AYUDANTÍA T2

Germán Leandro Contreras Sagredo
Ricardo Esteban Schilling Broussaingaray

IIC2333 [2019-2] - Sistemas Operativos y Redes

INTRODUCCIÓN

INTRODUCCIÓN

Los objetivos de esta ayudantía son:

- Entender el funcionamiento del algoritmo de scheduling a implementar en la tarea 2.
- 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).

SHORTEST REMAINING TIME FIRST

SHORTEST REMAINING TIME FIRST

¿Cómo funciona?

1. Atiende a los procesos que tengan un menor tiempo de ejecución restante en la cola.
2. Para el caso *preemptive*, interrumpe al proceso en ejecución si se crea un proceso (i.e. llega a la cola) que tenga un tiempo de ejecución menor. También existe interrupción si un proceso pasa de estado `WAITING` a `READY` y posee un tiempo de ejecución menor (detalle descrito más adelante).
3. Si bien depende mucho de la estimación que realice el sistema operativo de la duración que tendrá la ejecución de cada proceso, en el contexto de esta tarea asumiremos que conocemos el tiempo exacto.
4. Este algoritmo puede causar *starvation* de procesos con grandes ráfagas de ejecución. ¿Qué significa esto?

SHORTEST REMAINING TIME FIRST

En esta tarea, trabajaremos con dos versiones expropiativas del algoritmo:

1. short-sighted: En esta versión, se escoge el proceso con la ráfaga **actual** de ejecución menor.
2. long-sighted: En esta versión, se escoge el proceso cuyo **tiempo de ejecución total** (suma de todas sus ráfagas) es menor.

¿Y qué es una ráfaga de ejecución? Lo veremos a continuación.

SHORTEST REMAINING TIME FIRST

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).

SHORTEST REMAINING TIME FIRST

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.

SHORTEST REMAINING TIME FIRST

Ejemplo

```
    P1  1  3  4  5  6  5  5
    P2  2  3  6  3  4  5  5
    P3  0  1  3
```

¿Quién ejecuta primero en la versión short-sighted?
¿Y en la versión long-sighted?
¿Cómo evoluciona la ejecución?

Short-sighted

```
[t=0] El proceso P3 ha sido creado.
[t=0] El proceso P3 ha pasado a estado RUNNING.
[t=1] El proceso P1 ha sido creado.
[t=2] El proceso P2 ha sido creado.
[t=2] El proceso P3 ha pasado a estado FINISHED.
[t=3] El proceso P1 ha pasado a estado RUNNING.*
[t=6] El proceso P1 ha pasado a estado WAITING.
[t=7] El proceso P2 ha pasado a estado RUNNING.
...
```

Long-sighted

```
[t=0] El proceso P3 ha pasado a estado READY.
[t=0] El proceso P3 ha pasado a estado RUNNING.
[t=1] El proceso P1 ha sido creado.
[t=2] El proceso P2 ha sido creado.
[t=2] El proceso P3 ha pasado a estado FINISHED.
[t=3] El proceso P2 ha pasado a estado RUNNING.*
[t=8] El proceso P2 ha pasado a estado WAITING.
[t=9] El proceso P1 ha pasado a estado RUNNING.
...
```

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`. Para cada proceso, en esta tarea, se puede **deducir** su número mínimo de usos de CPU, ¿cuál es?
- 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 arribo de un proceso con tiempo de ejecución menor (según la versión).

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*

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

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

CASOS ESPECIALES

CASOS ESPECIALES

Existen casos especiales en que no necesariamente hay una única forma de manejarlos. Para facilitar la corrección y estandarizar las simulaciones, se ha determinado qué considerar en cada uno de estos.

CASOS ESPECIALES

- Si para un tiempo t = i el scheduler debe escoger un proceso y existe más de uno con **el mismo** tiempo de ejecución restante (según la versión del algoritmo), entonces escoge el de mayor PID (esto es, el proceso creado hace menos tiempo). Esto es **solo para escoger un proceso, no para interrumpir**.
- Su proceso puede llegar en tiempo t = 0. Debe asegurarse de que su programa no se caiga en este caso.

PRECAUCIONES, CONSEJOS, OTROS

PRECAUCIONES, CONSEJOS, OTROS

- 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 10 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, CONSEJOS, OTROS

- 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.
- NO ES NECESARIO QUE MODELEN LA COLA COMO UNA ESTRUCTURA DE DATOS A PARTIR DE LISTAS ENLAZADAS. Esto podría complicar el modelamiento del problema de forma innecesaria. Basta con definir dentro de `struct Queue` un atributo `Process ** processes`, donde pueden estar todos sus procesos inicializados.

¿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.

PRECAUCIONES, CONSEJOS, OTROS

¿Dudas, consultas?

FIN