Relatório do trabalho 2 de Sistemas Operativos

Simulador de Sistema Operativo

Acácio Uando - 55730
Luís Borges - 47297
Thawila Simbine - 49183

Évora, Junho de 2024

Introdução

O trabalho consiste na implementação de um simulador de sistema operativo incluindo um módulo de gestão de memória. Este trabalho é uma continuação do trabalho 1 que consistia na implementação de um simulador de sistema operativo centrado em escalonamento de processos usando um modelo de 4 estados (READY, EXECUTING, INTERRUPTIBLE e ZOMBIE). De acordo com a descrição do trabalho apresentada no enunciado, na implementação este deve ser dividido em duas partes:

- P**arte 1**: Implementar um simulador de memória que pode utilizar duas estratégias. de alocação diferente para paginação;
- **Parte 2:** Integrar o simulador de memória no trabalho 1, adicionando novos estados.

O grupo conseguiu implementar as duas partes do trabalho com sucesso e ambas as partes funcionam sem erros. O simulador resultante da integração do módulo de gestão de memória com o simulador do trabalho 1, executa sem erros, e usa os estados equivalentes apresentados na descrição dos enunciados e leva em conta todas as considerações apresentadas nos enunciados. O código do simulador em ambas as partes está em sua maioria comentado, e portanto caso o código em si não esteja perceptível o suficiente, os comentários podem ajudar na percepção.

Estrutura do programa

Parte 1: **Implementar um simulador de memória que pode utilizar duas estratégias**

O programa simulador desta parte está organizado em quatro diretórios como apresentado a seguir:

- ***bin*** - diretório onde fica o ficheiro executável do simulador l. Este ficheiro é gerado quando o programa é compilado. Neste diretório pode ser encontrado apenas um ficheiro de nome *so*.
- ***include*** - diretório onde ficam os ficheiros de cabeçalho do programa. Os ficheiros de cabeçalho do programa são:
    - *inputs.h* - ficheiro onde se encontram os inputs (exemplos de memória dos processos e sequência de execução dos processos) para testes do programa.
    - *list.h* - ficheiro onde está definida a estrutura de dados lista. Esta estrutura de dados é usada para armazenar a lista de processos no sistema.
    - *messages.h* - ficheiro onde estão definidas mensagens de erros usadas pelo programa.

    - *fifo_frame.h* - ficheiro onde está definido o tipo FIFOFrame, um tipo que representa um frame numa memória com política de substituição FIFO. Este tipo tem como atributo:
        - *data* - id do processo a que o frame pertence.
    - *base_memory* - ficheiro onde estão definidas constantes correspondentes ao tamanho da memória, número de frames que a memória pode comportar, o número de bytes por frame, e funções para navegação em posições de memória.
    - *fifo_memory.h* - ficheiro onde está definida a estrutura e funções para carregamento de dados em uma memória que usa como política de substituição de frames a política FIFO. Esta memória tem tamanho máximo de 20kb e cada frame tem tamanho de 2kb.
    - *memory_management.h* - ficheiro onde estão definidas funções para execução de uma lista de processos usando as duas políticas de gestão de memória. As funções aqui definidas imprimem as saídas da execução tanto no terminal como em ficheiros.
    - *system.h* - ficheiro onde está definida a função *execute_processes*, responsável pela execução de um pseudo-programa.
    - *processor.h* - ficheiro onde está definido o tipo PROCESSOR, tipo correspondente a um processo e suas funções de criação e eliminação. Este tipo possui os seguintes atributos:
        - *id* - ID do processo
        - *frames_start_at* - índice onde se encontra o primeiro frame do processo na memória. Este atributo é -1 se o processo ainda não entrou na memória.
        - *size* - tamanho do processo.
- ***obj*** - diretório onde ficam os ficheiros binários do programa. Estes ficheiros são gerados quando o programa é compilado.
- ***src*** - diretório onde ficam os ficheiros de implementação do programa. Os ficheiros desde diretório possuem os mesmos nomes dos ficheiros de cabeçalhos com distinção da extensão (.c) e acréscimo do ficheiro *main.c*, que é o ficheiro de inicialização do programa. No ficheiro *main.c* estão apresentados de forma explícita os inputs executados para obtenção das saídas. Para remover ou adicionar novos inputs para testes, basta modificar este ficheiro comentando os exemplos existentes ou adicionando os tais novos inputs.
- *README* – ficheiro com as instruções de compilação e execução do programa.
- *Makefile* - ficheiro com as regras de compilação do programa.
- *fifo0X.out* - ficheiros de output gerados pela execução do programa.

Parte 2: Integração do simulador de memória no trabalho 1

O programa simulador resultante desta parte está organizado em quatro diretórios como apresentado a seguir:

