INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em ciência da computação
Arquitetura de computadores e sistemas operacionais
Atividade de implementação: algoritmos de escalonamento 30 de novembro de 2024

Instruções para entrega dos exercícios

- Você deve entregar **um** arquivo de código em C/C++ (`escalona.c` ou `escalona.cpp`).
- O trabalho pode ser realizado em grupos de até 3 pessoas.

Esta atividade foi adaptada de uma atividade de Carlos Maziero, UFPR.

1 Algoritmos de escalonamento

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

1. FCFS (*First Come, First Served*)
2. SJF (*Shortest Job First*)
3. PRIOp (por prioridade), com preempção
4. RR (*Round-robin*), com $quantum = 2$, sem prioridade e sem envelhecimento
5. RR (*Round-robin*), com $quantum = 2$, com prioridade e com envelhecimento ($\alpha = 1$)

O programa deverá ler os dados dos processos da entrada padrão (*stdin*, teclado). Cada linha da entrada corresponde a um processo, com os seguintes dados fornecidos como inteiros separados por um ou mais espaços em branco:

- data 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
- tempo médio de espera
- 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               ##
```

2 Sugestão de implementação

Esse é um pseudo-código que você pode usar como ponto de partida e ajustar, para cada política.

(confira a próxima página)

```
inicio
  le dados das tarefas da entrada padrao
  imprime cabecalho do diagrama

  t = 0
  enquanto t < tmax

    se ha uma tarefa rodando
      se a tarefa rodando chegou ao fim da execucao
        migra a tarefa para o estado terminado
        libera o processador
      senao
        se a tarefa rodando chegou ao fim de seu quantum
        migra a tarefa para a fila de prontos
        libera o processador
      fim se
    fim se

    para cada tarefa i
      se a tarefa i inicia agora (em t)
        coloca a tarefa na fila de prontos
      fim se
    fim para

    se o processador estiver livre
      se houver tarefa na fila de prontas
        escolhe uma tarefa da fila de prontas
        migra essa tarefa para o estado ‘‘rodando’’
      fim se
    fim se

    imprime linha do diagrama com o estado de cada tarefa

    incrementa o tempo (t++)
    incrementa contadores da tarefa corrente (tempo de vida e de quantum)

  fim enquanto

  calcula e imprime tempos medios
fim
```

Sugere-se definir para cada tarefa:

- identificador
- datas de inicio e de conclusão
- duração (tempo necessário no processador)
- prioridades estática e dinâmica
- estado atual (nova, pronta, rodando, terminada, ...)
- tempo já executado (total e no quantum atual)
- ... (outros campos podem ser necessários para algumas políticas)