Projeto da Disciplina de Sistemas Operacionais (5954016)

Prof. Dr. Cléver Ricardo Guareis de Farias

1. Objetivo

Implementar em Java um escalonador de processos que simule a execução virtuais de acordo com uma política de escalonamento que favoreça processos intensivos de processamento, mas que ao mesmo tempo em não cause estagnação em processos que façam muita entrada e saída.

2. Descrição

O simulador será composto por três threads de execução. O thread UserInterface implementa a interface (gráfica) com o usuário do sistema. Através deste thread o usuário deverá prover os comandos de controle de simulação. O thread LongTermScheduler representa o escalonador de longo prazo do sistema. Este thread é responsável por receber as submissões de processos feitas ao sistema e controlar a carga total no sistema. O thread ShortTermScheduler representa o escalonador de curto prazo do sistema. Este thread, que deverá ser construído de acordo com a política de escalonamento definida anteriormente e será responsável pela execução dos processos submetidos ao simulador. Cada processo deverá ser descrito na linguagem de definição de programas simulados (veja seção 4). A carga máxima de processos que o escalonador de curto prazo pode suportar deve ser informada via linha de comando/parâmetro de execução quando da inicialização do simulador. A interação entre os threads do sistema é realizada por meio de um conjunto de interfaces. A Figura 1 ilustra a arquitetura do simulador de escalonamento.

O simulador deve ser inicializado com o valor em milissegundos de um quantum de tempo, e.g., 200 ms equivalem a uma fatia de tempo. O escalonador de longo prazo deve manter uma fila dos processos criados, mas ainda não admitidos no sistema. O escalonador de curto prazo deve manter diferentes filas para armazenar os processos prontos, bloqueados (realizando entrada e saída) e terminados.

3. Descrição das Interfaces

Quatro interfaces foram definidas para implementar as interações entre os threads do sistema (veja anexo A para a representação em Java). A interface SubmissionInterface representa as interações entre os threads UserInterface e LongTermScheduler. As seguintes operações são definidas nesta interface:

- submitJob, utilizada para submeter um novo programa (*job*) para o simulador. Esta operação tem como parâmetro o identificador do arquivo contendo o programa a ser executado. Ao receber a solicitação de submissão de um programa, o simulador deve criar um processo para representar o programa internamente. O processo criado deve ser armazenado na fila de processos a serem admitidos no sistema. Estes processos serão então encaminhados pelo escalonador de longo prazo ao escalonador de curto prazo de acordo com a carga de processos no mesmo. Um programa é descrito na linguagem definida na seção 4. Caso o identificador do arquivo fornecido seja inválido ou houver algum erro sintático com o programa descrito, esta operação deverá retornar um valor falso, caso contrário a operação irá retornar o valor verdadeiro.

- displaySubmissionQueue, utilizada para solicitar a descrição da fila de processos submetidos ao escalonador de longo prazo, mas ainda não encaminhados ao escalonador de curto prazo.

