

U.C. Sistemas Operativos Licenciatura em Engenharia Inform´atica

Modelo de 5 Estados

**Docente:** Pedro Sangueiro

Daniela Schmidt

**Discentes:** Gustavo Rebouças l46395

Helder Godinho

Mariana Silva l54389

julho 2023

# Introdu¸c˜ao

O presente relatório descreve o desenvolvimento de um simulador de um modelo de 5 estados: NEW, READY, RUN, BLOCKED e EXIT. O objetivo do simulador é demonstrar a comunicação entre processos, permitindo que um processo bloqueado seja desbloqueado por outro processo através da instrução UNBLOCK. O simulador também implementa filas de espera do tipo FIFO (first in first out) para os estados READY e BLOCKED.

A instrução UNBLOCK possibilita que um processo em execução desbloqueie outro processo que esteja no estado BLOCKED, fazendo com que o processo desbloqueado passe para o estado READY no próximo instante de tempo. É importante ressaltar que a instrução UNBLOCK não interfere na sequência de execução do processo que a emite, permitindo que o processo continue em execução caso ainda tenha instantes de tempo restantes.

Cada processo é definido por um conjunto de pseudo-instruções-máquina representadas por números. Os números alternam entre representar o tempo que o processo fica no estado RUNNING, o processo a ser desbloqueado (UNBLOCK) e o tempo que o processo gasta no estado BLOCKED após entrar na fila. Por exemplo, a sequência "0, 3, 2, 5, 3, -1, 2, 1, 0, 0" indica que o processo fica inicialmente no estado NEW, em seguida, passa por 3 instantes no estado RUNNING, emite a instrução UNBLOCK para o processo 2, permanece 5 instantes no estado BLOCKED, segue para mais 3 instantes no estado RUNNING e, por fim, conclui sua execução e entra no estado EXIT.

A segunda parte do trabalho consiste em ler o input do simulador a partir de um arquivo, no formato especificado. Essa funcionalidade permite testar diferentes conjuntos de pseudo-instruções-máquina e observar o comportamento do simulador em cada caso.

# Programa

Para este simulador são necessárias 3 structs:

* **cpuState:** armazena o estado atual do CPU e contém as seguintes informações:
  + **processPid:** o ID do processo que está na CPU. Se nenhum processo estiver a ser executado, esse valor é -1.
  + **state:** Estado atual da CPU (NEW, READY, RUNNING, BLOCKED, EXIT).
* **queue** : Implementação de uma fila como um buffer circula e contêm as seguintes informações:
  + **buffer:** um ponteiro para o buffer que armazena os elementos da fila.
  + **Head:** o índice do primeiro elemento na fila.
  + **Tail :** o índice do último elemento da fila.
  + **Size:** o tamanho máximo da fila.
* **process:** Representa um processo no sistema e contém as seguintes informações:
  + **PID:** ID do processo.
  + **State:** estado do processo (NEW, READY, RUNNING, BLOCKED ou EXIT).
  + **event:** Variável de controlo que indica o momento de um evento (processo bloqueado, pronto, etc.).
  + **programCounter:** Contador de programa para controlar a execução do programa.

O nosso programa ´e composto pelas seguintes funções:

* Funções de Fila:

## init\_queue

Inicializa uma nova fila, alocando memória para o buffer e definindo a cabeça e a cauda para zero.

## Is\_empty

Verifica se a fila está vazia, comparando as posições da head e da tail;

## Is\_full

Verifica se a fila está cheia, verificando se o próximo índice após a tail e a head;

## enequeue

Insere um novo elemento na fila, verificando se a fila está cheia antes de o fazer.

## dequeue

Remove o primeiro elemento da fila, verificando se a fila está vazia antes de o fazer.

## display\_queue

Exibe todos os elementos da fila, verificando se a fila está vazia antes de o fazer.

* Funções em Exibição

## printState

Imprime o estado atual de todos os processos.

* Funções de Utilidade

## isInsideQueue

Verifica se um processo está presente na fila.

## Init\_CPU

Inicializa o estado da CPU.

## createProcess

Cria um novo processo a partir das informações contidas num arquivo, inicializando as propriedades da estrutura ‘process’.

## readyState

Gerencia a transição de estado de pronto para em execução.

## newProcess

Verifica se existem novos processos a serem criados.

## newTimeout

Gerencia a transição de estado de novo para pronto.

## newState

Gerencia a transição de estados (novo, pronto, em execução, bloqueado).

## unBlock

Gerencia o desbloqueio de processos.

## runningTimeout

Gerencia o tempo limite do processo em execução.

## runningState

Gerencia o estado em execução.

## blockedTimeout

Gerencia o tempo limite de processos bloqueados.

## blockedState

Gerencia o estado bloqueado.

## stateHandler

Gerencia a lógica de transição de estados.

## dispatcher

Seleciona o próximo processo a ser executado.

# Conclus˜ao

Em suma, este relatório descreveu a implementação de um simulador de um modelo de 5 estados, onde os processos podem estar nos estados NEW, READY, RUN, BLOCKED ou EXIT. O trabalho foi dividido em duas partes principais. Na primeira parte, foi implementada a instrução UNBLOCK, que permite que um processo bloqueado seja desbloqueado por outro processo. Na segunda parte, foi adicionada a funcionalidade de ler o input do simulador a partir de um arquivo.

Durante o desenvolvimento, foram criadas estruturas de dados para representar os processos e o estado da CPU, além de filas para os estados READY e BLOCKED. Foram utilizadas funções para manipular essas estruturas e realizar as transições de estado dos processos.

O simulador foi testado com diferentes cenários, utilizando-se matrizes para representar as instruções dos programas. Os resultados foram satisfatórios, demonstrando o correto funcionamento do simulador.

Em conclusão, este trabalho prático proporcionou uma oportunidade para aplicar conhecimentos sobre sistemas operativos. A implementação do simulador permitiu compreender melhor os conceitos relacionados com estados dos processos e a comunicação entre eles. Além disso, a leitura de input a partir de um arquivo proporcionou uma abordagem mais flexível e escalável para a execução dos programas.

No geral, este trabalho contribuiu para o aprimoramento dos conhecimentos em sistemas operativos e programação de sistemas, fornecendo uma experiência prática e relevante para a compreensão desses conceitos.