Enunciado do trabalho-1 de Sistemas Operativos

Ano letivo 2024/25 - Universidade de Évora

Pretende-se implementar, usando a linguagem C, um simulador de Sistema Operativo considerando um modelo de 5 estados. O modelo deve incluir os seguintes estados (por conveniência entre parênteses são apresentados estados equivalentes nos modelos estudados): NEW, READY, RUNNING, BLOCKED e EXIT. Quando os processos estão em NEW, READY, e BLOCKED, estão em filas de espera do tipo FIFO (first in first out).

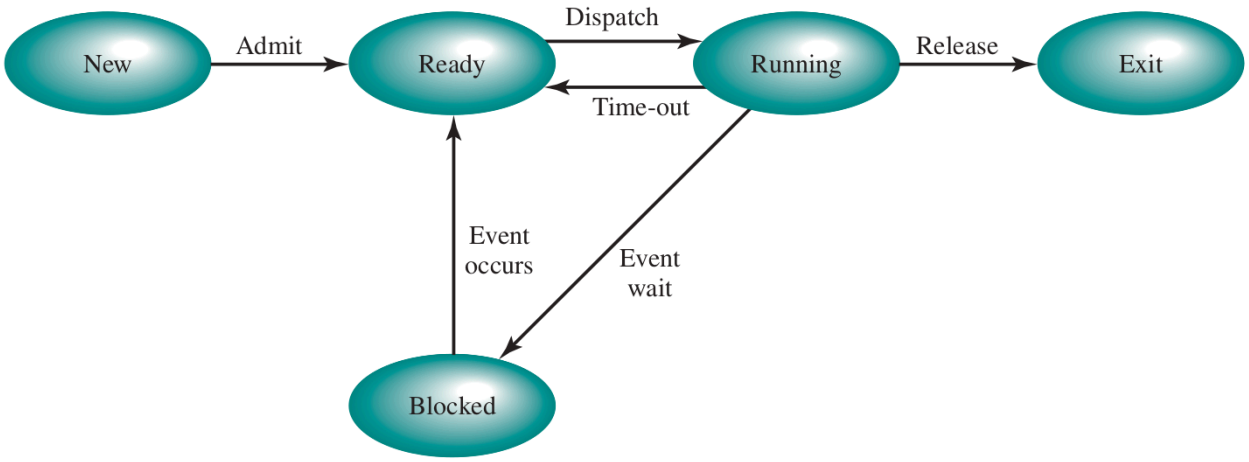

Figure 3.6 Five-State Process Model

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

– Instruções de 101 a 199 indicam instruções JUMP, que fazem com que o Program Counter do processo ande para trás *n* instruções, em que *n* é corresponde aos dois dígitos menos significativos. Por exemplo, a instrução 103 indica que o Program Counter vai recomeçar na 3ª instrução anterior, 101 indica a instrução imediatamente anterior à instrução JUMP.

Nota: Pode assumir que o JUMP é sempre um salto para trás, e não sai fora do programa. Ou seja, o PC nunca é alterado para um número negativo.

– Instruções de 201 a 299 indicam instruções EXEC, que executam um novo processo. Os dois dígitos menos significativos indicam o identificador do programa. Por exemplo, a instrução 204 lança um novo processo que executa o programa número 4.

– As instruções identificadas por números inteiros negativos são instruções de I/O que executam durante 1 instante no CPU e bloqueiam o processo, ou seja, o seu estado muda para BLOCKED. O processo é bloqueado durante um número de instantes igual ao módulo do identificador da instrução. Por exemplo, a instrução -5 executa um instante no CPU e depois bloqueia o processo durante 5 instantes.

– A instrução HALT, que é representada por um 0 (zero). Esta instrução executa durante um instante no CPU, e depois o processo passa para o estado EXIT. O processo fica no estado EXIT durante 1 instante, e depois termina.

– Deste modo todas as instruções executam durante 1 instante no CPU. Isto inclui instruções negativas (I/O que obrigam a mudar de estado para BLOCKED); o zero (instrução HALT); e positivas (instruções de JUMP, de EXEC, e quaisquer outras).