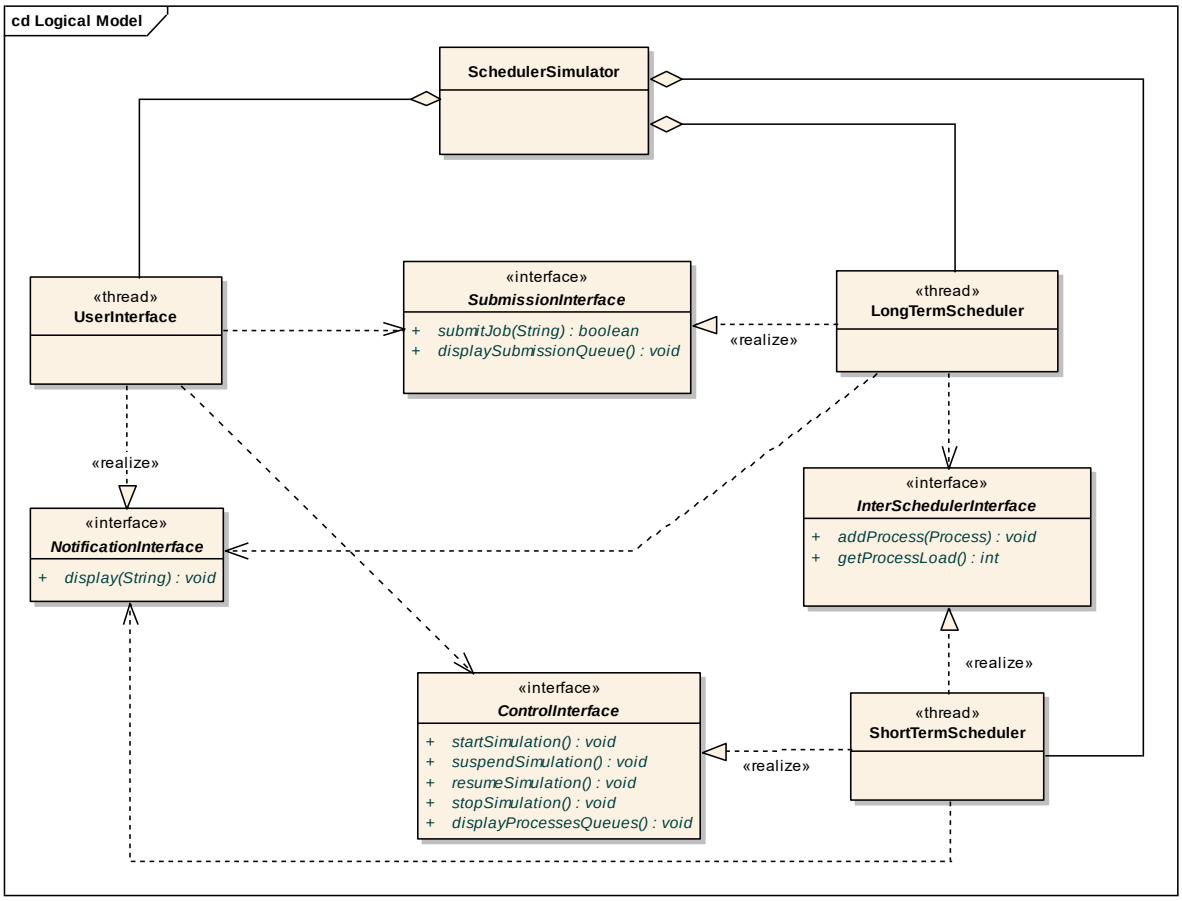

Figura 1. Arquitetura do simulador de escalonamento.

A interface InterSchedulerInterface representa as interações entre os threads LongTermScheduler e ShortTermScheduler. As seguintes operações são definidas:

- addProcess, utilizada para submeter um processo à execução. Esta operação tem como parâmetro um objeto do tipo *Process* (a ser definido);
- getProcessLoad, utilizada para obter a carga atual de processos no escalonador de curto prazo.

A interface ControlInterface representa as interações entre os threads UserInterface e ShortTermScheduler. As seguintes operações são definidas:

- startSimulation, utilizada para iniciar a simulação. A partir da invocação desta operação, o simulador deve escolher um processo que esteja pronto para execução. O simulador deve processar uma instrução por vez. Duas diferentes instruções são providas pela linguagem de definição de programas simulados: execute, a qual representa um *burst* de CPU (execução de um conjunto de instruções), e block, a qual representa um burst de CPU seguido de uma instrução de E/S (bloqueio). A execução de uma instrução execute deve ser realizada por meio de uma espera cronometrada. Ao final de um quantum, o simulador deve proceder de acordo

com a política de escalonamento implementada. A execução de uma instrução block deve ser realizada por meio de uma espera cronometrada, seguida do bloqueio do processo em execução por um determinado período (medido em quantum). Ao final de cada quantum, o escalonador deverá verificar se algum dos processos bloqueados tornou-se disponível, tornando-o pronto para execução. A simulação termina quando não houver mais processos prontos ou bloqueados para execução.

- suspendSimulation, utilizada para interromper temporariamente a simulação. As informações sobre os processos em execução, prontos, bloqueados e terminados não devem ser alteradas.
- resumeSimulation, utilizada para continuar a simulação a partir do ponto em que ela foi interrompida temporariamente.
- stopSimulation, utilizada para encerrar definitivamente uma simulação.
- displayProcessQueues, utilizada para solicitar a descrição das informações sobre todos os processos no escalonador de curto prazo (processo em execução, processos prontos, processos bloqueados e processos terminados).

