UNIVERSIDADE DE ÉVORA

Relatório de Sistemas Operativos 1

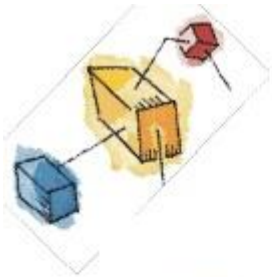
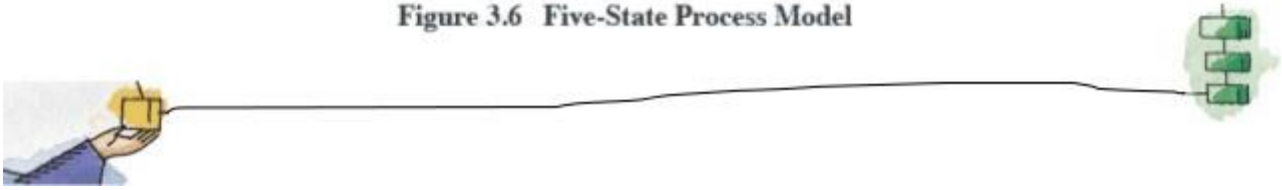


Figure 3.6 Five-State Process Model



Trabalho realizado por:

Ruben Bravo nº37548;
Jailson Varela nº40699;
Marcelo Feliz nº38073.

Professor: Luís Rato
Professor: Nuno Miranda

Introdução:

No âmbito do trabalho final da cadeira de Sistemas Operativos 1, usando a linguagem C, para implementar uma arquitetura sobre o modelo de 5 estados que consome programas constituídos por um conjunto instruções, 12 mais especificamente, instruções essas que são codificadas por 3 números inteiros, cada um tendo o seu significado. Usando o escalonamento Round Robin com Quantum de 4.

Este trabalho foi dividido em duas partes, em que na primeira parte foi pedido para implementar o escalonador e 3 das instruções do programa sendo elas a 8, que guarda uma variável no disco, a 9, que carrega a variável do disco para uma variável, e a 11, que termina o processo; e na segunda e ultima parte do trabalho foi nos pedido para implementar o resto das instruções e os gestores da memoria: Best Fit e Next Fit.

Constituição do programa

Para a funcionamento deste programa tivemos que criar algumas funções e estruturas de entre elas:

- A função **new_Queue()** que cria e inicializa uma nova Queue;
- as funções **enqueue()** e a **dequeue()** adicionam e removem respetivamente um processo da Queue;
- a função **front()** que retorna o elemento que está na head da Queue;
- a função **New_Dados()** que cria um novo elemento dessa struct;
- as funções **New_PCB_BestFit()** e **New_PCB_NextFit()**, criam um novo elemento da struct PCB conforme a gestão de memoria for Best Fit ou Next Fit;
- as funções **controlo_chegada_block()**, **controlo_chegada_exit() e controlo_chegada_new()** são semelhantes no sentido que todas verificam a chegada de processos nos respetivos estados;
- a função **controlo_instrucao()**, essa função, em cada instante executa a instrução a ser realizada pelo processo que se encontra no RUN;
- a função **print()**, em cada instante, dá print do estado atual de cada processo;
- a função de **controlo_estado()** que cria e controla o estado de cada processo durante toda a execução do programa;

Conclusão:

Em suma, após a realização do trabalho sobre a simulação de um escalonador de 5 estados podemos concluir que o trabalho apresenta todas as instruções codificadas e gestores de memória necessárias ao seu funcionamento, contudo existe um erro no último teste que não foi possível resolver. As maiores dificuldades foi a gestão da memoria e a instrução fork. É importante destacar que com este trabalho conseguimos compreender melhor como funciona o modelo de 5 estados.