Sistemas Operativos 1 – Trabalho Final

Primeira Parte

Tenha em conta uma arquitetura com o modelo de 5 estados, construa um escalonador Round Robin com quantum configurável (default: 3) com as seguintes características:

- O Sistema tem um número máximo de processos em execução simultânea (configurável, default: 6)
- Cada processo é constituído por um conjunto de instruções:

Cod.	Instrução	Descrição
1	CPU	Executa um cálculo no CPU
2	DISK	Executa um acesso ao disco (I/O)
3	GOTOBEGIN10	Salta para o inicio do programa (PC->0)
4	FORK	Duplica o processo. O novo processo vai para estado Ready
0	EXIT	Termina a execução do programa

Exemplo de programa: 1 1 1 2 2 1 2 1 0

- Cada programa reside num ficheiro individual, ex: *prog1.in* *prog2.in*
- O sistema devera ficar a correr na consola e à espera que o utilizador insira novos programas (nome dos ficheiros acima referidos)
- O sistema não para de correr caso todos os processos cheguem ao estado exit, deve ficar à espera de novos programas.
- O output deve ser guardado num ficheiro *scheduler.out* com o seguinte formato:

  INSTANTE | ESTADO_P1 | ESTADO_P2 | . . . | ESTADO_Pn

  Exemplo:

  . . .
  9 | exit | exit | ready | block | run | block | block | new
  10 | exit | exit | run | ready | exit | block | block | ready
  11 | exit | exit | ready | run | exit | block | block | ready | new
  . . .

- Podemos observar o output em "direto" noutra consola com o comando *tail -f schedule.out*
- O tempo é definido em unidades discretas: 0, 1, 2, 3, 4, 5, …
- Quando um processo tem a instrução DISK necessita de fazer 1 ciclo de CPU e salta então para o estado BLOCKED para aceder a Disco, ficando nesse estado durante 3 ciclos.
- Se já houver outro processo no estado BLOCKED, o novo processo ficará em fila de espera, até entrar efetivamente em BLOCKED durante 3 ciclos.

- A instrução GOTOBEGIN10 salta para o início do programa, e efetua esse salto 10 vezes. Após essas 10 vezes essa instrução espira e deixa de ter efeito.
- Quando um novo processo chega ao sistema, o escalonador deve-lhe criar um PCB e adicionar o processo à fila do estado NEW
- O estado READY só permite no máximo 4 (configurável) processos. Limitação valida apenas para processos vindos do NEW.
- A prioridade de entrada no READY é:
    - 1º RUNNING
    - 2º BLOCKED
    - 3º NEW (sujeito à limitação acima descrita)
- As filas inicialmente estão todas vazias e serão todas FIFO
- Quando um processo termina, passa para o estado EXIT

Entrega da Primeira Parte

O trabalho será realizado individualmente ou por **grupos de 2 pessoas**.

A entrega será realizada no Moodle até **dia 21 de Maio às 23h59** e deve conter um ficheiro zip ou rar com o seguinte nome: *XXXXX_YYYYY_SO1T1.zip* (sendo XXXX e YYYY os números de aluno) e conter o código fonte do programa e o relatório em Pdf.

A **discussão** do trabalho será realizada dia **22 de Maio** durante as aulas práticas.

Segunda Parte

Vai existir um único array com 50 posições a simular a memória do sistema para guardar as instruções de cada processo. Cada posição do array poderá conter apenas uma única instrução.

Dessa forma, após as instruções serem lidas do ficheiro devem ser guardadas nesse array único, e o PCB deve ser alterado para conseguir lidar com essa alteração.

NOTA: Fica proibido o PCB conter as instruções do processo.

A inserção no array da memória deverá ser implementada de duas maneiras:

Best Fit

Worst Fit

E deverá haver um parâmetro para poder selecionar o modo de operação.

O output deverá ser alterado para a seguir ao print de cada ciclo, conter o print do array da memória.

Como no exemplo seguinte:

```
. . .
9   | exit | exit | ready | block  | run | block | block | new
MEM | 1 1 1 2 3 4 0 1 1 1 2 0 x x x x x x x 1 2 1 2 1 3 4 3 0 x x x x x x …
10 | exit | exit |   run   | ready | exit | block | block | ready
MEM | 1 1 1 2 3 4 0 1 1 1 2 0 1 1 1 1 1 2 1 0 1 2 1 2 1 3 4 3 0 x x x x x x …
11 | exit | exit | ready |   run   | exit | block | block | ready | new
MEM | 1 1 1 2 3 4 0 1 1 1 2 0 1 1 1 1 1 2 1 0 1 2 1 2 1 3 4 3 0 1 1 1 1 2 1 0 …
. . .
```

Nota: O x representa os espaços livres na memória.

No caso do array de memória ficar cheio, por ocorrência de um fork ou por o utilizador inserir mais um processo, o sistema deve enviar para o output a mensagem

Memória cheia – impossível criar processo

Entrega da Segunda Parte

O trabalho será realizado pelos mesmos grupos da primeira parte.

A entrega será realizada no Moodle até **dia 18 de Junho às 23h59** e deve conter um ficheiro zip ou rar com o seguinte nome: *XXXXX_YYYYY_SO1T2.zip* (sendo XXXX e YYYY os números de aluno) e conter o código fonte do programa e o relatório em Pdf.

A **discussão** do trabalho será realizada dia **19 de Junho** em hora e local a anunciar futuramente.