U.C. Sistemas Operativos
Licenciatura em Engenharia Informática

1º Trabalho - Modelo de 5 Estados

Docente: Luís Rato

Discentes: André Baião 48092
Gonçalo Barradas 48402
Guilherme Grilo 48921

abril 2022

Introdução

Um Sistema Operativo é um programa que tem como função controlar os recursos dos programas em execução. Este fornece uma interface entre os programas e o hardware da máquina. Existem vários tipos de sistemas, sendo abordado neste trabalho o sistema de multiprocessamento, que consiste em recorrer a mais do que um processador de maneira a controlar o dispositivo em questão, isto é, cada processador contém uma cópia do próprio Sistema Operativo e as cópias comunicam entre si de forma a coordenar as diferentes operações de cada uma. O Multiprocessamento divide cada tarefa em várias subtarefas e atribui cada uma destas a um processador disponível, tornando mais rápida a execução das mesmas.

As vantagens de utilizar um Sistema Operativo com Multi-processamento são a confiabilidade, a rapidez do processo e a diminuição do custo, pois as tarefas podem ser distribuídas por todos os processadores e no caso de um processador falhar, a subtarefa é transferida para outro processador, quanto maior o número de processadores, mais tarefas conseguem ser executadas em menos tempo e como os sistemas que contém mais de um processador partilham periféricos, memória, etc..., entre si, estes são relativamente mais baratos do que sistemas com apenas um processador. Um processo pode ser classificado como um programa ou instância de um programa em execução, uma entidade que pode aceder ou ser executada no processador ou com uma atividade caracterizada por uma única sequência de *threads* de execução, um estado atual e ainda um conjunto de recursos associados.

O modelo escolhido para o trabalho foi o modelo de cinco estados sendo estes os estados *New*, *Ready*, *Running*, *Exit* e *Blocked*. Estes estados apresentam como ou onde se encontra o processo, isto é, ou está preparado para ser executado, ou está a ser executado, se já terminou, bloqueado ou se é um novo processo, tendo em atenção que quando um processo está num dos estados (*Ready* ou *Blocked*) tem de estar contido numa *queue* (fila de espera do tipo FIFO), quando o processo se encontra no estado *Run* o escalonamento dos estados deve ser executado com o algoritmo *Round Robin*). Este atribui uma porção de tempo (*quantum time*) igual a cada processo para que este possa ser executado no processador (estado *Run*) e, se ao fim desse tempo não for finalizada a execução desse processo, este passa para o estado *BLOCK*)e outro processo entra para o processador para se proceder à sua execução.

Posto isto, o objetivo é implementar um simulador de um Sistema Operativo considerando o modelo de 5 estados acima mencionado, utilizando a linguagem de programação C.

Programa

Para este simulador são usadas três *structs*:

- **so** : Guarda todas as informações necessárias para o funcionamento do modelo de 5 estados, como os vários programas, *quantum time* e as *queues* que irão ser utilizadas;
- **program** : Guarda todas as informações sobre um programa, como o estado em que se encontra, o ciclo do programa e a sua posição;
- **queues** : Guarda todas as informações das diferentes *queues*;

O nosso programa é composto por 11 funções:

- **isEmpty**
  Esta função recebe como argumento uma *queue* e verifica se uma *queue* não contém programas;
- **peek**
  Esta função recebe como argumento uma *queue* e devolve o primeiro programa da *queue*;
- **dequeue**
  Esta função recebe como argumento uma *queue* e remove o primeiro programa da *queue*;
- **enequeue**
  esta função recebe como argumento uma *queue* e um *programa* e coloca esse programa na respetiva *queue*

- **getState**
  Esta função recebe como argumento o id do programa e devolve o estado em que o programa se encontra;
- **setState**
  Esta função recebe como argumentos o id do programa e um estado e altera o estado do programa para o estado passado no argumento;
- **getCycle**
  Recebe como argumentos o id do programa e a posição do ciclo e devolve o valor do ciclo naquela posição;
- **getNow**
  Esta função recebe como argumento o id do programa e devolve a posição do ciclo em que o programa se encontra;
- **changeProgram**
  Esta função desempenha um papel muito importante no nosso Sistema Operativo, pois esta é responsável por alterar o estado dos nossos programas. Após receber o id do programa, esta vai analisar o estado em que o programa se encontra e determina o novo estado do programa.
- **quantumChangeProgram**
  Apenas usada quando o método de escalonamento é o *Round-Robin*. É responsável por alterar o programa que atualmente se encontra no estado *Run* por outro programa da *queue ready*.
- **printProgramState**
  Esta função recebe como argumento o id do programa e imprime o estado atual em que o programa se encontra.
- **run**
  É fundamental para o funcionamento do modelo de 5 estados. É responsável pela execução dos nossos programas.

  Ao longo da sua execução vai imprimindo os instantes e os estados dos vários programas (recorrendo á *printProgramState*). Submete ainda cada programa a um conjunto de condições com o objetivo de perceber se é necessário alterar o estado atual de cada programa.
- **main**
  Responsável por inicializar todas as estruturas necessárias para a execução dos programas.

Conclusão

A realização deste trabalho, permitiu consolidar e ampliar os nossos conhecimentos acerca de Sistemas Operativos, mais concretamente sobre os modelos de 5 estados, como estes funcionam e como são importantes para manter o bom funcionamento do mesmo.

Concluímos ainda que o *Round-Robin* é um método vantajoso quando utilizado em processos mais curtos, mas quando surgem processos mais longos este revela-se ineficiente, devido ao facto de que se o *quantum time* for inferior ao tempo de duração do processo, a execução deste é interrompida prematuramente.

Bibliografia

Rato, Luís in "Aulas de Sistemas Operativos". 2022 in Universidade de Évora.

Rato, Luís in "Trabalho1-SO-2022.pdf". Universidade de Évora's MOODLE.

`https://www.e-konomista.pt/o-que-e-um-sistema-operativo/`

`https://science.jrank.org/programming/Multiprocessing_Operating_Syst.html`

`https://www.javatpoint.com/multiprocessing-operating-system`