U.C. Sistemas Operativos
Licenciatura Engenharia Informática

2º Trabalho

Docente: Luís Rato

Discentes: André Baião 48092
Gonçalo Barradas 48402
Guilherme Grilo 48921

1 de junho de 2022

Introdução

O presente trabalho consiste em criar um simulador de um Sistema Operativo que utilize o Modelo de 5 estados, recorrendo ao uso de *Threads* e à gestão de memória através do método de segmentação de memória do Sistema Operativo.

Uma *Thread* pode ser definida como um caminho de execução dentro de um processo. Um só processo pode conter múltiplas *Threads*. Com a utilização de *Threads* pretendemos tornar a execução de um determinado processo mais rápida e eficiente. As vantagens de dividir um determinado processo em várias *Threads*, para além da rapidez e eficiência, é a superior capacidade de resposta, pois quando uma *Thread* termina a sua execução, o seu *output* pode ser imediatamente devolvido; a partilha de recursos entre as *Threads* como código, dados ou até mesmo ficheiros; a utilização mais eficiente de um sistema multiprocessamento, devido a possuirmos múltiplas *Threads* num processo, então podemos também utilizar várias *Threads* em múltiplos processos, o que torna a execução do processo muito mais rápida e eficiente, etc.

Existem vários métodos de gestão de memória sendo paginação e segmentação os métodos mais utilizados. A paginação é um método mais simples onde apenas é necessário encontrar uma página livre. A segmentação é um método mais complexo devido ao facto de os tamanhos dos segmentos ser variável, este vai ser o método de gestão de memória abordado neste trabalho.

Segmentação da memória é um método de gestão de memória do Sistema Operativo que consiste em dividir a memória principal de um computador em segmentos e cada segmento pode ser alocado a um processo. A tabela de segmentação é uma tabela que guarda todas as informações de todos os segmentos e é responsável por mapear o endereço lógico bidimensional em apenas um endereço físico unidimensional. Para definir a escolha dos segmentos são utilizados vários algoritmos, tais como, First-Fit, Best-Fit , Next-Fit, entre outros, o algoritmo que nos foi proposto para ser utilizado foi o Next-Fit.

Memória

Para representar a memória foi criado um array de dimensão 200 e utilizamos um array auxiliar, com a mesma dimensão, cujo objetivo é indicar se uma determinada posição da memória se encontra ocupada no momento.

De forma a implementar o Next-Fit é necessário ter duas variáveis, uma que indique o espaço de memória livre e outra que indique a última posição de memória utilizada.

Programa

Para a facilitar na implementação do nosso programa, utilizámos três structs:

- `runner`
  Guarda as informações necessárias para o funcionamento de todos os programas.
- `Process`
  Aqui são guardadas todas as informações acerca de cada processo, inclusivé as threads de cada um.
- `Program`
  As informações relativamente a cada programa são todas guardadas nesta struct.

Por fim temos as funções que são responsáveis por garantir que o nosso simulador vai de encontro ao pedido pelo docente:

- `getMax`
  Devolve o máximo entre o valor máximo atual e o valor da instrução em análise.

- `printMemory`
  Responsável por imprimir o *array* da memória, o *array* de bits e ainda o espaço livre.
- `removeProcess`
  Responsável por remover o processo desejado da memória. Primeiramente são removidas todas as threads do processo (no caso deste possuir alguma(s)) e, após isso, é libertado da memória.
- `allocateThread`
  Esta função serve para alocar uma thread na memória.
- `allocate`
  Aloca um processo na memória.
- `getInstructionID`
  Esta função retorna o ID de uma determinada instrução.
- `readFile`
  Esta função lê o ficheiro e guarda as respetivas instruções de cada programa.
- `getNumOfPrograms`
  Esta função lê o ficheiro e retorna o número de programas presentes no mesmo.
- `executeThread`
  Executa a instrução de uma determinada thread.
- `executeProgram`
  Executa a instrução de um determinado programa.
- `canProced`
  Verifica se um determinado processo, que se encontra à espera de uma ou mais threads, pode prosseguir.
- `blocked2Ready`
  Serve para verificar se um determinado processo pode transitar de BLOCKED para READY.
- `newProcess`
  Verifica se é possível inicializar um novo processo.
- `new2Ready`
  Serve para que todos os processo que se encontrem no estado NEW transitem para o estado READY.
- `run2exit_blocked_run`
  Executa o processo e verifica o quantum time. E caso a instrução a executar seja PRNT, descobre o valor a imprimir.
- `ready2run`
  Caso nenhum processo se encontre no estado RUN, significa que é possível colocar lá um processo.
- `exit2finish`
  Serve para colocar no estado FINISH todos os processos que se encontrem no estado EXIT.
- `runner`
  Responsável pela execução dos processos, por imprimir os instantes, os estados e quais os processos que neles se encontram. É ainda responsável por lidar com qualquer tipo de exceção ou anomalia que ocorra durante a execução do programa.

Conclusão

A realização deste trabalho foi um pouco mais complexa do que aquilo que pensámos e por isso houve algumas complicações. Contudo foi-nos possível perceber e realizar a implementação de um programa utilizando o algoritmo de alocação Next-Fit.

A pesquisa para este trabalho e a sua realização ajudou a consolidar os conhecimentos já adquiridos e entender melhor o funcionamento destes algoritmos.

Referências

[Rato, 2022] Rato, L. (2022). Aulas de sistemas operativos. *Universidade de Évora*.