Disciplina: Sistemas Operacionais

Atividade Prática: Escalonamento de Processos

Objetivo: O objetivo deste projeto é escrever um programa para simular o escalonamento de um conjunto de tarefas usando os algoritmos de escalonamento de processador mais conhecidos. O programa deverá suportar os seguintes algoritmos:

- FCFS (First Come, First Served)
- Shortest Job First
- Shortest Remaining Time First
- Por prioridade, sem preempção
- Por prioridade, com preempção por prioridade
- Round-Robin com quantum, sem prioridade
- Round-robin com prioridade e envelhecimento

Ao final desta atividade, o aluno deverá ser capaz de:

- Compreender o funcionamento dos algoritmos de escalonamento
- Entender a diferença entre abordagens preemptivas e colaborativas
- Ver os resultados destes algoritmos em métricas de tempo de espera, tempo de execução e tempo de resposta para processos

Diretrizes para Implementação

O valores de quantum e taxa de envelhecimento devem ser estabelecidos por meio de um arquivo de configuração em texto plano, conforme formato abaixo:

```
quantum:2
aging:1
```

No caso do escalonamento round-robin com prioridade e envelhecimento, o envelhecimento deve ocorrer a cada quantum e não há preempção por prioridade. O programa deverá ler os dados dos processos da entrada padrão (stdin). Cada linha da entrada corresponde a um processo, com os seguintes dados fornecidos como inteiros separados por um ou mais espaços em branco:

- instante de criação
- duração em segundos
- prioridade estática (escala de prioridades positiva)

Um exemplo de entrada para o simulador poderia ser:

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

Nesse exemplo de entrada, o processo P1 tem data de criação 0, sua execução dura 5 segundos e sua prioridade é 2. Esse formato de entrada deverá ser respeitado, pois o professor pode testar seu simulador com outros dados de entrada. Observe que essa listagem não precisa necessariamente estar ordenada por data de criação de cada processo.

Para cada algoritmo, o simulador deverá produzir as seguintes informações em sua saída padrão (stdout):

- tempo médio de vida (*turnaround time*, tt)
- tempo médio de espera (*waiting time*, tw)
- número de trocas de contexto
- diagrama de tempo da 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               ##
```

Pontos importantes:

- O simulador pode ser escrito em C, Java, Javascript ou Python
- Cada equipe deve fornecer o código devidamente comentado para análise, juntamente com um documento explicando as decisões de implementação utilizadas (classes, estruturas de dados utilizadas, padrões de projeto (se for o caso). Sugere-se que cada processo possua alguma estrutura que mapeie informações para controle, como id, status, prioridade. Essa estrutura deve estar descrita no documento
- Em casos de haver um empate para escolha de um processo a ocupar o processador, deve-se usar a estratégia de alocar (i) o processo que já esteja com processador, para evitar troca de contexto, (ii) processo com menor tempo restante de processamento e , em último caso, (iii) escolha aleatória.
- **Trabalhos que optarem em fornecer uma interface visual terão um bônus de 2.5 pontos. Esta interface fica a critério da equipe, mas pode se basear em vários exemplos de simuladores de escalonamento existentes, especialmente com animações. Alguns exemplos:**
  https://process-scheduler.vercel.app/app
  https://cpu-scheduling-sim.netlify.app/
  https://www.youtube.com/watch?v=Rf6Ec8PjOyg