Unidade Curricular Sistemas Operativos

Licenciatura em Engenharia Informática

2º Trabalho - Modelo de 5 Estados

Docentes: Pedro Salgueiro e Daniela Schmidt

Discentes: Gustavo Dias Gomes 48392

Bárbara Loureiro 48469

Introdução

O simulador de escalonamento de processos é um programa desenvolvido em linguagem C que simula o funcionamento de um sistema operacional simples. Ele implementa um algoritmo de escalonamento de processos básico, permitindo que os programas sejam executados em um ambiente simulado. O objetivo do simulador é ilustrar o funcionamento do escalonamento de processos e fornecer insights sobre o comportamento dos programas em diferentes estados.

Nesta 2ª parte do trabalho continuamos a implementação do modelo de 5 estados, desta vez com a implementação da instrução UNBLOCK e com a leitura do input a partir de um ficheiro.

Implementação

O código do simulador está dividido em várias seções, cada uma responsável por uma parte específica do sistema. A seguir, estão as principais estruturas e funções implementadas:

1. Estruturas:

- SO (Sistema Operacional): Armazena o estado atual do sistema, incluindo o instante de tempo, o número de programas e a fila de prontos e bloqueados.
- Queue (Fila): Define as filas de programas prontos e bloqueados, usando vetores para armazenar os programas.
- Program: Representa um programa em execução, com informações como tempo atual de execução, estado, ciclo de execução e ID do programa a ser desbloqueado.

2. Funções principais:

- isEmpty(enum Queues Q): Verifica se uma fila está vazia.
- peek(enum Queues Q): Retorna o primeiro programa de uma fila.
- enqueue(int id, enum Queues Q): Insere um programa na fila correspondente.
- dequeue(enum Queues Q): Remove o primeiro programa de uma fila.
- getState(int id): Retorna o estado atual de um programa.
- setState(int id, enum States state): Define o estado de um programa.
- unblockProcess(int id): Desbloqueia um programa, colocando-o na fila de prontos.
- changeProgram(int id): Analisa o estado de um programa e determina o seu próximo estado.
- printProgramState(enum States state): Imprime o estado atual de um programa.

3. Funções auxiliares:

- run(): Executa o simulador, avançando o tempo e atualizando o estado dos programas.

- readFile(const char *filename, int programas[ROWS][COLS]): Lê um arquivo de entrada contendo os ciclos de execução dos programas.

Funcionamento do Simulador:

O simulador inicia através da leitura de um arquivo de entrada chamado "input.txt" que contém os ciclos de execução dos programas. Os ciclos são armazenados na matriz "programas", onde cada linha representa um programa e cada coluna representa um instante de tempo. O número de programas é definido pela constante ROWS.

Em seguida, o simulador inicializa a estrutura do sistema operacional (SO), definindo o número de programas, as filas de prontos e bloqueados, e o estado inicial de cada programa. O estado inicial é definido como NONCREATE, indicando que os programas ainda não foram criados.

Após a inicialização, o simulador executa o loop principal do escalonador. Em cada iteração do loop, o simulador avança o tempo em um instante, verifica o estado de cada programa e realiza as operações correspondentes, como atualizar o estado, desbloquear programas, enqueue e dequeue de programas.

Durante a execução do escalonador, são exibidos no terminal o instante de tempo atual e o estado de cada programa em execução. Os estados possíveis são: NEW, READY, RUN, BLOCKED, READY_UNBLOCK, EXIT, FINISH e NONCREATE.

O escalonador continua a executar até que todos os programas tenham finalizado sua execução, ou seja, até que não haja programas em execução e a fila de prontos esteja vazia.

Conclusão

O simulador de escalonamento de processos implementado em C demonstra o funcionamento básico de um sistema operacional. Ele fornece uma visão simplificada do processo de escalonamento de programas e permite observar o comportamento dos programas em diferentes estados. O simulador pode ser útil para estudar e entender os princípios do escalonamento de processos em sistemas operacionais. Infelizmente não obtivemos o output no terminal esperado, mas a tentativa de implementação permaneceu.