Sistemas Operativos 1

UNIVERSIDADE DE ÉVORA

Trabalho Final

Escalonador Round-Robin

Alunos Henrique Raposo nº 33101
Tiago Reis nº 33205

Licenciatura de Engenharia Informática
Ano Letivo 2016/2017

Conteúdo

1 Introdução

No âmbito da disciplina de Sistemas Operativos 1 foi proposto a criação de um simulador de um escalonador Round-Robin com algumas caracteristicas especificas.

Este trabalho foi desenvolvido utilizando a linguagem de programação C com a utilização de algumas bibliotecas extra necessárias para que conseguia simular um escalonador com o algoritmo pretendido.

Foi implementado queues para simular os estados do programa e também foi utilizada a struct programa para simular o processo e o PCB.

2 Round-Robin

O escalonador funciona segundo o modelo de 5 estados, exemplificado na figura 1. Foram implementadas todas as funções e estados presentes na figura.

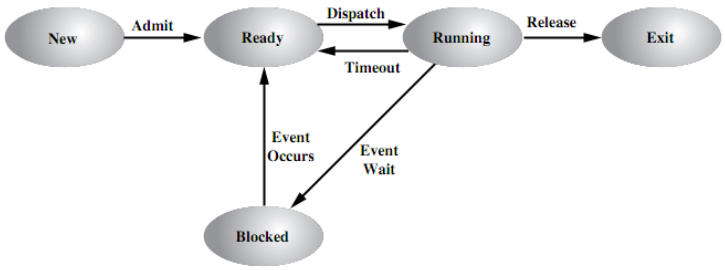

Figura 1: Modelo de 5 Estados

2.1 Configurações

As configurações default utilizadas para o escalonamento foram:

- Quantum: 3
- Nº Máximo Processos: 6
- Nº Máximo Processos Ready: 4

Estas configurações foram definidas como variáveis com este valor em default e podem ser alteradas pelo utilizador ao inicio quando se corre o programa, sendo pedido um input com o novo valor na consola.

Caso o utilizador insira um novo valor, estas configurações serão substituidas pelo valor inserido. Caso o utilizador digite 0 na consola, é assumido o valor default para essa configuração.

3 Programa

Nesta secção está explicada o funcionamento das funções do programa que são aquelas que não fazem parte do modelo de 5 estados. Estas funções são "auxiliares"e sem elas não era possivel o correto funcionamento do escalonamento.

3.1 main

A função main começa por pedir um input ao utilizador para as configurações do round-robin e inicia as queues com os valores do input fornecido pelo utilizador. Cria também as threads para estar constantemente a pedir o nome do programa ao utilizador.

Existe também um ciculo while que faz os 5 principais estados.

3.2 ask_input

Esta função pede um input do nome do ficheiro do programa a executar.

Se o nome do ficheiro tiver a extenção '.in', o nome é aceite e o programa procede para a função le_ficheiro.

Se o nome do ficheiro não conter a extenção .in é considerado "extensão inválida", pelo que o programa volta a pedir ao utilizador que insira um input correto.

3.3 le_ficheiro

Esta função funciona como uma verificador para se o ficheiro existe ou não.

Se o ficheiro não existir, a variável ficheiro retorna NULL, é impresso uma mensagem de "Ficheiro não encontrado"e o programa volta a pedir um input ao utilizador.

Se o ficheiro existir, é criado um novo programa e as instruções contidas no ficheiro do programa são lidas e transformadas em instruções do programa na estrutura.

3.4 output

Trata de verificar a cada instante qual o estado dos programas e cria um array "outputS"em que cada elemento representa o estado atual de um programa, sendo este estado representado por numeros, em que:

- 0 - New
- 1 - Ready
- 2 - Run
- 3 - Blocked
- 4 - Exit

Após criar o array, a função invoca a função escreve_ficheiro que recebe o array "outputS"e faz a escrita do ficheiro.

3.5 escreve_ficheiro

A função escreve_ficheiro faz a escrita do output para um ficheiro chamado "scheduler.out". Esta função tambem traduz o resultado do array, recebido como argumento, que apenas contem o numero dos estados, para o nome dos estados do modelo.

3.6 read_program

Esta função trata apenas de colocar as instruções no programa, retirando os espaços.

4 Simulador

Nesta secção é explicado o funcionamento das funções do modelo de estados.

4.1 admit

Função que permite que um processo entre no estado ready.

4.2 dispatch

Esta função envia os processos para o estado run e coloca o run-counter a zero.

4.3 timeout

Esta função serve para fazer um dequeue na queue do Run. Isto significa que retira o processo do run e envia para o ready quando esta função é chamada (essa chamada é feita na função run quanto o quantum é feito).

4.4 event_wait

Função que faz o dequeue no Run, ou seja, retira o processo do run e coloca-o no blocked.

Esta é invocada no Run quando é executada uma instrução de acesso ao disco.

4.5 event_occurs

Retira o processo do Blocked, fazendo o dequeue e coloca o return do Nó retirado do Blocked no Ready.

4.6 release

Trata de enviar o processo do Running para o Exit (fazendo o dequeue do estado Run e colocando este em Exit).

4.7 run

O run é o estado 2 no nosso simulador.

Esta função tem uma condição para cada instrução (de 0 a 4), sendo que vai lendo as intruções do programa e consoante a instrução lida vai fazer o que é suposto a instrução fazer.

Após correr a instrução, a função incrementa o PC (Program Counter) consoante o que é suposto.

Esta função tem também a restrição do quantum, pelo que só permite que o programa esteja a correr durante o tempo definido ao inicio.

4.8 blocked

A função blocked obriga o processo a ficar 3 ciclos neste estado.

Assim que o blocked_counter chegar a 0 (passaram os 3 ciclos), o processo já pode sair do blocked e este processo é feito ao ser invocada a função event_occurs().

4.9 inst_fork

Esta função serve para simular um fork.

É criado assim um processo filho, com outro ID, com as instruções seguintes ao momento do fork.

Este programa é depois executado da mesma forma que os outros.

4.10 enter_new

Cria um novo Node para o processo que é admitido.

4.11 enter_ready

Função que verifica se existe espaço na queue do ready.

Caso exista, o processo entra na fila de espera do ready.

Caso não exista o processo vai ter de ficar à espera que exista espaço.

4.12 enter_blocked

Para inserir um processo no blocked, é verificado se existe espaço na Queue e se for é feito uma enqueue para o estado Blocked.