MAC-212 — Laboratório de Computação

Exercício Programa 2: Simulação a Eventos Discretos

Prazo: 29 de junho

1 Introdução

Nosso objetivo é escrever um programa para simular o comportamento de um sistema de computação que executa tarefas (*jobs*) cuja duração é conhecida de antemão. Para cada tarefa, são dados:

- o tempo de chegada da tarefa
- a duração da tarefa.

O sistema simulado executa uma tarefa de cada vez. Uma tarefa tipicamente não começa a ser executada no instante t_0 de sua chegada ao sistema (pois este pode estar ocupado executando outra tarefa), mas em algum instante $t_1 \geq t_0$. A duração de uma tarefa é o tempo que o sistema efetivamente leva para executar essa tarefa. Ela não inclui o tempo que a tarefa fica aguardando sua execução.

Seu programa simulará o comportamento do sistema de computação para três algoritmos de escalonamento de tarefas:

- FIFO (*first in, first out*)

  Enquanto o sistema estiver ocupado executando uma tarefa, novas tarefas serão colocadas em uma fila de espera. Essa fila segue a disciplina FIFO, ou seja as tarefas são enfileiradas em ordem de chegada. Quando acabar a execução da tarefa atual, o sistema iniciará a execução da primeira tarefa da fila.

- SJF (*shortest job first*)

  Aqui a única mudança em relação ao caso anterior é a disciplina empregada na fila de espera. Agora a fila segue a disciplina *shortest job first*, ou seja, quando acabar a execução da tarefa atual, o sistema iniciará a execução da tarefa mais curta da fila.

- FIFO com preempção

  Como no caso FIFO, porém a tarefa atual pode permanecer no estado "em execução" por no máximo T unidades de tempo. Transcorrido esse intervalo (que é a "fatia de tempo" alocada a cada tarefa), a tarefa atual vai para o final da fila de espera e o sistema passa a executar a primeira tarefa dessa fila. Essa mudança no estado da tarefa que foi para o final da fila — de "em execução" para "em espera" — é denominada preempção.

2 Simulação a Eventos Discretos

Em vez de fazer a simulação em tempo real, seu programa usará o conceito de "tempo simulado" ou "tempo da simulação". O que caracteriza a simulação a eventos discretos é o fato do tempo da simulação ser descontínuo. Ele anda aos saltos: suponha que um evento e_a ocorreu no instante t_a do tempo simulado e foi sucedido pelo evento e_b, o qual ocorreu no instante t_b do tempo simulado. Se não aconteceu nenhum evento de interesse entre e_a e e_b, então o tempo simulado pulou de t_a diretamente para t_b.

2.1 A Fila de Eventos

Cada evento está associado a um tempo simulado. Os eventos que ainda não foram tratados pelo simulador (ou, em outras palavras, que "ainda não aconteceram" no tempo simulado) são mantidos numa fila com prioridade (*priority queue*) denominada fila de eventos. A prioridade de um evento é o tempo simulado associado a ele.

Antes da simulação começar, todos os eventos iniciais (aqueles que não são gerados pela própria simulação) são colocados na fila de eventos. Um exemplo de evento inicial: "chegada de tarefa com duração 500, no instante $t = 35$ do tempo simulado". O evento é a chegada de tarefa com duração 500. Ele está associado ao tempo simulado $t = 35$. Nessa fase de iniciação da simulação o tempo simulado ainda não começou a correr.

2.2 O Laço de Eventos

O tempo simulado começa a correr quando o simulador entra no seu laço principal, também conhecido como laço de eventos (*event loop*). Enquanto existirem elementos na fila de eventos, esse laço faz o seguinte:

1. Retira o ítem mais prioritário da fila de eventos. O ítem mais prioritário é o evento associado ao menor tempo simulado. Esse evento é o evento atual.

2. Atribui à variável `tempoAtual` o valor do tempo simulado associado ao evento atual.

3. Trata o evento atual. Esse tratamento pode adicionar novos eventos à fila de eventos.

Note que o tratamento de um evento depende do tipo desse evento. Exemplos de tipos de eventos: "chegada de tarefa com duração X", "início da execução de tarefa com duração X", e "fim da execução de tarefa". Note também que só o primeiro desses tipos de eventos corresponde a eventos iniciais. Os outros dois tipos são gerados durante a simulação. Exemplo: Nos esquemas FIFO e SJF, o tratamento do primeiro evento do tipo "chegada de tarefa com duração X" gera um evento "fim da execução de tarefa" associado ao tempo simulado `tempoAtual + X`. (Isso não é necessariamente verdade no esquema FIFO com preempção, pois a duração da tarefa pode ser maior que a fatia de tempo.)

2.3 Dados de Entrada da Simulação

Esses dados determinam o conjunto de eventos iniciais (uma lista de tempos de chegada e durações de tarefas) ou, opcionalmente, permitem a geração aleatória desse conjunto (tempo médio entre chegadas de tarefas e duração média de uma tarefa).

2.4 Resultados da Simulação

A simulação acaba quando a fila de eventos esvaziar. Ao final da simulação, devem ser impressos os seguintes dados:

- duração em tempo simulado (último valor do tempo simulado menos primeiro valor do tempo simulado)
- número de tarefas tratadas
- duração média de uma tarefa
- tempo de espera médio de uma tarefa
- tempo total médio de uma tarefa
- comprimento médio da fila de espera

3 Orientação a Objetos

Seu simulador deve ser usar as técnicas de orientação a objetos.

3.1 A Classe `FilaComPrioridade`

Você deverá ter uma classe `FilaComPrioridade`, que servirá tanto para a fila de eventos como para a fila de espera do caso SJF. Lembre-se que uma fila com prioridade é uma estrutura de dados com as operações `adiciona` e `removeMinimo`.

3.2 A Classe `Evento` e suas Subclasses

Os diferentes tipos de eventos devem ser implementados por classes derivadas (subclasses) de uma classe abstrata `Evento`. Além de possuir um campo com o tempo simulado associado ao evento, essa classe deve ter um método abstrato `trata()`, o qual deve ser definido pelas subclasses concretas de `Evento`. Cada uma dessas subclasses (`ChegadaDeTarefa`, por exemplo) deve definir um método `trata()` com o tratamento específico para esse tipo de evento. Uma subclasse de evento pode ter ainda campos adicionais, com informações específicas para esse tipo de evento. (Exemplo: o campo `duracaoDaTarefa`, na classe `ChegadaDeTarefa`.)

Note, ainda, que o tratamento de um evento depende também do algoritmo de escalonamento de tarefas (FIFO, SJF, ou FIFO com preempção) do sistema simulado. Assim, um mesmo tipo de evento ("final de execução de tarefa", por exemplo) poderá corresponder a várias classes, uma para cada algoritmo de escalonamento (`FinalDeTarefaFIFO`, `FinalDeTarefaSJF`, ...), pois o método `trata()` varia conforme do algoritmo de escalonamento.

3.3 A Classe `Simulacao`

Essa classe contém a fila de eventos, o campo `tempoAtual` e o laço de eventos. Ela possui os métodos `adicionaEventoInicial(Evento e)` e `executa()`. O segundo desses métodos roda o laço de eventos.

Bom Trabalho!