Midterm 2 - MLQ

Oscar Vargas Pabon

Repositorio

https://github.com/osvarp/sistemas_operativos_midterm2.git

Organización

En primitives.h se encuentran las definiciones de las clases que uso para guardar los datos referentes a los procesos (Burst Time, Arrival Time, Tag,etc.).

En scheduler_algo.h tengo las clases virtuales que marcan la interfaz de mis algoritmos de scheduling (porque hago polimorfismo). Aquí aparece un concepto muy importante para entender mi implementación y es el de los 'steps'. Los algoritmos que pueden hacer parte de una de las colas del MLQ los implemento para que realicen solo 1 paso a la vez. Esto lo hago porque es el MLQ el que finalmente sabe cuanto tiempo puede destinarle a una cola especifica, potencialmente interrumpiendo la cola en ejecución bajo la llegada de un nuevo proceso. Otra cosa para notar es que el MLQ lo que retorna es el diagrama de Gantt representado por una lista de tuplas ('Tag','tiempo').

En mis clases 'Rr_step', 'Sjf_step', 'Stcf_step' y 'Fcfs_step' implemento los algoritmos Round Robin, Shortest Job First, Shortest to Completion First y First Come First Serve. Estas están todas separadas en un .h y .cpp.

En main.cpp gestiono toda la entrada y la salida, además de procesar las estadísticas (los promedios y los Waiting Time, Completion Time,etc.). Notar que asumo que todos los datos de la columna queue son válidos y identifican correctamente una de las colas del MLQ en ejecución. Trabajo las colas con indexación 0.

Por último, quiero resaltar que haciendo MLQ<FCFS> (MLQ con una única cola ejecutando FCFS) es equivalente a tener un FCFS. Este truco lo utilicé para probar mis implementaciones de RR, FCFS, SJF, STCF.

Ejecutando el código

Cuando no se le añade ningún parámetro, o el main.cpp no reconoce el parámetro, se ejecuta un Round Robin con quanta=5. Cuando se le añade '-1' ejecuta MLQ<RR(1), RR(3), SJF>. Cuando se le añade '-2' ejecuta MLQ< RR(3), RR(5), FCFS >. Cuando se le añade '-3' ejecuta MLQ< RR(2), RR(3), STCF >. Cuando se le añade '-4' ejecuta FCFS. Cuando se le añade '-5' ejecuta STCF. Cuando se le añade '-6' ejecuta SJF.

Para compilar utilizo (que se encuentra en compilar.cmd):

```
g++ main.cpp mlq.cpp rr_step.cpp fcfs_step.cpp stcf_step.cpp sjf_step.cpp -o exec
```

Verificando casos

Input/test002.txt

```
# Archivo: test002.txt
# etiqueta; burst time (BT); arrival time (AT);Queue (Q);Priority(5>1)
A; 6; 1; 0; 5
B; 9; 0; 0; 4
C;10; 0; 0; 3
D;15; 0; 0; 3
E; 8; 0; 0; 2
```

Un algoritmo como SJF ejecuta E por completo antes de pasar a A, sin embargo, STCF interrumpe a E apenas llega A. Por otro lado, FCFS ejecuta A de último. RR(5) termina A, B , C y E en dos vueltas, D en tres.

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -5 < input\test002.txt
Info: Ejecutando un Shortest Time to Completion First (STCF).
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 1 ; 0 ; 5 ; 0 ; 7 ; 1 ; 6;
B ; 9 ; 0 ; 0 ; 4 ; 14 ; 23 ; 0 ; 23;
C ; 10 ; 0 ; 0 ; 3 ; 23 ; 33 ; 23 ; 33;
D ; 15 ; 0 ; 0 ; 3 ; 33 ; 48 ; 33 ; 48;
E ; 8 ; 0 ; 0 ; 2 ; 6 ; 14 ; 0 ; 14;
WT=15.2; CT=25; RT=11.4; TAT=24.8;
```

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -6 < input\test002.txt
Info: Ejecutando un Shortest Job First (SJF).
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 1 ; 0 ; 5 ; 7 ; 14 ; 8 ; 13;
B ; 9 ; 0 ; 0 ; 4 ; 14 ; 23 ; 14 ; 23;
C ; 10 ; 0 ; 0 ; 3 ; 23 ; 33 ; 23 ; 33;
D ; 15 ; 0 ; 0 ; 3 ; 33 ; 48 ; 33 ; 48;
E ; 8 ; 0 ; 0 ; 2 ; 0 ; 8 ; 0 ; 8;
WT=15.4; CT=25.2; RT=15.6; TAT=25;
```

