UNIVERSIDADE FEDERAL DO PARANÁ
ENGENHARIA DE ELÉTRICA
TE355 -*Sistemas Operacionais Embarcados*
Primeira Prova - 18/10/2024

Nome: _____

Por favor, desligue seu celular. Não será permitido o uso de calculadoras. As questões podem ser respondidas a caneta ou lápis. Esta prova possui 7 questões, com um total de 100 pontos

10 1. A seguir estão algumas afirmações sobre as funções de um sistema operacional. Para cada uma delas, indique se é **Verdadeiro** (V) ou **Falso** (F), justificando a sua resposta:

1. (V) O sistema operacional é responsável por gerenciar os recursos de hardware de um computador, como CPU, memória e dispositivos de entrada/saída.
2. (F) Um dos papéis do sistema operacional é fornecer bibliotecas para permitir que as aplicações possam acessar diretamente os recursos de hardware do sistema.
3. (V) Sistemas operacionais podem oferecer suporte para execução de múltiplos processos simultaneamente, utilizando técnicas como *multitasking*.
4. (V) A proteção e segurança dos dados armazenados no sistema são funções do sistema operacional.
5. (F) Em um sistema de tempo compartilhado, o sistema operacional deve garantir que todos os processos tenham exatamente a mesma quantidade de tempo de CPU.

Atenção. nesta questão não há nota parcial.

1. (V) Motivo: Essa é uma das funções primárias de um sistema operacional. Ele age como um intermediário entre o hardware e as aplicações, gerenciando recursos como CPU (escalonamento), memória (alocação/desalocação), e dispositivos de I/O (controle de acesso).

2. (F) Motivo: O sistema operacional não permite acesso direto ao hardware. Em vez disso, ele fornece APIs ou chamadas de sistema (system calls) que permitem às aplicações solicitar recursos de maneira controlada e segura. Bibliotecas que acessam o hardware são geralmente fornecidas por frameworks ou fabricantes de dispositivos, não diretamente pelo sistema operacional.

3. (V) Motivo: Sistemas operacionais modernos implementam multitarefa (multitasking), permitindo que vários processos sejam executados "simultaneamente" (de forma concorrente em sistemas monoprocessados ou em paralelo em sistemas multiprocessados). Isso é feito por meio de técnicas como escalonamento de processos e gerenciamento de threads.

4. (V) Motivo: O sistema operacional oferece mecanismos de proteção e segurança, como controle de acesso a arquivos e recursos, criptografia, autenticação de usuários, e isolamento de processos, para proteger os dados armazenados e evitar acessos não autorizados.

5. (F) Motivo: Sistemas de tempo compartilhado buscam garantir que todos os processos tenham acesso à CPU, mas a quantidade de tempo pode variar de acordo com fatores como prioridade, comportamento do processo (CPU-bound ou I/O-bound), e políticas de escalonamento. O objetivo é um uso eficiente e justo dos recursos, não necessariamente um tempo igual para todos os processos.

10 2. A arquitetura CISC (Complex Instruction Set Computer) possui características que a diferenciam de outras arquiteturas, como a RISC (Reduced Instruction Set Computer), especialmente em relação à execução de *pipeline*. Explique por que a arquitetura CISC encontra dificuldades para implementar de forma eficiente um *pipeline* comparado à arquitetura RISC.

Pipeline é uma técnica de arquitetura de computadores que organiza o processamento de instruções de forma sequencial e sobreposta, dividindo a execução de uma tarefa em múltiplos estágios. Cada estágio do pipeline executa uma parte específica da tarefa, permitindo que várias instruções sejam processadas simultaneamente, mas em diferentes etapas.

Em CISC, instruções podem variar amplamente em complexidade. Por exemplo:

- Uma simples operação de soma pode levar 1 ciclo.
- Uma instrução de acesso à memória (como MOV [MEM]) pode levar vários ciclos, dependendo da latência do barramento de memória.
- Essa imprevisibilidade complica o planejamento do pipeline, já que o processador não sabe com antecedência quantos ciclos serão necessários para concluir a instrução em execução.

