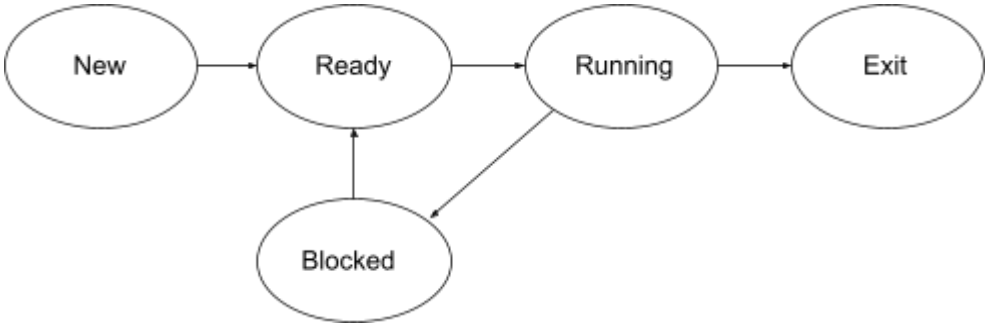
Sistemas Operativos

Trabalho Prático - Parte 2

Este trabalho vem na sequência do trabalho 1. O trabalho pode ser realizado individualmente ou em grupos de dois ou três alunos. No Moodle deverá submeter um .zip com os números de aluno no nome do ficheiro, ex “14444_22222.zip” e deverá conter o código fonte do trabalho assim como um relatório em PDF. O trabalho está dividido em 2 partes: parte 1) Implementar a instrução UNBLOCK; e parte 2) ler o input do simulador a partir de um ficheiro. Para a avaliação, o grupo pode optar por desenvolver apenas a parte 1, ou as duas partes. Os pesos máximos considerados para a nota final são 17 e 20, respectivamente.

Parte 1: Instrução UNBLOCK

Desenvolva um simulador de um modelo de 5 estados: NEW, READY, RUN, BLOCKED, EXIT. Quando os processos estão em READY ou BLOCKED, estão em filas de espera do tipo FIFO (first in first out).



Implemente a comunicação entre processos, permitindo que um processo bloqueado seja desbloqueado (UNBLOCK) por outro processo. Por exemplo, o processo 1, no estado RUNNING, poderá desbloquear o processo 2, se este se encontrar no estado BLOCKED, passando no próximo instante para o estado READY.

Note que a instrução para desbloquear o processo 2, não interfere na sequência de execução do processo 1, que poderá permanecer em execução, caso ainda tenha instantes para correr.

A instrução UNBLOCK, quando existir, deve correr sempre em paralelo com o primeiro instante em RUNNING do processo que dá a ordem de desbloqueio. Quando um processo recebe uma ordem para ser desbloqueado, este deve ser desbloqueado no próximo instante de tempo, mesmo que ainda tenha instantes no estado BLOCKED.

Os processos são definidos por um conjunto de pseudo-instruções-máquina representadas por números, que representam alternadamente:

- o tempo que o processo fica no CPU (estado RUNNING);
- o processo a ser desbloqueado (UNBLOCK). Se não existir nenhum processo a desbloquear, deve ser usado o número -1;
- o tempo que gasta no estado BLOCKED, **depois de chegar à fila de BLOCKED.**

Por exemplo: 0, 3, 2, 5, 3, -1. 2, 1, 0, 0 significa que o processo (quando chegar a sua vez) fica:

- O instante inicial do processo é 0;
- 3 instantes no CPU (RUNNING);
- UNBLOCK p2;
- 5 instantes no BLOCKED; seguido de
- 3 instantes no CPU (RUNNING);
- Nenhum processo é desbloqueado (-1);
- 2 instantes no BLOCKED;
- 1 instante no CPU (RUNNING) seguido de
- EXIT

Note que a partir de certo instante de tempo, o programa tem sempre zeros (um ou mais zeros). Isso significa que após o último valor diferente de zero (o qual representa tempo de CPU) o processo passa para o estado EXIT e termina.

Parte 2: Ler input de um ficheiro

A parte 2 do trabalho consiste em ler o input do simulador a partir de um ficheiro. O formato do ficheiro.

Testes

```
int programas[3][10] = {
      {0, 3, -1, 2, 2, 3, 4, 2, 2, 0} ,
      {1, 4, 1, 4, 5, 1, 4, 3, 3, 0} ,
      {2, 2, 2, 6, 2, -1, 4, 2, 0, 0 }
 };
```

```
int programas[8][10] = {
      {0, 2, 2, 4, 5, 1, 1, 1, 0, 0} ,
      {0, 1, -1, 2, 2, -1, 3, 3, 0, 0 } ,
      {1, 2,-1, 1, 6, -1, 4, 1, 0, 0 } ,
      {2, 2, 2, 1, 5, 2, 4, 2, 0, 0 } ,
      {2, 3, 3, 1, 2, 1, 1, 1, 0, 0 } ,
      {3, 4, 4, 2, 7, 4, 2, 2, 0, 0 } ,
      {4, 4, 4, 2, 4, 3, 4, 3, 0, 0 } ,
      {5, 5, 3, 6, 5, 1, 3, 5, 0, 0 }
};
```

Output:

```
int programas[3][10] = {
      {0, 3, -1, 2, 2, 3, 4, 2, 2, 0} ,
      {1, 4, 1, 4, 5, 1, 4, 3, 3, 0} ,
      {2, 2, 2, 6, 2, -1, 4, 2, 0, 0 }
 };
```

Instante	proc1	proc2	proc3
0	NEW		
1	RUN	NEW	
2	RUN	READY	NEW
3	RUN	READY	READY
4	BLOCK	RUN	READY
5	READY(UNBLOCK)	RUN	READY
6	READY	RUN	READY
7	READY	RUN	READY
8	READY	BLOCK	RUN
9	READY	READY(UNBLOCK)	RUN
10	RUN	READY	BLOCK
11	RUN	READY	READY(UNBLOCK)

12	BLOCK	RUN	READY
13	READY(UNBLOCK)	RUN	READY
14	READY	RUN	READY
15	READY	RUN	READY
16	READY	RUN	READY
17	READY	BLOCK	RUN
18	READY	BLOCK	RUN
19	RUN	BLOCK	BLOCK
20	RUN	READY(UNBLOCK)	BLOCK
21	EXIT	RUN	BLOCK
22		RUN	READY(UNBLOCK)
23		RUN	READY
24		EXIT	RUN
25			RUN
26			EXIT