Trabajo Práctico de Sistemas Operativos

16 de septiembre de 2014

Universidad de Buenos Aires - Departamento de Computación - FCEN

Integrantes:

- Castro, Damián L.U.: 326/11 ltdicai@gmail.com
- Toffoletti, Luis L.U.: 827/11 luis.toffoletti@gmail.com
- Zanollo, Florencia L.U.: 934/11 florenciazanollo@gmail.com

Índice

1. Introducción

1.1. Ej 1: TaskConsola

Algorithm 1 TaskConsola

```
1: procedure TaskConsola(n, bmin, bmax)
2:     srand(time(NULL))                                  ▷ seteo semilla de random
3:     for i ← 0, n do                                    ▷ n veces
4:         random_number ← modulo(rand(), bmax − bmin + 1) + bmin   ▷ [1]
5:         Uso_IO(random_number)
6:     end for
7: end procedure
```

[1]: rand() retorna un número arbitrariamente grande. Tomando su módulo en (bmax - bmin + 1) nos aseguramos que esté entre 0 y (bmax - bmin).
Luego sumamos bmin, resultando un número entre bmin y bmax.

1.2. Ej 2: Ejecutar TaskConsola con FCFS

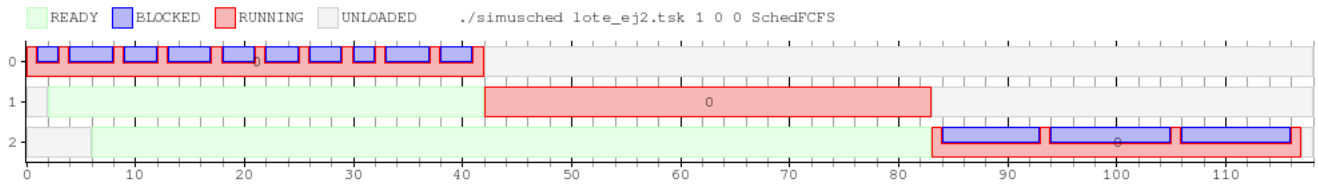
FCFS (First-Come First-Served) un scheduler simple en el cual los procesos son asignados al CPU en el orden en que estos lo requieren.

```
TaskConsola 10 2 4
@2:
TaskCPU 40
@6:
TaskConsola 3 6 11
```

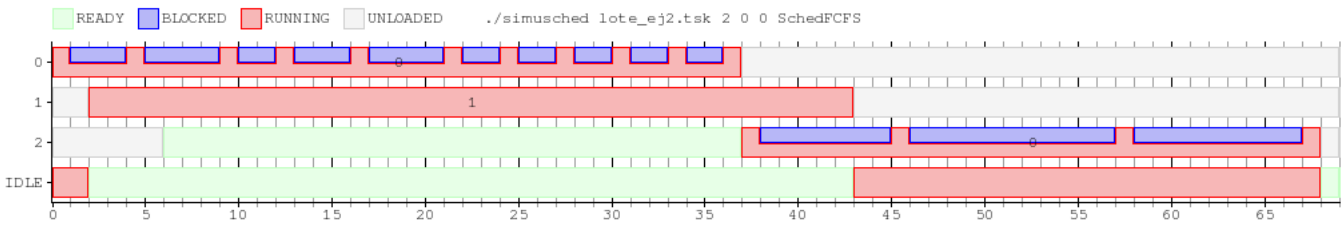
Básicamente hay una sola cola (FIFO) de procesos 'Ready'. Cuando un proceso requiere CPU y éste está libre, se lo deja correr tanto tiempo como quiera y sin interrupciones.

A continuación se muestran los gráficos para 1, 2 y 3 núcleos, usando SchedFCFS para el lote de tareas del cuadro.

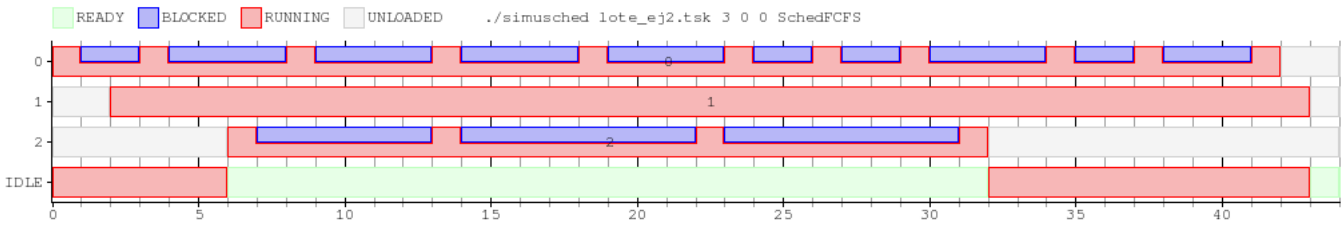
1 Núcleo:



2 Núcleos:



3 Núcleos:



Obs: Los parámetros de costo de cambio de contexto y migración fueron seteados en 0 ya que en este scheduler en particular no son tomados en cuenta. Porque núnca se desalojan las tareas ni se cambian de núcleo.

En los gráficos se hace evidente que a más cántidad de núcleos, mejor rendimiento, con FCFS. Esto se debe a que FCFS no soporta múltitarea por sí sólo (ya que núnca desaloja a las tareas), sino que necesita varios núcleos para lograrlo.

