# Simulação de um Sistema Operacional Multitarefa de Tempo Compartilhado

Versão 0.1

Autor
***Prof. Dr. Marco Aurélio Wehrmeister***

**Objetivos**

1. Aprofundar o estudo do gerenciamento de tarefas através de implementação de escalonadores, a execução de múltiplas tarefas e do escalador, e a apresentação gráfica dessa execução do sistema ao longo do tempo.
2. Aprimorar o conhecimento sobre interação entre tarefas usando serviços de exclusão mútua comumente encontrados em sistemas operacionais multi-tarefa;
3. Compreender e resolver os problemas de escalonamento, p.ex., inanição e inversão de prioridades;

**Requisitos Gerais**

**Todos os requisitos abaixo devem ser obrigatoriamente obedecidos em ambos os projetos:**

1. O projeto deve implementar um software que simula a execução de um sistema operacional multitarefa preemptivo de tempo compartilhado;
2. O software deve permitir a visualização gráfica da execução das tarefas dentro do sistema operacional ao longo do tempo. Todos os eventos importantes/relevantes ao longo da execução das tarefas devem aparecer na visualização.
3. O software deve ser configurável e permitir que o usuário controle a simulação do sistema, bem como as características do sistema que está sendo simulado.
4. A linguagem a ser utilizada para implementação do projeto é de livre escolha, no entanto, é necessário que o software criado possa ser executado sem a necessidade de instalar nenhuma biblioteca extra, ou qualquer outro software/runtime, ou equivalente.
5. Todo o código fonte deve ser comentado de forma que qualquer outra pessoa consiga compreender o que cada parte/função/componente de código faz e as razões/justificativas de todas as decisões/escolhas de implementação;

**PROJETO A – Implementação Inicial do Simulador**

**Requisitos:**

1. O projeto deve implementar um software que simula a execução de um sistema operacional multitarefa preemptivo de tempo compartilhado.
    1.1. O sistema simulado deve incluir um relógio global do sistema, sendo que na passagem do tempo deve ser simulada na forma de *ticks* de relógio.
    1.2. No projeto A, o sistema possui apenas uma única CPU/core para executar as tarefas;
    1.3. As informações de cada tarefa (antes, durante, depois da simulação) devem ser armazenadas durante a simulação em uma única estrutura de dados d, p.ex., *Task Control Block* (TCB);
    1.4. A simulação da execução das tarefas deve seguir as características de tempo informadas para cada tarefa, p.ex., instante de ingresso, duração, instante da ocorrência de eventos, etc.
    1.5. Dois modos de execução da simulação devem ser implementados: (a) execução passo-a-passo; (b) execução completa sem intervenção humana.
        1.5.1. No caso execução passo-a-passo (opção a), deve ser possível examinar o estado atual e as informações correntes do sistema e também de cada tarefa individualmente. A ideia é que esse modo de execução forneça um *debugger* do sistema.
        1.5.2. No caso execução completa (opção b), o software mostra apenas o resultado final da simulação do sistema. Não é necessário mostrar os passos intermediários da simulação.

2. O software deve permitir a visualização gráfica da execução das tarefas dentro do sistema operacional ao longo do tempo.
    2.1. Apresentar um gráfico de gantt que mostra as tarefas sendo executadas ao longo do tempo. Quando uma tarefa está executando, tal informação deve ser indicada com uma cor (pode ser um parâmetro de entrada) e cada tarefa deve ter uma cor diferente. Quando uma tarefa existe no sistema e ela não está sendo executada por

qualquer motivo, tal informação deve ser indicada como a ausência da cor. A visualização deve ser similar as figuras da seção 6.4 do livro do Prof. Maziero.

2.2. A atualização do grafico de gantt deve ser compatível com o modo de execução da simulação (conforme requisito 1.4);

2.3. Ao final da simulação, o software deve gerar um arquivo do tipo imagem (p.ex. JPG, PNG, SVG, etc) contendo o gráfico de gantt correspondendo a situação final do sistema após a execução de todas as tarefas;

3. O software deve ser configurável e permitir que o usuário controle a simulação do sistema, bem como as características do sistema que está sendo simulado.

3.1. O software deve permitir a configuração/parametrização do sistema a ser simulado antes de execução da simulação, incluindo, mas não limitado a: conjunto de tarefas, características individuais de cada tarefa, algoritmo de escalonamento, eventos como solicitação/liberação de um mutex, operação de entrada/saída, envio/recebimento de dados, entre outros.

3.2. O software deve sugerir valores padrão para todo os parâmetros de configuração da simulação ou do software em si. Esses valores padrão devem ser sobrescritos por valores indicados pelo usuário;

3.3. A configuração/parametrização do sistema a ser simulado deve ser carregada a partir de um arquivo texto, formato texto simples (plain text), e deve seguir o seguinte formato:

```
algoritmo_escalonamento;quantum
id;cor;ingresso;duracao;prioridade;lista_eventos
```

onde:

`algoritmo_escalonamento` indica o algoritmo de escalonamento a ser usado na simulação;

`quantum` indica o período máximo de tempo que uma tarefa pode executar;

`id` é o identificador único da tarefa pode executar;

`cor` é cor que identifica a execução da tarefa;

`ingresso` indica o instante de tempo que a tarefa foi criada;

`duracao` indica o tempo de execução da tarefa;

`prioridade` indica a prioridade da tarefa;

`lista_eventos` indica uma lista de eventos que ocorre durante a execução da tarefa.

3.3.1. A primeira linha do arquivo de configuração indica parâmetros gerais da simulação ou do sistema operacional. A partir da segunda linha, cada linha indica os parâmetros de uma tarefa do sistema. O arquivo de configuração deve ter pelo menos duas linhas (primeira linha, parâmetros do sistema, segunda linha parâmetros de uma tarefa), sendo que não há limite para o número de tarefas a serem executadas no sistema.

3.3.2. A lista de eventos de cada tarefa vai ser tratada no projeto B. No entanto, deve-se atentar para o fato da possibilidade de ocorrência de inúmeros eventos ao longo da execução de uma tarefa.

4. Os seguintes algoritmos de escalonamento devem ser implementados: FIFO, SRTF, Prioridade preemptivo.
   4.1. O usuário deve indicar, antes da simulação, qual é o algoritmo de escalonamento a ser usado;
   4.2. O mecanismo de implementação do algoritmo de escalonamento deve ser flexível/configurável/parametrizável de modo que novos algoritmos possam ser facilmente incluídos no simulador sem a necessidade de modificar o código da simulação. Idealmente, o escalonador é apenas uma função que retorna qual é a próxima tarefa a ser executada, que pode, inclusive, estar contida em uma biblioteca dinâmica fora do arquivo binário do executável do simulador;

5. A interface de interação do usuário o simulador deve ser intuitiva, fácil de entender e usar.

**PROJETO B – Refinamento da Simulação do Sistema Operacional Multitarefa**

Enunciado está sendo desenvolvido e será publicado oportunamente.