UNIVERSIDADE DE ÉVORA

Desenvolvimento de um simulador de sistema operativo

U.C. Sistemas Operativos

Docentes: Pedro Salgueiro, Daniela Schmidt
Discentes: Helder Godinho 42741, Gustavo Oliveira 46395, Mariana Silva 54389

1 de maio de 2023

1 Introdução

Neste trabalho foi proposto a implementação de um simulador de sistema operativo considerando um modelo de 5 estados. Estes estados consistem em **NEW**, **READY**, **RUNNING**, **EXIT**e **BLOCKED**. Cada um destes estados representa 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.
Quando um processo está num dos estados (**READY** ou **BLOCKED**, tem de estar contido numa **queue** (fila de espera do tipo FIFO).
Quando um processo se encontra num estado **RUN** é atribuído uma porção de tempo para que este possa ser executado no processador, e no fim desse tempo, se o processo ainda não foi finalizado, este passa para o estado **BLOCKED**.

2 Desenvolvimento

Primeiramente começámos por esboçar uma ideia de como iríamos abordar o nosso problema e concluímos que de modo a implementar o simulador iríamos precisar de diversas **structs**.

As **structs** são as seguintes:

Struct cpuState: Representa o estado do CPU e contém as seguintes informações:

- **processPid**: Id do processo que está na CPU. Se nenhum processo estiver a ser executado, esse valor é -1.
- **processArrivalTime**: o tempo em que o processo atual chegou à CPU. Se a CPU estiver desocupada, esse valor é -1.

Struct queue: Representa uma estrutura de dados do tipo FIFO e contém as seguintes informações:

- **buffer**: um pointer para o buffer que armazena os dados da fila.
- **head**: o índice do primeiro elemento da fila.
- **tail**: o índice do último elemento da fila.
- **size**: o tamanho da fila.

Struct process: Representa um processo no sistema e contém as seguintes informações:

- **pid**: ID do processo.
- **state**: estado atual do processo (NEW, READY, RUNNING, BLOCKED ou EXIT).
- **arrival_time**: o tempo que o processo chegou ao sistema.
- **Nextcpu_time**: o tempo necessário para a próxima execução da CPU.
- **Remaining_io_time**: o tempo total que o processo fica bloqueado.
- **programCounter**: índice da instrução a ser executada.

De seguida, começámos a desenvolver as funções responsáveis pelo funcionamento do programa. Este é composto por 15 funções, sendo 6 destinadas ás queues e as restantes ao desenvolvimento do programa. As funções são as seguintes:

- **printState**: imprime o estado atual.
- **isInsideQueue**: verifica se um determinado valor está presenta em uma fila.
- **Init_runningProcess**: inicializa o estado da cpu.
- **createProcess**: cria um novo processo.
- **checkIfNew**: verifica se há novos processos a serem criados no momento atual e, se houver, adiciona à fila dos novos processos.
- **newToRunning**: verifica se há algum processo na fila de novos processos e, se houver, move-o para a CPU.
- **runningToBlocked**: muda o estado do processo de run para blocked.
- **blockedToReady**: muda o estado do processo de blocked para ready.
- **runningToExit**: muda o estado de run para exit.
- **readyToRunning**: muda o estado de ready para run.

3 Balanço crítico

Neste trabalho fomos confrontados com alguns obstáculos que condicionaram um pouco o desenvolvimento do trabalho, mais especificamente nas funçoes de **readyToRunning** e na **blockedToReady**. O problema surgiu na implementação das queues que não faziam corretamente a mudança de estado.

4 Conclusão

A realização deste trabalho revelou-se um desafio interessante que nos ajudou a perceber melhor o funcionamento de um sistema operativo.
Ao realizar este trabalho tivemos oportunidade de aplicar os nossos conhecimentos teóricos num problema prático e, assim, ganhar experiència em programação nesta linguagem.