Las desventajas de FCFS son varias:

- Como ya dijimos, no soporta múltitarea.
- Puede generar mucho 'waiting time'.
  Si está ejecutando tareas muy largas las nuevas no entran hasta que éstas terminen.
- No tiene buen rendimiento, a menos que se sepa la duración de las tareas de antemano.
- Carece de 'fairness' i.e. no distribuye el/los procesador/es de forma justa entre las tareas.

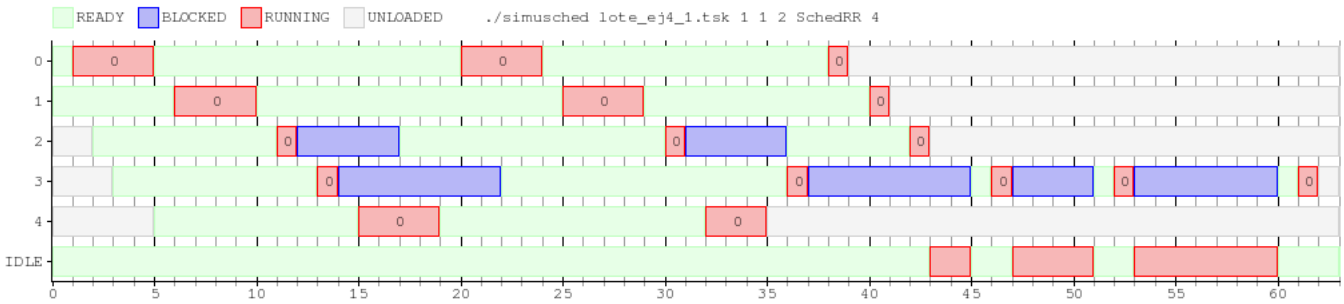
1.3. Ej 3: Round-Robin Implementación

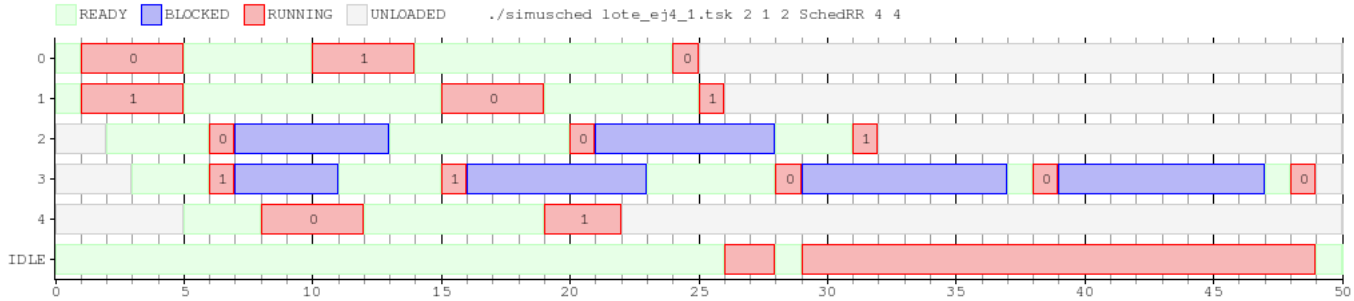
-Explicar la idea y la funcionalidad de c/u de las estructuras de datos usadas -Hacer pseudocódigo

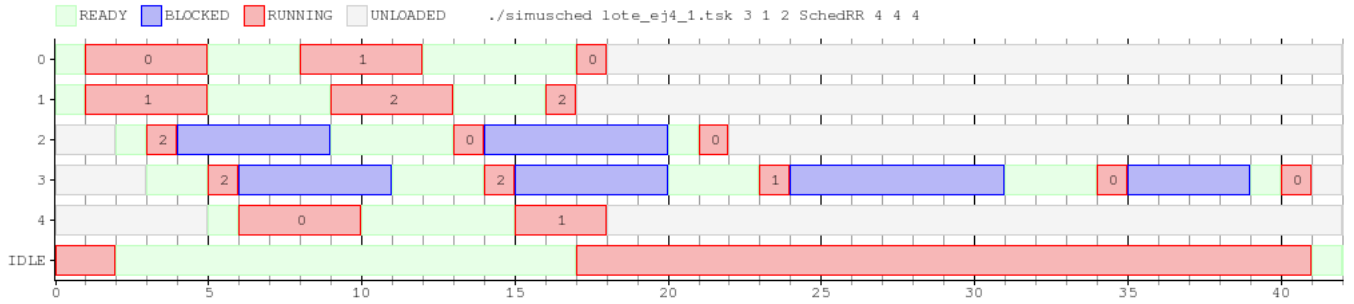
1.4. Ej 4: Round-Robin Simulaciones

!Explicar por qué: -elegimos 1 costo cambio de contexto y 2 costo cambio de nucleo (hay una lista para todos los nucleos) -es efectivamente un RR -no siempre es mejor tener más núcleos en un RR con lista global Esto depende de cuándo entren las tareas, que procesadores estaban libres y el quantum de c/u

En estas im todos los cpu tienen quantum = 4 -Hacer otros experimentos con dif quantums y ver cuál queda mejor







2. Discusión

3. Conclusiones