Universidade Federal do Rio Grande do Norte - UFRN
Centro de Ensino Superior do Seridó - CERES
Departamento de Computação e Tecnologia - DCT

Curso: Bacharelado em Sistemas de Informação
Disciplina: DCT2101 – Sistemas Operacionais
Professor: João Borges
Data: 26 de maio de 2025

Atividade 2.1
Unidade 2 - Tarefa 1
Gerenciamento de Processos - Escalonamento

ATENÇÃO 1: Só serão aceitos trabalhos **Individuais** ou em **Dupla**, mais participantes invalidará o trabalho.

ATENÇÃO 2: Não serão permitidos plágios entre os componentes, sendo punidos, ambos os alunos que tiverem seus trabalhos iguais, com nota 0 (zero).

1. Esta atividade consiste em duas etapas:
   (a) Implementação de 5 algoritmos de Escalonamento de Processos; e
   (b) Análise de performance comparativa entre os algoritmos.

2. A implementação deverá ser baseada no Simulador de Escalonador disponível no repositório do endereço:

   `https://github.com/labepi/sched_sim`

   Este simulador já possui a implementação de um algoritmo de Escalonamento FIFO Preemptivo, que deverá ser utilizado como base para a implementação dos demais algoritmos, e que está disponível no arquivo:

   `scheduler_fifo.c`

3. A tarefa consistirá em, com base na implementação citada, implementar os seguintes algoritmos de escalonamento:
   (a) SJF (*Shortest Job First*)
   - O escalonador deverá, a cada rodada, selecionar o processo que possui menor tempo restante (`remaining_time`).

   (b) LJF (*Longest Job First*)
   - Este é um escalonador apenas de testes, pois pode produzir resultados não ideais.
   - Ele é o inverso do SJF.
   - O escalonador deverá, a cada rodada, selecionar o processo que possui maior tempo restante (`remaining_time`).

   (c) PRIO_STATIC: Com prioridades estáticas
   - Possui duas filas de prioridades
   - Não há realimentação entre as filas.
   - Os processos, ao serem iniciados, são inseridos sempre na mesma fila (`ready`), mas deverão ir para a sua fila de acordo com o que é estabelecido abaixo.
   - Ao saírem da execução, deve-se verificar em qual fila o processo deve ficar.
   - Para isto, considere os seguinte critérios:
     – Processos com menor tempo restante deverão ir para a primeira fila (`ready`).
     – Processos com maior tempo restante deverão ir para a segunda fila (`ready2`).

  - O limite para definir a fila do processo é de acordo com o tempo máximo de execução total de um processo, disponível na variável global `MAX_TIME`, definida no arquivo `main.c`.
  - Processos com `remaining_time` até 20% de `MAX_TIME` irão para a primeira fila, os demais irão para a segunda fila.
- ATENÇÃO: é importante definir a variável '`queue`' do processo indicando a qual fila ele pertence:
  - 0: primeira fila (`ready`)
  - 1: segunda fila (`ready2`).
- Sempre que um processo sair do processador ou da fila de bloqueados, devem voltar para a mesma fila de início.
- O escalonador deve utilizar a seguinte regra para selecionar um processo, de acordo com as probabilidades entre as filas:
  - Fila 1 (`ready`) - FIFO = 80% de probabilidade
  - Fila 2 (`ready2`) - FIFO = 20% de probabildiade

(d) PRIO_DYNAMIC: Com prioridades dinâmicas

- Possui duas filas de prioridades
- Há realimentação entre as filas.
- Os processos, ao serem iniciados, são inseridos sempre na mesma fila (`ready`), mas poderão mudar de fila de acordo com a sua execução.
- As regras para mudança de fila são as seguintes:
  - Fila 1 (`ready`) - FIFO = Processos que saíram por E/S, após voltar de bloqueado, deverão voltar para a primeira fila.
  - Fila 2 (`ready2`) - FIFO = Processos que saíram por preempção, devem retornar para a segunda fila.
- ATENÇÃO: é importante definir a variável '`queue`' do processo indicando a qual fila ele pertence:
  - 0: primeira fila (`ready`)
  - 1: segunda fila (`ready2`).
- O escalonador deve utilizar a seguinte regra para selecionar um processo, de acordo com as probabilidades entre as filas:
  - Fila 1 (`ready`) - FIFO = 80% de probabilidade
  - Fila 2 (`ready2`) - FIFO = 20% de probabildiade

(e) PRIO_DYNAMIC_QUANTUM: Com prioridades dinâmicas ajustáveis

- Possui duas filas de prioridades
- Há realimentação entre as filas.
- Os processos, ao serem iniciados, são inseridos sempre na mesma fila (`ready`), mas poderão mudar de fila de acordo com a sua execução.
- As regras para mudança de fila são as seguintes:
  - Fila 1 (`ready`) - FIFO = Processos que utilizaram mais de 50% do seu valor disponível de QUANTUM, deverão voltar para a primeira fila.
  - Fila 2 (`ready2`) - FIFO = Processos que utilizaram menos de 50% do seu valor disponível de QUANTUM, deverão voltar para a segunda fila.
- Utilize a variável global `QUANTUM` para a obtenção de seu valor.
- ATENÇÃO: é importante definir a variável '`queue`' do processo indicando a qual fila ele pertence:
  - 0: primeira fila (`ready`)
  - 1: segunda fila (`ready2`).
- O escalonador deve utilizar a seguinte regra para selecionar um processo, de acordo com as probabilidades entre as filas:
  - Fila 1 (`ready`) - FIFO = 80% de probabilidade
  - Fila 2 (`ready2`) - FIFO = 20% de probabildiade

4. Para a implementação dos algoritmos, os seguintes arquivos deverão ser modificados:

   - `scheduler_sjf.c` - implementação do SJF
   - `scheduler_ljf.c` - implementação do LJF
   - `scheduler_prio_static.c` - implementação do PRIO_STATIC
   - `scheduler_prio_dynamic.c` - implementação do PRIO_DYNAMIC
   - `scheduler_prio_dynamic_quantum.c` - implementação do PRIO_DYNAMIC_QUANTUM

   São permitidas modificações no código original, a fim de realizar as implementações exigidas. Mas, o professor deverá ser informado das modificações nos demais arquivos além dos listados acima.

5. Considerando que haverão implementações de 5 algoritmos disponíveis ao final da etapa de implementação, sendo 4 algoritmos produzidos e 1 algoritmo já implementado com o exemplo, a etapa de análise de performance consistirá em:

   (a) Executar o código de cada algoritmo variando-se linearmente a quantidade de processos:
   - Para cada algoritmo realizar no mínimo 10 rodadas para cada quantidade de processos;
   - Sendo a quantidade de processos a seguinte:
     `10 20 30 40 50 60 70 80 90 100`
   - Isto é, para cada algoritmo serão realizadas, no mínimo, 100 rodadas. Sendo um total de 500 simulações ao todo, para os 5 algoritmos.

6. Após a execução de cada rodada de simulação, deverá ser armazenado o valor do "Tempo Médio de Espera" (TME) para cada uma das rodadas, a fim de posterior análise estatística.

7. De posse das amostras do TME para cada simulação, deverão ser obtidas as seguintes medidas estatísticas:

   (a) Valor médio (μ) das amostras do TME, para cada par "algoritmo x quantidade de processos", que pode ser obtida pela seguinte equação:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1}$$

   Onde n é a quantidade de amostras e x_i é a *iésima* amostra obtida;

   (b) Variância (σ^2) entre as amostras do TME para cada um destes pares, obtida pela seguinte equação:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (\mu - x_i)^2 \tag{2}$$

8. De posse de toda a análise estatística, organizar os valores sob a forma de um gráfico:

   (a) Grafico - Tempo médio de espera (total)
   - Eixo X: a quantidade de processos
   - Eixo Y: o valor da média do tempo de espera (total)

   Os valores de variância serão utilizados para ilustrar os erros de cada uma das métricas simuladas, e deverão ser plotadas nos gráficos acima como barras de erro (errorbars).

9. Por fim, todas estas informações deverão ser organizadas sob a forma de relatório, onde deverão ser discutidas as implementações dos algoritmos, bem como uma análise dos gráficos apresentados (por exemplo, qual algoritmo possui menor TME, por qual motivo isto acontece, o que acontece à medida que o número de processos aumenta, ...).

   O modelo de relatório seguirá o *Modelo para publicação de artigos* da Sociedade Brasileira de Computação (SBC):

   `https://www.sbc.org.br/wp-content/uploads/2024/07/modelosparapublicaodeartigos.zip`

10. Os códigos-fonte dos algoritmos deverão ser enviados juntamente com o relatório da atividade. No entanto, para melhor explicar a sua implementação, no relatório poderão ser inseridos trechos do código, ou algoritmo, conforme sua necessidade.

11. O envio da atividade deverá ser feita pelo SIGAA, até a data estabelecida na tarefa cadastrada no sistema.