Instruções que demoram mais ciclos para serem concluídas introduzem dependências de dados que interrompem o fluxo do pipeline:

- Instruções subsequentes que dependem do resultado da instrução longa não podem prosseguir sem que essa seja concluída.
- Isso leva a atrasos adicionais, exacerbando o problema de eficiência.

O pipeline opera com a premissa de que cada estágio será executado em um intervalo fixo de ciclos de clock. No entanto, se uma instrução requer um número variável de ciclos para ser concluída (por exemplo, devido a operações complexas como multiplicação, divisão ou acesso à memória), os estágios seguintes do pipeline ficam "ociosos", aguardando a conclusão dessa instrução. Isso resulta em bolhas no pipeline (pipeline stalls), reduzindo a eficiência geral.

Esses fatores tornam o design de pipelines em CISC mais desafiador. No entanto, arquiteturas modernas baseadas em CISC (como x86) implementam estratégias avançadas, como tradução dinâmica de instruções para micro-operações RISC-like, a fim de mitigar essas limitações e alcançar um desempenho competitivo.

20 3. Considere um sistema operacional que utiliza o algoritmo de escalonamento Round Robin para gerenciar processos em um ambiente de tempo compartilhado. Cada processo tem um *quantum* de tempo fixo para execução antes de ser preemptado, e é inserido novamente na fila de prontos (ready queue) se não tiver terminado. Neste sistema, é aplicado um mecanismo de incremento de prioridade dinâmica:

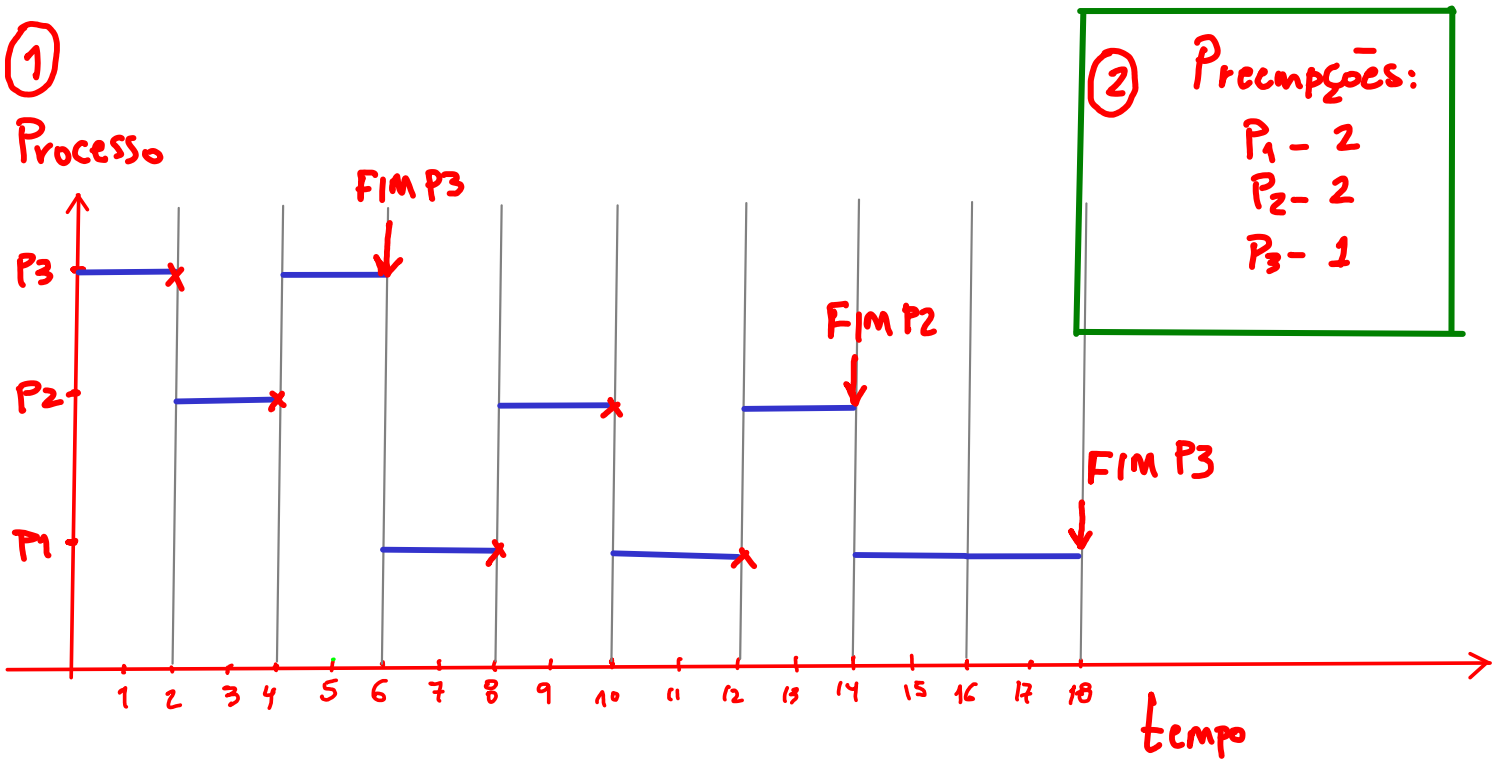
- Cada vez que um processo ocupar a CPU por um quantum de tempo, ele é preemptado e colocado de volta na fila de prontos, sua prioridade reduz em um nível.
- Os processos com maior prioridade são escolhidos para execução antes daqueles com prioridade mais baixa.

