INSTITUTO FEDERAL
Paraná
Campus Pinhais

Curso: Bacharelado em Ciência da Computação **Disciplina:** Arquitetura e Sistemas Operacionais
Trabalho 1, v1.0 3 de novembro de 2026

1 Introdução

Seu trabalho consiste em implementar alguns algoritmos de escalonamento de tarefas que estudamos em sala, e simular o seu funcionamento.

Um ponto importante é que, enquanto iremos focar no escalonamento de tarefas, vários subsistemas de um sistema operacional utilizam algoritmos muito parecidos ou idênticos. Exemplos de outros subsistemas com ideias similares são escalonamento do acesso ao disco, alocação de memória e escolha de páginas vítimas para *caches* e *swap*.

2 Algoritmos de escalonamento

Você deve implementar e simular os seguintes algoritmos de escalonamento de tarefas:

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

3 Entrada e saída

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 (números inteiros):

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

Um exemplo de entrada para o simulador é o seguinte:

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

No exemplo acima, o processo P1 tem tempo de criação 0, sua execução dura 5 segundos e sua prioridade é 2. Esse formato de entrada deverá ser respeitado, pois o professor irá testar seu simulador com outros dados de entrada, mas seguindo o mesmo formato. Perceba que os processos listados na entrada não precisam necessariamente estar ordenados por tempo de criação.

Para cada algoritmo, o simulador deverá imprimir na tela (*stdout*) as seguintes informações:

- tempo médio de vida das tarefas
- tempo médio de espera das tarefas
- número de trocas de contexto do sistema
- diagrama de tempo da execução (mostrado abaixo)

O diagrama de tempo de cada execução deve ser gerado na vertical, de cima para baixo (uma linha por segundo):

```
FCFS
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           ##

tempo_vida = 9.5
tempo_espera = 6
trocas_contexto = 5
<linha em branco>
```

O símbolo ## denota que a tarefa está executando naquele instante de tempo, −− denota que a tarefa está no sistema, esperando para executar, e os espaços em branco representam que a tarefa não está no sistema (ainda não foi iniciada, ou já terminou de executar).

Note que antes do diagrama, você dever imprimir o nome do algoritmo, como definido na seção anterior. O tempo deve sempre ocupar duas casas. Respeite a separação entre processos e o modelo de saída. Após o diagrama de tempo, você deve imprimir os tempos médios e as trocas de contexto. Por fim, imprima uma linha em branco para separar as informações de cada algoritmo.

Lembre-se: o professor irá comparar as saídas dos testes com gabaritos. Se a sua saída não respeitar o modelo acima, a comparação acusará erro!

4 Sugestão de implementação

Abaixo está uma sugestão, no formato de pseudo-código, que pode ser utilizada como ponto de partida para a implementação de cada política de escalonamento.

(confira a próxima página)

```
inicio
        le dados das tarefas da entrada padrao

        para cada algoritmo de escalonamento
                imprime cabecalho do diagrama

                t = 0
                enquanto t < t_max
                        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, tempo de processador, etc)
                fim enquanto

                calcula e imprime tempos medios
        fim para
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)

5 Entrega

Você deverá enviar para o email do professor, até 26/novembro, 23h59, um arquivo *.zip* contendo uma pasta, que por sua vez contém:

- O arquivo `escalona.cpp` que implementa as suas políticas de escalonamento e a simulação;
- Um arquivo de texto chamado `README.md` contendo o nome completo dos integrantes do grupo, informações sobre como compilar o programa, decisões de implementação, as referências de pesquisa e código, e outras informações que a equipe julgar importantes;

O arquivo compactado descrito acima deve ser nomeado com as iniciais do nome de cada integrante separados por um hífen. Por exemplo: se o Fulano da Silva Sousa fez o trabalho com o João de Souza, o arquivo deve ser nomeado `fss-js.zip`. A pasta, dentro do arquivo compactado, deve ter o mesmo nome, a não ser pela extensão.

O trabalho pode ser feito em grupos de até 3 pessoas.

5.1 Critérios de correção

Seu trabalho será corrigido considerando:

- Seu programa e as políticas devem funcionar corretamente!
- Objetividade e clareza do arquivo `README.md`.
- Uso dos métodos e construções adequadas de programação (estruturas de dados, funções, comentários, ...).
- Domínio do trabalho na defesa/acompanhamento, se houver.
- Adequação à esta especificação.

Não serão aceitos trabalhos entregues fora do prazo. Trabalhos sem `README.md` ou que não sejam defendidos/acompanhados, quando requisitados, não serão considerados!

Plágio não será tolerado. Se isto acontecer, os trabalhos envolvidos serão desconsiderados e o caso será encaminhado para as Coordenações do campus.

O uso de inteligência artificial pode acarretar na redução de nota parcial ou totalmente, se o professor identificar que não há domínio do trabalho proposto por parte do estudante.

Histórico das Revisões:

- 03/nov/2026 - v1.0: primeira versão.