Por exemplo 2,202,-5,1,-3,0... significa que o processo (quando chegar a sua vez) …

1. corre a instrução com o código 2 por um instante;
2. corre a instrução com o código 202 (que também cria um novo processo com o programa número 2) por um instante;
3. corre a instrução de I/O com o código -4 por um instante e bloqueia;
4. fica bloqueado durante 4 instantes;
5. após os 4 instantes sai de BLOCKED e vai para o fim da fila de READY.

Quando é novamente a sua vez de correr no CPU, o processo

1. corre a instrução 1 por um instante
2. depois corre a instrução -3 por um instante e bloqueia
3. fica bloqueado durante 3 instantes.
4. após os 3 instantes sai de BLOCKED e vai para o fim da fila de READY.

Quando volta ao CPU, o processo,

1. corre a instrução 0 (HALT) por um instante e passa ao estado EXIT.
2. fica no estado EXIT 1 instante e termina

Resumindo, o processo corre 3 instantes seguidos no CPU (RUNNING), interrompe 5 instantes. Quando volta a correr, corre 2 instantes e interrompe 3 instantes. Quando volta, corre 1 instante e termina.

Se em vez de encontrar um HALT, o processo terminar em **103**, por exemplo, 2,202,-5,1,-3,**103.** Quando corre a instrução 103 executa um instante e depois salta 3 instruções para trás e executa a instrução -5 (neste caso teríamos um loop infinito).

Considere os seguintes 3 pseudo-programas. Neste sistema é sempre iniciado um processo no instante 1 executando o 1º programa (1ª coluna). Os restantes poderão ser executados a partir do primeiro usando as instruções EXEC.

```
int programas[5][20] = {
{203,    4,     2},
{   1,   -3,  202},
{  -2, 102,     5},
{   1,    0,    10},
{   0,    0,     0}       };
```

Implemente um simulador dos processos neste sistema tendo em consideração que:

1) Os processos quando são criados entram no estado NEW e passados 2 instantes passam para o estado READY.

2) O id dos processos é atribuído incrementalmente a começar no 1, à medida que são criados, ou seja, o primeiro processo a ser criado tem o id 1, e o seguinte o id 2. Mesmo que um processo termine, o seu id não poderá ser reutilizado.

3) Os instantes na fila BLOCKED, para cada processo, contam assim que o processo chega à fila.

4) Os processos quando terminam, passam do CPU (estado RUNNING) para o EXIT, onde permanecem 1 instante de tempo.

5) Os processos depois de estarem 1 instante de tempo em EXIT desaparecem do sistema.

6) O escalonamento a implementar deve ser o algoritmo Round Robin com o Quantum de 3.

7) Os processos que saem de BLOCKED, passam para READY.

8) O número máximo de processos a dar entrada no sistema é de 20, e cada programa tem a dimensão máxima de 11 instruções. No caso de haver uma chamada a uma instrução EXEC e o limite de 20 tiver sido atingido, o novo processo não é criado, a função EXEC comporta-se como qualquer outra instrução positiva.

9) Se no mesmo instante puderem entrar vários processo na fila de READY vindos de RUNNING, BLOCKED e NEW estes entrarão na fila de READY pela seguinte ordem: 1º os de BLOCKED, 2º os de NEW e 3ª o processo de RUNNING/CPU.

10) O simulador corre no máximo durante 100 instantes.

11) O programa deve ter como output, o estado de cada processo em

cada instante. Por exemplo:

time inst	proc1	proc2	proc3	…	proc20
1	NEW				
2	NEW				
3	RUN				
4	RUN	NEW			
5	RUN	NEW			
7	BLOCKED	RUN			
8	BLOCKED	RUN	NEW		
9	READY	RUN	NEW		

...etc...

Este trabalho deve ser desenvolvido em grupos de 2 ou 3 alunos até às 23:59 de dia 22 de Abril, e submetido no Moodle como um ficheiro .zip com os números de aluno no nome do ficheiro, ex “l4444_l5555.zip” e deverá conter o código fonte do simulador, os outputs dos testes a disponibilizar no Moodle e um relatório em PDF. O código fonte deverá ser escrito na linguagem C e incluir um ficheiro README instruções, um makefile ou shellscript que permitam a compilação. Os outputs devem estar em ficheiros “outputXX.out” em que XX é o número do teste de input, por exemplo, para o input número 3, deverá ser gerado o ficheiro “output03.out”. O relatório deve conter uma descrição sucinta da estrutura do programa proposto (máximo 4 pág).