa por los siguientes estados:

```
[t = 12] El proceso GERMY ha sido creado.
[t = 15] El proceso GERMY ha pasado a estado RUNNING.
[t = 17] El proceso GERMY ha pasado a estado WAITING.
[t = 18] El proceso GERMY ha pasado a estado READY.
[t = 19] El proceso GERMY ha pasado a estado RUNNING.
[t = 20] El proceso GERMY ha pasado a estado FINISHED.
```

En base al ejemplo anterior, tenemos que:

- *Turnaround time*

$$T_{\text{Término}} - T_{\text{Llegada}} = 20 - 12 = 8$$

- *Response time*

$$T_{\text{Atendido}} - T_{\text{Llegada}} = 15 - 12 = 3$$

- *Waiting time*

$$\sum_{i}(T^{i}_{\texttt{RUNNING}} - T^{i}_{\texttt{READY}}) + \sum_{j}(T^{j}_{\texttt{READY}} - T^{j}_{\texttt{WAITING}})$$

$$= ((15 - 12) + (19 - 18)) + ((18 - 17)) = 5$$

PRECAUCIONES, CONSEJOS, OTROS

PRECAUCIONES Y CONSEJOS

- Estamos trabajando con unidades adimensionales de tiempo, una buena referencia es simplemente usar una iteración para considerar esta medida.
- No existe un orden predeterminado para realizar los cambios en los procesos, ya sea de estados o estadísticas. Lo importante es que sean consistentes con su implementación.
- Al pedir un tiempo de ejecución de su tarea menor a 30 segundos, no buscamos que sea muy eficiente con una gran cantidad de procesos. Con esta restricción simplemente esperamos que su implementación tome un tiempo lo suficientemente razonable para no quedarse pegada en casos simples.

PRECAUCIONES Y CONSEJOS

- Lean bien el enunciado y modelen el problema según lo pedido, será más fácil para ustedes después simular el comportamiento del scheduler si tienen todo estructurado.
- Usen punteros, ocupan menos espacio y es menos probable que incurran en errores.
- Comiencen implementando la versión no expropiativa, les dará una buena base para poder implementar el resto.

TESTS

¿Tendremos *tests*?

Sí, se subirán dos archivos de prueba y sus soluciones para que puedan verificar el resultado de su tarea.

¿Cuándo estarán disponibles?

A más tardar el fin de semana, les pedimos paciencia.

DUDAS, CONSULTAS

¿Dudas, consultas?

FIN