Relatório do trabalho 1 de Sistemas Operativos

Simulador de Sistema Operativo

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

Évora, Abril de 2024

Introdução

O trabalho consiste na implementação de um simulador de Sistema Operativo considerando um modelo de 4 estados baseados numa simplificação do Linux para tarefas (threads). A linguagem de programação a usar na implementação do programa é a linguagem C. O sistema neste trabalho, para efeitos de escalonamento/scheduling e estados considera tarefas como processos, estamos, portanto, perante Kernel-Level threads. As tarefas têm uma relação M:1 com os processos, ou seja um processo pode ter várias tarefas. O modelo deve incluir os seguintes estados (por conveniência entre parênteses são apresentados estados equivalentes nos modelos estudados): READY, EXECUTING (RUNNING), INTERRUPTIBLE (BLOCKED) e ZOMBIE (EXIT). Quando os processos estão em READY, e INTERRUPTIBLE, estão em filas de espera do tipo FIFO (first in first out).

O programa simulador por nós implementado executa sem erros, e usa os estados equivalentes apresentados na descrição acima e leva em conta todas as considerações apresentadas no enunciado. O código do programa está em sua maioria comentado, portanto caso não esteja perceptível o suficiente, basta olhar para os comentários.

Estrutura do programa

O programa simulador por nós criado está organizado em quatro diretórios como apresentado a seguir:

- ***bin*** - diretório onde fica o ficheiro executável. 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) 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 os não criados, os criados, assim como os removidos).
    - *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.
    - *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.
    - *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_id - id do processo em que a tarefa pertence
        - instruction_pointer - apontador do índice da instrução atual. Como as intruções 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 retante 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.
- *output0X.out* - ficheiros de output gerados pela execução do programa.

Funcionamento do programa

O programa funciona segundo a descrição apresentada no enunciado. Quando o programa é executado, este carrega os pseudo-programas para as estruturas de dados do sistema, onde cada pseudo-programa é interpretado como uma tarefa com os atributos descritos na seção anterior. Depois do carregamento dos pseudo-programas (como tarefas) para as estruturas de dados, o programa inicia a execução das tarefas resultantes e escreve as saídas tanto na linha de comando, assim como num ficheiro de nome *output0X.out*, onde *X* é o número do input usado.

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 (*output0X.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 no ficheiro *input.c*