3.4. Manual de uso del simulador

El objetivo de esta práctica es completar el simulador de planificadores que se ha implementado. Antes de comenzar la descripción del simulador y de las modificaciones propuestas en esta práctica, vamos a comentar brevemente cómo usarlo y sus principales funciones. En su estado actual (tal y como se entrega al alumno), el entorno es funcional y permite:

- Especificar el número de CPUs que serán simuladas
- Escoger qué planificador se usará. *RR (round-robin)* y *SJF* (primero el trabajo más corto) ya están implementados.
- Decidir si queremos la versión *preemptive* del planificador seleccionado
- Para el planificador *round-robin*, especificar el *quanto* de tiempo que se asignará a las tareas
- Seleccionar el periodo con el que se ejecutará el equilibrador de carga (además, se ejecutará siempre que una CPU esté ociosa)
- Generar gráficas de la planificación realizada en cada procesador, incluyendo información del tiempo de ejecución, tiempo bloqueado, tiempo en espera....

3.4.1. Ejecución y generación de diagramas

Una vez compilado (usando el *Makefile* entregado), podemos proceder a ejecutar el simulador. Para ver todas las opciones disponibles haremos lo siguiente:

```
$ ./schedsim -h
```

que devolverá un listado parecido al siguiente:

```
Usage: ./schedsim -i <input-file> [options]

List of options:
        -h: Displays this help message
        -n <cpus>: Sets number of CPUs for the simulato (default 1)r
        -m <nsteps>: Sets the maximum number of simulation steps (default 50)
        -s <scheduler>: Selects the scheduler for the simulation (default RR)
        -d: Turns on debug mode (default OFF)
        -p: Selects the preemptive version of SJF or PRIO (only if they are selected
            with -s)
        -t <msecs>: Selects the tick delay for the simulator (default 250)
        -q <quantum>: Set up the timeslice or quantum for the RR algorithm (default 3)
        -l <period>: Set up the load balancing period (specified in simulation steps,
            default 5)
        -L: List available scheduling algorithms
```

Muchas de las opciones son autoexplicativas, y coinciden con las funcionalidades enumeradas anteriormente. A continuación, podemos realizar una primera simulación:

```
$ ./schedsim -i examples/example1.txt
```

La ejecución imprimirá por pantalla estadísticas de la ejecución de cada una de las tareas especificadas en el fichero *examples/example1.txt* y creará un fichero `CPU_0.log` que usaremos para generar un diagrama de la planificación. Para ello, usaremos la herramienta `generate_gantt_chart` en el directorio `gantt-gplot`:

```
$ cd ../gantt-gplot
$ ./generate_gantt_chart ../schedsim/CPU_0.log
```

Si todo ha ido bien en el directorio `schedsim` se habrá generado `CPU_0.eps`. La figura 3.1 muestra el resultado para ese ejemplo concreto. Vemos la progresión de cada una de las cuatro tareas que forman este ejemplo. Las partes azules indican tiempo de ejecución en CPU (de ahí que nunca solapen), la parte amarilla es tiempo de bloqueo (p.ej., simulando entrada/salida) y las zonas grises representan el tiempo en que una tarea estaba preparada para su ejecución pero la CPU estaba ocupada con otra tarea.

Podemos ejecutar el mismo ejemplo pero definiendo un sistema con 2 CPUs:

```
$ ./schedsim -i examples/example1.txt -n 2
```

La figura 3.2 muestra el resultado obtenido tras generar las diagramas.

Como podemos comprobar, al usar dos CPUs el tiempo total de ejecución pasa de 18 unidades de tiempo a 11. El reparto inicial de tareas a CPUs se hace de forma circular en función del número de la tarea: la tarea `P1` a la CPU 0, la tarea `P2` a la CPU 1, la tarea `P3` a la CPU 0.... En este ejemplo el planificador no ha llevado a cabo ninguna migración, por lo que cada tarea finaliza en la misma CPU en la que comenzó su ejecución.

3.4.2. Descripción del perfil de ejecución de las tareas

Cada fila del fichero representa una nueva tarea que se simulará en el sistema. La primera columna representa el nombre de la tarea. La segunda columna, su prioridad. Posteriormente se indica el tiempo de llegada de la tarea al sistema (si es 0, se indica que la tarea está disponible desde el comienzo de la simulación). A partir de ahí encontramos el perfil de ejecución de cada tarea: el primer número indica el tiempo de ejecución de la primera ráfaga de CPU; luego, tiempo de espera (por E/S o similar). Y el patrón se repite hasta que la tarea finaliza (es decir, el mismo formato que se usa en la hoja de ejercicios).

Por ejemplo, en `example1.txt` se especifica que se crearán 4 tareas. La primera tarea, llamada `P1` comienza su ejecución al principio de la simulación y tiene prioridad 1. Al comenzar, tratará de usar la CPU durante una unidad de tiempo. Posteriormente, se bloqueará durante 5 unidades de tiempo. Finalmente, volverá a requerir la CPU durante 4 unidades de tiempo y finalizará su ejecución.

Ejercicio 3: escribe un fichero de entrada que simule el conjunto de tareas del ejercicio 7 de la hoja de problemas. Ejecuta el simulador para resolver los apartados *b,c* y *d* de dicho problema.

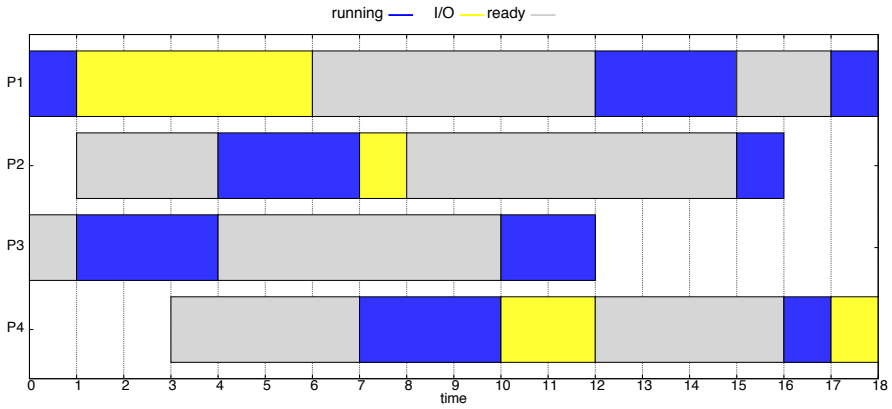
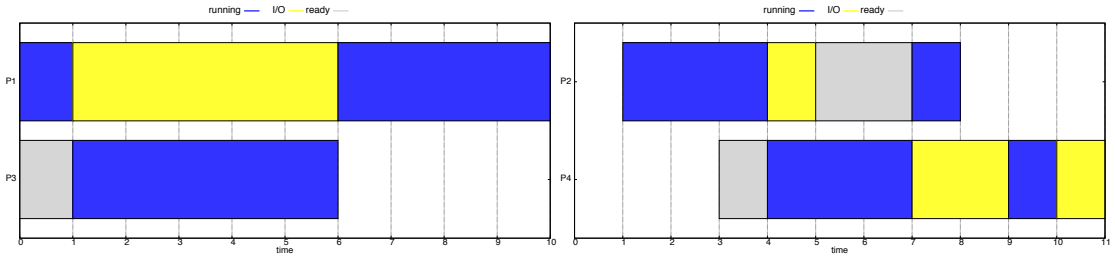


Figura 3.1: Resultado de simular el fichero example1.txt con una sola CPU



(a) Planificación en CPU 0 (b) Planificación en CPU 1

Figura 3.2: Simulación de example1.txt con 2 CPUs