Programação Orientada a Objetos

Gonzalo Travieso

2021

Execução de *jobs* em sistemas de multiprogramação

Alguns sistemas computacionais são dedicados à execução de tarefas submetidas por usuários, denominadas *jobs*, ao invés da execução interativa dos comandos fornecidos pelo usuário. Nessas máquinas, o usuário configura um *job* com o programa a executar e os dados necessários, e **submete** esse *job* para execução pelo computador. Quando a execução termina, o usuário é notificado e os resultados pode ser encontrado em um local conhecido pelo usuário. A idéia é que esse sistema computacional atende a múltiplos usuários simultaneamente, e cada usuário pode submeter múltiplos *jobs*. Desta forma, múltiplos *jobs* podem estar simultaneamente pendentes para execução. Para possibilitar executar rapidamente os *jobs*, esses sistemas dispõe normalmente de diversos processadores.

Quando um usuário submete um *job*, este é enviado para um sub-sistema de filas. Quando um processador termina a execução de um *job*, ele busca no sistema de filas o próximo *job* a executar, caso algum esteja esperando. O sub-sistema de fila é essencial, pois ele define quais entre os *jobs* prontos para execução serão enviados para processamento, e portanto pode influenciar o tempo de espera desses *jobs*.

Características do sistema a simular

Você vai implementar um simulador baseado em eventos discretos desse tipo de sistema com as seguinte características:

- Existem p processadores para executar os *jobs*. Todos os processadores são idênticos.
- Os *jobs* a executar são de dois tipos: normais ou de alta prioridade. Todos são caracterizados pelo seu tempo total de execução que é um valor aleatório com distribuição normal de média τ e desvio padrão σ (valores independentes da prioridade).
- A geração de novos *jobs* ocorre entre os instantes 0 e T.
- São gerados m *jobs* do tipo normal em instantes aleatórios entre 0 e T.
- São gerados $\lfloor m/\alpha \rfloor$ *jobs* de alta prioridade em instantes aleatórios entre 0 e T.
- Quando um processador fica disponível, o sistema de filas sempre despacha um *job* de alta prioridade, se algum estiver esperando, e apenas despacha um de prioridade normal se não houver de alta prioridade aguardando. *Jobs* de mesma prioridade são executados na ordem da submissão: os que forem submetidos antes são enviados para execução primeiro.

Desta forma, a simulação tem os seguintes seis parâmetros: p, τ, σ, T, m e α.

Trabalho

Você deve implementar um simulador para um sistema com as características descritas acima. Para a simulação de eventos discretos, use o módulo `desimul.py` apresentado em aula.

O seu código deve ler os parâmetors da simulação da linha de comando e executar a simulação até que não existam mais eventos a serem processados. Como resultado, o programa deve gerar e escrever os seguintes dados:

- Um arquivo `jobs.dat` com uma linha para cada um dos *jobs* executados, e nessa linha as seguintes informações (em ordem), separadas por espaço em branco: instante em que o *job* foi submetido, se é alta prioridade ou não (0 para normal, 1 para alta prioridade), tempo total de execução, instante em que foi posto para execução.

- Um arquivo `processors.dat` com uma linha para cada processador. Na linha do processador devem estar, separadas por espaço em branco, na ordem: número total de tarefas executadas, quantas foram de alta prioridade e tempo total desocupado (não esteve executando nenhuma tarefa).

Além de gerar esse dois arquivos, o programa deve, antes de terminar, imprimir uma mensagem dizendo o tempo total de simulação (quer dizer, o instante em que a última tarefa terminou de executar), considerando que a execução começa em $t = 0$.

Algumas observações:

- O sistema a ser simulado aqui é em boa parte similar ao sistema de caixas de banco usado no exemplo de aula. Você pode construir seu simulador do zero usando os conhecimentos adquiridos (mas usando o módulo `desimul.py`) ou pode alterar o código do exemplo. Se optar por esta segunda opção, lembre-se de que, apesar da similaridade, os sistemas não são os mesmos, e portanto o código precisa ser adequadamente alterado. Por exemplo, dois pontos que exigem alterações (além dos pontos onde há diferença no modelo) são, primeiro que este sistema é menos flexível (mais simples) do que o de caixas, e portanto toda a complexidade não necessária deve ser eliminada; segundo, que elementos similares em código são diferentes em termos conceituais do sistema simulado, portanto os nomes devem ser apropriadamente ajustados.

- Você pode usar para a implementação, além do módulo de simulação baseada em eventos discretos da aula, qualquer característica da linguagem que desejar, bem como módulos pré-definidos no Python, mas não outros módulos da comunidade. Veja a lista de módulos do Python.

- Entregue o arquivo com o código Python do seu simulador, com qualquer informação adicional necessária em comentários no começo do arquivo (além dos normais comentários de código).