Dado o cenário descrito, considere os seguintes processos com suas respectivas prioridades iniciais (quanto menor o número, maior a prioridade):

Processo	Prioridade Base	Tempo de Execução Total
P_1	3	8 unidades de tempo
P_2	2	6 unidades de tempo
P_3	1	4 unidades de tempo

Suponha que o *quantum* de tempo seja de 2 unidades de tempo e que, inicialmente, todos os processos estão prontos para execução. O processo com a menor prioridade inicial (ou seja, P_3) começa a executar.

1. Descreva a sequência de execução dos processos, considerando o incremento de prioridade dinâmica e a política de Round Robin.
2. Determine quantas vezes cada processo é preemptado antes de terminar sua execução.
3. Explique como o incremento de prioridade dinâmica influencia a sequência de execução em comparação com um Round Robin sem este incremento.



Em um sistema operacional com time sharing, a política de escalonamento define a ordem em que os processos utilizam a CPU. O incremento de prioridade dinâmica influencia significativamente essa sequência em comparação com um esquema simples como o Round Robin (RR) sem incrementos de prioridade.

Incremento de Prioridade Dinâmica:

- A prioridade de cada processo pode mudar dinamicamente, geralmente com base no tempo de espera ou no comportamento do processo:
    * Processos que esperam muito tempo pela CPU, por exemplo aguardando I/O, têm sua prioridade dinâmica aumentada.
    * Processos que consomem muito tempo de CPU podem ter sua prioridade reduzida.
    * Processos com maior prioridade são escalonados antes de processos com menor prioridade.
    * Processos que frequentemente aguardam I/O têm suas prioridades aumentadas, permitindo que retomem a execução rapidamente após o término de suas operações de entrada/saída.

- Processos de baixa prioridade que ficam muito tempo sem executar eventualmente terão suas prioridades aumentadas, garantindo sua execução.
- O sistema se adapta automaticamente ao comportamento dos processos, priorizando aqueles mais críticos ou que esperam por mais tempo.
- Processos CPU-bound podem ter sua prioridade reduzida, prevenindo que monopolizem a CPU.

10 4. Explique o funcionamento do algoritmo de escalonamento *Rate Monotonic* (RM) para sistemas de tempo real. Considere três tarefas periódicas T_1, T_2 e T_3, com períodos $P_1 = 3$ ms, $P_2 = 5$ ms e $P_3 = 10$ ms, e tempos de execução $C_1 = 1$ ms, $C_2 = 1$ ms e $C_3 = 2$ ms, respectivamente. Determine se essas tarefas podem ser escalonadas de forma viável utilizando o algoritmo *Rate Monotonic*. Justifique sua resposta mostrando os cálculos necessários.