A interface NotificationInterface representa as interações entre o thread UserInterface e os threads LongTermScheduler e ShortTermScheduler. A operação abaixo é definida:

- display, utilizada pelos escalonadores de curto e longo prazo para notificar qualquer informação de interesse relacionada ao acompanhamento da simulação para o thread UserInterface.

4. Linguagem para a definição dos programas simulados

A Figura 2 apresenta a BNF que descreve a linguagem para a definição de um programa simulado. Todo programa contém um cabeçalho contendo a palavra-chave "program" seguido do identificador do programa (identificador deve ter o mesmo nome do arquivo contendo o programa). Finalmente, <EndOfLine> representa o(s) caracter(es) de fim de linha e o símbolo + representa que o elemento pode ser repetido (elemento aparece uma ou mais vezes).

```
<program> ::= <program_statement> <program_body>

<program_statement> ::= "program" <whitespace> <file_name> <EndOfLine>

<program_body> ::= <begin_statement> <program_behaviour> <end_statement>
<begin_statement> ::= "begin" <EndOfLine>
<end_statement> ::= "end" <EndOfLine>

<program_behaviour> ::= (<behaviour_statement>)+
<behaviour_statement> ::= "execute" <EndOfLine> | "block" <whitespace> <block_period> <EndOfLine>

<whitespace> ::= " "
<block_period> ::= "1" | "2" | "3" | "4" | "5"
```

Figura 2. BNF da linguagem de representação de programas simulados

A Figura 3 apresenta um exemplo de um programa definido utilizando a linguagem de definição de programas simulados.

```
program teste.txt
begin
execute
block 4
execute
execute
execute
execute
execute
execute
execute
execute
execute
block 2
execute
end
```

Figura 3. Exemplo que programa definido na linguagem para a definição de programas simulados.

5. Observações finais

Poderão ser constituídos grupos de no máximo três integrantes. A composição dos grupos deverá ser informada via lista distribuída na sala de aula ou via e-mail (farias@ffclrp.usp.br) até dia 17/05.

Eventuais dúvidas deverão ser encaminhadas apenas presencialmente (durante as aulas ou na sala do professor). Não serão respondidas dúvidas encaminhadas por e-mail. Caso seja feito algum esclarecimento que resulte na alteração deste documento de projeto, uma nova versão deste documento será produzida e disponibilizada no próprio TIDIA-Ae. Neste caso, um novo aviso será adicionado à plataforma.

O projeto deverá ser entregue exclusivamente via TIDIA-Ae até o dia 22 de junho (veja informações sobre o prazo de entrega da atividade na própria ferramenta). A entrega consistirá na implementação da solução, incluindo os arquivos fontes e um arquivo executável (formato jar), e documentação da implementação. Junto com estes arquivos deverá ser entregue um documento (formato PDF) descrevendo/justificando como a solução implementada satisfaz a política de escalonamento especificada no projeto.

Se necessário, o grupo deverá incluir um arquivo contendo as instruções de uso do programa. Apenas um membro do grupo precisa realizar a entrega (upload) da implementação. Contudo, a entrega deverá incluir também um arquivo contendo os nomes dos membros dos grupos que efetivamente contribuíram para a implementação, bem como uma descrição sucinta das classes/métodos que cada membro implementou. Não será aceita a entrega do projeto sem estas informações.

A apresentação do projeto deverá ser agendada com o professor após a entrega do projeto. Todas as apresentações deverão ser feitas até o dia 02 de julho. Todos os membros do grupo deverão estar presentes durante a apresentação.

Além dos aspectos funcionais do projeto, serão também avaliadas a estrutura, a documentação e a usabilidade da aplicação desenvolvida.

Anexo A

1. Interface SubmissionInterface

```
public interface SubmissionInterface {
    boolean submitJob(String fileName);
    void displaySubmissionQueue();
}
```

2. Interface InterSchedulerInterface

```
public interface InterSchedulerInterface {
    void addProcess(Process bcp);
    int getProcessLoad();
}
```

3. Interface ControlInterface

```
public interface ControlInterface {
    public void startSimulation();
    public void suspendSimulation();
    public void resumeSimulation();
    public void stopSimulation();
    public void displayProcessQueues();
}
```

4. Interface NotificationInterface

```
public interface NotificationInterface {
    void display(String info);
}
```