

Simulação de Algoritmos de Escalonamento

O presente trabalho tem por objetivo escrever um programa para simular o escalonamento e abstração de um conjunto de processos/threads através do uso de algoritmos de escalonamento de processos conhecidos na literatura. O programa deve suportar os seguintes algoritmos de escalonamento de processos:

- FCFS (First Come, First Served)
- Shortest Job First
- Por prioridade, sem preempção
- Por prioridade, com preempção por prioridade
- Round-Robin com quantum = 2s, sem prioridade

O programa deve ler os dados dos processos através de um arquivo de entrada, onde cada linha do arquivo corresponde a um processo, com as seguintes informações inteiras separadas por um espaço em branco:

- data da criação do processo
- duração em segundos
- prioridade estática (escala de prioridades positiva, ex: prioridade 3 > 2).

Um exemplo de arquivo de entrada para o programa seria (note que não tem linha em branco ao final do arquivo):

```
0 5 2
0 2 3
1 4 1
3 3 4
```

Neste exemplo, o processo P1 tem data de criação 0, sua execução dura 5 segundos e sua prioridade é definida como 2. Esse formato deve ser respeitado e não deve ser modificado em nenhuma hipótese (a avaliação seguirá esse formato e caso haja erro de leitura o programa não executará e consequentemente terá sua nota zerada). Note que a ordem dos processos não precisa estar ordenada por data de criação de cada processo.

Foi disponibilizado junto com a descrição do trabalho um código em C++ para realizar a leitura do arquivo de entrada.

Para cada algoritmo, o programa de simulação de escalonamento de processos deverá produzir as

seguintes informações em sua saída padrão (stdout):

- *Turnaround time* (tempo transcorrido desde o momento em que o processo é criado até o instante em que termina sua execução). Imprimir o turnaround time para cada processo e também o valor médio.
- Tempo média de espera (soma dos períodos em que o processo estava no seu estado pronto).
- Número total de trocas de contexto (considerar que na primeira execução do primeiro processo há uma troca de contexto).
- Diagrama de tempo de execução.

Para simplificar, o diagrama de tempo de cada execução pode ser gerado na vertical, de cima para baixo (uma linha por segundo), conforme mostra o exemplo a seguir:

tempo	P1	P2	P3	P4
0- 1	##	--		
1- 2	##	--	--	
2- 3	--	##	--	
3- 4	--	##	--	--
4- 5	--		##	--
5- 6	--		##	--
6- 7	##		--	--
7- 8	##		--	--
8- 9	--		--	##
9-10	--		--	##
10-11	--		##	--
11-12	--		##	--
12-13	##			--
13-14				##

Sendo que “##” significa que o processo em questão está em execução, “--” significa que o processo já foi criado e está esperando, “ ” (espaço em branco) o processo ainda não iniciou ou já acabou, ou seja, não está na fila de prontos.

Sugere-se definir para cada processo as seguintes informações:

- identificador
- datas de início e de conclusão
- duração (tempo necessário no processador)
- prioridades estática e dinâmica (se houver)
- estado atual (novo, pronto, executando, terminado)
- tempo já executado (total e no quantum atual)
- ... (outros campos podem ser necessários para algumas políticas de escalonamento)

Além disso, o programa de simulação deve abstrair o contexto de um processador (CPU) hipotético chamado INE5412 que possui 6 registradores de propósito geral, SP (stack pointer), PC (program counter) e ST (status), todos com 64 bits. Desta forma, cada abstração de processo também deve ter um contexto (como implementar o contexto de forma a permitir a mudança de processador para a abstração processo?). O programa de simulação deve então realizar a troca de contexto (salvar e restaurar o contexto do processo que estava em execução e do próximo processo a ser executado - Onde salvar o contexto do processo?)

O trabalho deve ser escrito em C++ e portanto deve apresentar as abstrações da simulação em

classes apropriadas e seus relacionamentos. A solução deve conter também uma estrutura de classes usadas para implementar os algoritmos de escalonamento a fim de reaproveitar o código e facilitar a adição de novas políticas no simulador (engenharia de software).

Formato de Entrega e Avaliação

O trabalho em duplas deverá ser entregue no Moodle no dia especificado pela tarefa. Todos os arquivos contendo o código do trabalho, bem como Makefile e um **relatório** apresentando **sucintamente** a solução e seu projeto (i.e., projeto OO, com diagramas UML), deverão ser submetidos pelo Moodle.

A avaliação se dará em 4 fases:

1. Avaliação de compilação: compilar o código enviado. Caso haja erros de compilação, a nota do trabalho será automaticamente zerada.
2. Avaliação de execução: para validar que a solução executa corretamente sem falhas de segmentação. Caso haja falhas de segmentação, a nota é zerada. Será também avaliado o uso de variáveis globais (-5 pontos) e vazamentos de memória (-20%).
3. Avaliação da organização do código: busca-se nesta fase avaliar a organização do código orientado a objetos e o seguimento das diretrizes do trabalho (saída, contexto, processo, CPU, troca de contexto, algoritmos de escalonamento). Deve-se usar classes e objetos e não estilo de programação baseado em procedimentos (como na linguagem C). Alguns itens para avaliação são: (i) funcionamento do programa; (ii) saída do programa (conforme especificação); (iii) clareza do código (utilização de comentários e nomes de variáveis adequadas); (iv) qualidade do relatório; (v) compilação sem warnings; (vi) sem vazamento de memória e (vii) modelagem do software desenvolvido com diagramas UML.
4. A quarta fase consiste na apresentação do trabalho em dia e horário agendado pelo professor. Durante as apresentações, o professor irá avaliar o **conhecimento individual dos alunos sobre os conteúdos teóricos e práticos vistos em aula e sobre a solução adotada no trabalho**. A nota atribuída à cada aluno i no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma, onde A_i é a nota referente à apresentação do aluno i e S é a nota atribuída à solução do trabalho:

$$NotaTrabalho_i = \frac{A_i \times S}{10}$$

Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos.

Caso a dupla apresente a saída da simulação utilizando uma interface gráfica, terá um bônus de até 1 ponto na nota do trabalho.

Referências

- Disciplina de Sistemas Operacionais do curso de Ciência da Computação da UFPR.
Professor Carlos Maziero.

- Sistemas Operacionais Modernos. 3ª edição. Andrew S. Tanenbaum. Pearson. 2010.