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -4 < input\test002.txt
Info: Ejecutando un First Come First Serve (FCFS).
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 1 ; 0 ; 5 ; 41 ; 48 ; 42 ; 47;
B ; 9 ; 0 ; 0 ; 4 ; 0 ; 9 ; 0 ; 9;
C ; 10 ; 0 ; 0 ; 3 ; 9 ; 19 ; 9 ; 19;
D ; 15 ; 0 ; 0 ; 3 ; 19 ; 34 ; 19 ; 34;
E ; 8 ; 0 ; 0 ; 2 ; 34 ; 42 ; 34 ; 42;
WT=20.6; CT=30.4; RT=20.8; TAT=30.2;
```

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec < input\test002.txt
Info: Ejecutando un RoundRobin con quanta 5
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 1 ; 0 ; 5 ; 36 ; 43 ; 20 ; 42;
B ; 9 ; 0 ; 0 ; 4 ; 20 ; 29 ; 0 ; 29;
C ; 10 ; 0 ; 0 ; 3 ; 24 ; 34 ; 5 ; 34;
D ; 15 ; 0 ; 0 ; 3 ; 33 ; 48 ; 10 ; 48;
E ; 8 ; 0 ; 0 ; 2 ; 34 ; 42 ; 15 ; 42;
WT=29.4; CT=39.2; RT=10; TAT=39;
```

Nótese del STCF que A tuvo WT=0. En SJF A tuvo WT=7. Para FCFS A tuvo Wt=41. Esto corrobora el comportamiento esperado.

Input/mytest001.txt

```
# mytest01.txt
# tag    ;  BT    ; AT     ;   Q    ; Pr(no la uso)

A        ;6       ;6       ;1       ;-1
B        ;8       ;10      ;0       ;-1
C        ;4       ;1       ;2       ;-1
D        ;11      ;0       ;2       ;-1
```

Este caso lo revisaremos con los esquemas recomendados con 3 colas.

MLQ< RR(1), RR(3), SJF > Ejecuta D hasta la llegada de A, momento en el cual pasa a las colas 0 (y posteriormente 1). Cuando vuelve a darle espacio en la cola 2, este escoge C y luego D.

MLQ< RR(3), RR(5), FCFS > Ejecuta D hasta la llegada de A, momento en el cual pasa a las colas 0 (y posteriormente 1). Cuando vuelve a darle espacio en la cola 2, este escoge D y luego C.

MLQ< RR(2), RR(3), STCF > Ejecuta D por un tiempo y luego ejecuta C hasta que termina. Cuando llega A, pasa a la cola 0. De aquí en adelante trabaja según la prioridad de las colas.

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -1 < input\mytest01.txt
Info: Ejecutando un MLQ< RR(1), RR(3), SJF >.
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 6 ; 1 ; -1 ; 12 ; 24 ; 9 ; 18;
B ; 8 ; 10 ; 0 ; -1 ; 0 ; 18 ; 10 ; 8;
C ; 4 ; 1 ; 2 ; -1 ; 14 ; 19 ; 6 ; 18;
D ; 11 ; 0 ; 2 ; -1 ; 18 ; 29 ; 0 ; 29;
WT=11; CT=22.5; RT=6.25; TAT=18.25;
```

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -2 < input\mytest01.txt
Info: Ejecutando un MLQ< RR(3), RR(5), FCFS >.
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 6 ; 1 ; -1 ; 12 ; 24 ; 18 ; 18;
B ; 8 ; 10 ; 0 ; -1 ; 0 ; 18 ; 10 ; 8;
C ; 4 ; 1 ; 2 ; -1 ; 5 ; 10 ; 6 ; 9;
D ; 11 ; 0 ; 2 ; -1 ; 18 ; 29 ; 0 ; 29;
WT=8.75; CT=20.25; RT=8.5; TAT=16;
```

```
C:\Users\oscar\Local\ponti\6\sisoper\2midterm>exec -3 < input\mytest01.txt
Info: Ejecutando un MLQ< RR(2), RR(3), STCF >.
# Powered by Oscar
# etiqueta; BT; AT; Q; Pr; WT; CT; RT; TAT
A ; 6 ; 6 ; 1 ; -1 ; 8 ; 20 ; 6 ; 14;
B ; 8 ; 10 ; 0 ; -1 ; 0 ; 18 ; 10 ; 8;
C ; 4 ; 1 ; 2 ; -1 ; 0 ; 5 ; 1 ; 4;
D ; 11 ; 0 ; 2 ; -1 ; 18 ; 29 ; 0 ; 29;
WT=6.5; CT=18; RT=4.25; TAT=13.75;
```