- ***bin*** - diretório onde fica o ficheiro executável do simulador l. Este ficheiro é gerado quando o programa é compilado. Neste diretório pode ser encontrado apenas um ficheiro de nome *so*.
- ***include*** - diretório onde ficam os ficheiros de cabeçalho do programa. Os ficheiros de cabeçalho do programa são:
    - *inputs.h* - ficheiro onde se encontram os inputs (exemplos de pseudo-programas e memória dos processos) para testes do programa.
    - *list.h* - ficheiro onde está definida a estrutura de dados lista. Esta estrutura de dados é usada para armazenar a lista completa de tarefas no sistema (tanto as não criadas, as criadas, assim como as removidas do sistema).
    - *queue.h* - ficheiro onde está definida a estrutura de dados fila circular (*queue* - FIFO). Esta estrutura de dados apresenta tanto o comportamento de fila com prioridade, assim como de fila sem prioridade e é usada para armazenar tarefas nos estados READY, INTERRUPTIBLE, ZOMBIE e tarefas não criadas.
    - *messages.h* - ficheiro onde estão definidas mensagens de erros usadas pelo programa.
    - *fifo_frame.h* - ficheiro onde está definido o tipo FIFOFrame, um tipo que representa um frame numa memória com política de substituição FIFO. Este tipo tem como atributo:
        - *data* - id do processo a que o frame pertence.
    - *base_memory* - ficheiro onde estão definidas constantes correspondes ao tamanho da memória, número de frames que a memória pode comportar, o número de bytes por frame, e funções para navegação em posições dsememória.
    - *fifo_memory.h* - ficheiro onde está definida a estrutura e funções para carregamento de dados em uma memória que usa como política de substituição de frames a política FIFO. Está memória tem tamanho maximo de 20kb e cada frame tem tamanho de 2kb.
    - *system.h* - ficheiro onde está definida a função *run*, responsável pela execução de um pseudo-programa.
    - *sheduling.h* - ficheiro onde está definido o núcleo do programa. Neste ficheiro está definida a estrutura *Processor* que simula um processador e a função *execute_tasks*, responsável pelo execução das tarefas. A composição da estrutura Processor é a seguinte:
        - *timer* - relógio do sistema
        - *remaing_waiting_time* - tempo de espera restante. Este atributo é usado para definir tempos de espera do processador depois da

mudança de tarefas, podendo ser 3 instantes para casos em que se efetue mudança de uma tarefa para outra, mas que sejam de processos diferentes e 1 instante para casos em que a tarefas sejam do mesmo processo.

- *processor.h* - ficheiro onde está definido o tipo PROCESSOR, tipo correspondente a um processo e suas funções de criação e eliminação. Este tipo possui os seguintes atributos:
  - *id* - ID do processo
  - *frames_start_at* - índice onde se encontra o primeiro frame do processo na memória. Este atributo é -1 se o processo ainda não entrou na memória.
  - *size* - tamanho do processo.
- *task.h* - ficheiro onde está definido o tipo TASK, tipo correspondente a uma tarefa e funções de criação e eliminação. Este tipo possui os seguintes atributos:
  - *id* - ID da tarefa
  - *process* - referência do processo a que a tarefa pertence
  - *instruction_pointer* - apontador do índice da instrução atual. Como as instruções do programa (execução ou bloqueio) são definidos num array, então este atributo referencia o indice da instrução atualmente em execução ou o instante de chegada da tarefa (índice 0).
  - *instructions* - instruções do programa (instante de chegada, tempo de execução, tempo de bloqueio)
  - remaining_time_to_leave_the_system - tempo restante para a tarefa ser removida do sistema. Esta variável somente é usada quando a tarefa entra no estado ZOMBIE.
  - status - estado atual da tarefa

- ***obj*** - diretório onde ficam os ficheiros binários do programa. Estes ficheiros são gerados quando o programa é compilado.
- ***src*** - diretório onde ficam os ficheiros de implementação do programa. Os ficheiros desde diretório possuem os mesmos nomes dos ficheiros de cabeçalhos com distinção da extensão (.c) e acréscimo do ficheiro *main.c*, que é o ficheiro de inicialização do programa. No ficheiro *main.c* estão apresentados de forma explícita os inputs executados para obtenção das saídas. Para remover ou adicionar novos inputs para testes, basta modificar este ficheiro comentando os exemplos existentes ou adicionando os tais novos inputs.
- *README* – ficheiro com as instruções de compilação e execução do programa.
- *Makefile* - ficheiro com as regras de compilação do programa.
- *simulador0X.out* - ficheiros de output gerados pela execução do programa.

Notas:

- As regiões realçadas com fundo azul representam acréscimos feitos ao trabalho 1 para se alcançar o comportamento exigido no trabalho 2.
- As regiões realçadas com fundo verde representam ficheiros (.h e .c) que necessitam de modificações no trabalho 1 para se alcançar o comportamento exigido no trabalho 2.

Funcionamento do programa

Parte 1: A parte 1 do trabalho funciona segundo a descrição apresentada no enunciado. Quando o programa é executado, este carrega os processos em uma lista, e depois os insere em cada uma das memórias implementadas seguindo a ordem apresentada nos inputs. Para cada um dos exemplos apresentados no enunciado, o programa gera ficheiros com as saídas denominadas por fifoXX.out, segundo o tipo de memória usado na execução dos processos.

Parte 2: O simulador apresenta o mesmo comportamento com o do trabalho 1, com exceção que agora aplica novos estados (SWRDY e SWINT) e faz gestão de memória também.

Instruções de compilação e execução do programa

- Para compilar o programa, é necessário executar o comando "make" na linha de comando.
- Após a compilação do programa simulador, um executável é criado e armazenado do diretório "bin" com o nome "so". Então
- para executa-lo, basta executar o ficheiro de nome *so* como no exemplo a seguir:

  ```
  $ ./bin/so
  ```

- Quando o simulador entra em execução, este executa todos os pseudo-programas (inputs) apresentados no ficheiro *input.c* pelo professor, e escreve a saída na linha de comando e nos ficheiros de output (*simulador0X.out*, sendo X o número do input executado).

Anexos ao código fonte

Junto do código fonte do programa estão os ficheiros de output para cada input fornecido pelo professor nos ficheiros *inputs_part1.c* e *inputs_part2.c*