(1) A utilização do sistema deve ser menor que 100%

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} < 0 \quad \therefore \quad U = \frac{1}{3} + \frac{1}{5} + \frac{2}{10}$$

$$U = \frac{10+6+6}{30} = \frac{22}{30} = \frac{11}{15}$$

$$U = 0{,}733 \text{ ou } 73{,}3\%$$

Critério necessário mas não suficiente ⇒ Ok, mas ainda tem que continuar

(2) Com o algoritmo Rate Monotonic, $U \leq N(2^{1/N} - 1)$

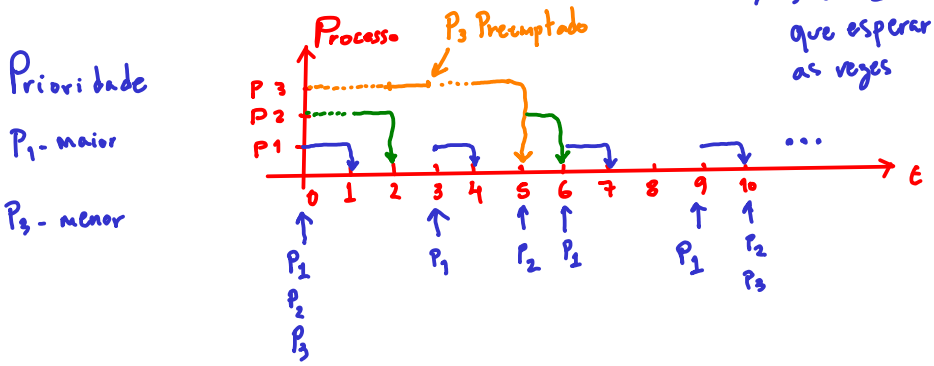
$$\therefore \quad U \leq 3(2^{1/3} - 1) \quad \therefore \quad U \leq 0{,}779$$

Critério necessário ⇒ Ok, mas ainda tem que continuar
mas não suficiente

(3) Para confirmar tem que analisar se os deadlines podem ser cumpridos - no entanto, a questão não apresenta os deadlines

Neste caso, teria que usar um diagrama de grant para analisar - ex:

⇒ P_3 e P_2 tem que esperar as vezes

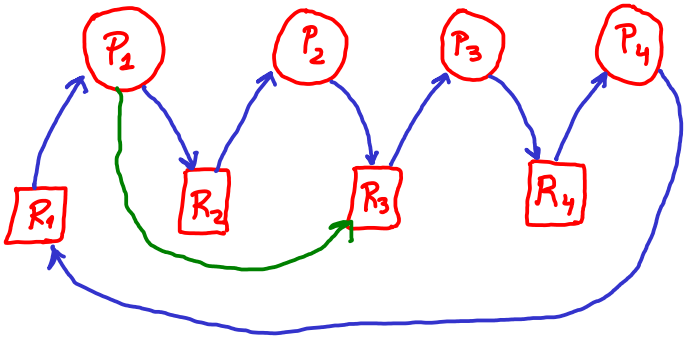


10 5. Considere um sistema com quatro processos, P_1, P_2, P_3, e P_4, e quatro recursos, R_1, R_2, R_3, e R_4. Suponha que:

- P_1 tenha alocado R_1 e esteja aguardando por R_2.
- P_2 tenha alocado R_2 e esteja aguardando por R_3.
- P_3 tenha alocado R_3 e esteja aguardando por R_4.
- P_4 tenha alocado R_4 e esteja aguardando por R_1.
- P_1 também requisitou R_3.

Desenhe o grafo de alocação de recursos correspondente a essa situação e explique se o sistema está em *deadlock*. Justifique sua resposta e identifique os processos e recursos envolvidos.

Grafo de alocação de recursos:



- O caminho fechado no grafo orientado (em azul) indica um deadlock!

6. O sistema de arquivos FAT-16 é utilizado em dispositivos de armazenamento. Considere um disco formatado com FAT-16 e as seguintes especificações:

   - Tamanho total da partição (ou do volume): 2 GB
   - Tamanho do bloco lógico (ou cluster): 32 KB

   Responda às perguntas a seguir:

10 (a) **Espaço Utilizado por Arquivos**: Suponha que você tenha 100 arquivos no disco, onde cada arquivo possui um tamanho médio de 25 KB. