Grupos de 1 até 2 participantes: Entregar Tema 1 (Entrega de Refeições) e Tema 2 (SOSim)

Grupos de 1 até 4 participantes: Entregar Tema 1, Tema 2 e Tema Extra.

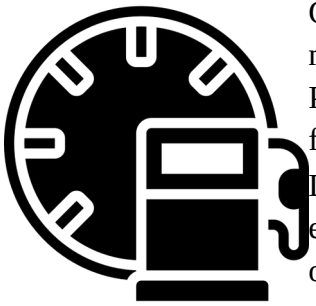
O que entregar?

- Etapa 1 – Código e relatório
- Etapa 2 – Vídeo
- Etapa Extra – Código e relatório.

OBSERVAÇÕES IMPORTANTES:

- Trabalho em grupo deve ser realizado por todos participantes.
- É importante ter em mente que ao participar de um grupo, está assinando por todo o trabalho. Logo, poderá ser questionado por qualquer atividade relacionada (Não será aceita a resposta: "eu não fiz essa parte, foi o fulano").
- Posso sortear algum grupo para uma apresentação extra via Google Meet. Caso isso ocorra, os participantes podem receber notas diferentes.
- É um trabalho em grupo e todos devem participar. Em casos de problemas no grupo, me enviem um e-mail e esse grupo será um forte candidato a ser convocado para apresentação via Google Meet.
- Problemas com cópia/compartilhamento de trabalho (mesmo parcial), a nota será zerada.

ETAPA 1 – Entrega de refeições visando economia de combustível



Considere a seguinte situação: Brasil tem a 3ª gasolina mais cara do mundo, diz Oxford Economics. O preço médio do litro da gasolina no País fechou o mês de março a R$ 7,323. Por esse motivo, é cada vez mais frequente a busca de alternativas para economizar combustível.

Dado esse panorama, vamos supor que uma grande empresa do ramo de entrega de comidas pela internet (por exemplo, ifood), definiu que todos os pedidos de alimentos de restaurantes do Laranjal com destino para o centro de Pelotas, devem ser concentrados em uma central de distribuição do Laranjal (ponto A), para serem enviados em conjunto para uma central de distribuição localizada no centro (ponto B). Essa solução não é adequada pensando no cliente e, também, por se tratar de alimento. Mas vamos considerá-la para fins didáticos.

Esta atividade consiste em utilizar Programação Concorrente para simular a entrega de alimentos de um ponto A até um ponto B, mas só quando o número de refeições acumuladas chegar em 10 (nenhuma a mais e nenhuma a menos). Características que precisam ser levadas em consideração:

- Inicialmente o entregador fica em A, esperando que existam 10 refeições para transportar, e dormindo quando não houver. Quando a quantidade de refeições a serem entregues chegar a 10, o entregador (motorista) deve levá-las de A para B, e, em seguida, deverá retornar para o ponto A.
- No retorno, caso existam outras 10 refeições a espera, ele partirá imediatamente para realizar as entregas no ponto B; Caso contrário, ele dorme de novo até que existam as 10 refeições.
- As refeições são acomodadas pelos funcionários dos restaurantes dentro do baú da moto do entregador; Portanto, caso o entregador tenha saído, os funcionários dos restaurantes precisam aguardar o retorno para armazenar corretamente as refeições que serão enviadas para o ponto B.
- O funcionário do restaurante, ao acomodar a décima refeição, deverá acordar o entregador, caso esteja dormindo.

Utilizando semáforos e/ou mutex, modele o processo funcionário e o processo entregador, lembrando que existe um único entregador e diversos funcionários de restaurantes (um funcionário de cada restaurante). Identifique regiões críticas na vida do entregador e dos funcionários.

- Para a simulação, defina a quantidade de funcionários (idêntica a quantidade de restaurantes) e adote tempos aleatórios entre os novos pedidos, que serão entregues no ponto A.
- Defina um critério para encerrar a simulação (tempo?/meta?) e informe o total de refeições entregues (e, talvez, não entregues).
- Imprima tudo que ocorrer na simulação (Por exemplo: "Chegou refeição do Restaurante A", "Entregador saiu para realizar a entrega no ponto B", ….). Utilize a função sleep em pontos estratégicos para "suavizar" a impressão das informações na tela.

ETAPA 2 – Simulador SOsim

Com o objetivo de explorar os conceitos estudados de Sistemas Operacionais, utilize o simulador SOsim[1] (funciona no Linux com o uso do Wine) e crie um vídeo apresentando todos os itens listados abaixo:

a) Visão geral do simulador

- Apresente as principais janelas e funcionalidades

b) Sobre processos

- Utilize no mínimo dois tipos de processos: CPU-*bound* e I/O-*bound*.
- No simulador, quais são os possíveis estados para os dois tipos de processos acima?
- Explique sobre fatia de tempo, clock e qual impacto desses conceitos no simulador?
- Demonstre o uso do "escalonamento circular" (dica: é um parâmetro do sistema).
- Faça o seguinte experimento:

  Utilize escalonamento circular com Prioridade estática e crie:

  - 2 processos com prioridade 3 para I/O;
  - 2 processos com prioridade 2 para misto;
  - 2 processos com prioridade 1 para CPU.

  E se as prioridades forem definidas ao contrário?

c) Gerência de Memória

- Mostre experimentos trocando a política de busca (paginação antecipada e paginação por demanda). Sugestão: Também, analise a quantidade de "*page fault*" no arquivo de log por um determinado período de tempo.
- Mostre as janelas de gerência de memória, paginação e log.
- Apresente e analise a Tabela de Páginas (PCB → ver tab. de páginas)

d) Conclusão

- Faça uma análise crítica do simulador SOSim.
- Compare os conceitos vistos em aula com o SOSim.
- Existe outro simulador disponível? Descreva brevemente.

1 http://www.training.com.br/sosim/

ETAPA EXTRA (Somente para grupo com até 4 participantes) – Desenvolvimento de um escalonador

Crie um programa para simular o escalonamento de um conjunto de tarefas conhecidas (é de conhecimento o tempo de execução de cada tarefa).

O programa deverá receber como parâmetro na linha de execução: nome do arquivo que possui informações sobre as tarefas que serão escalonadas (cada linha apresenta o nome da tarefa e um número inteiro que representa o tempo de execução) e um número que indicará a quantidade de processadores que se deseja utilizar na simulação.

Implemente dois algoritmos de escalonamento: o primeiro é o clássico SJF (*Shortest Job First*) que irá executar as menores tarefas primeiro. O segundo algoritmo é o oposto do SJF, executando as maiores tarefas primeiro.

Como saída espera-se dois arquivos (um para cada escalonamento). O arquivo deve conter o id do processador e o nome de cada tarefa com o instante inicial e final (em segundos). Respeite EXATAMENTE o formato do exemplo abaixo, pois essa saída será entrada de outro programa para validação do resultado.

Por exemplo, forma de executar o programa: ./trabalho_escalonador tarefas.txt 2

Conteúdo do arquivo de entrada: “tarefas.txt” (nome da tarefa e tempo total de execução separados pelo caractere espaço).

```
a1      5
a2      1
a3      10
b1      10
b2      3
b3      7
b4      8
c1      8
c2      2
```

SAÍDA1 (menor_primeiro.txt):

```
Processador_1
a2;0;1
b2;1;4
b3;4;11
c1;11;19
b1;19;29

Processador_2
c2;0;2
a1;2;7
b4;7;15
a3;15;25
```

SAÍDA 2 (maior_primeiro.txt)

```
Processador_1
a3;0;10
b4;10;18
b3;18;25;
c2;25;27

Processador_2
b1;0;10
c1;10;18
a1;18;23
b2;23;26
a2;26;27
```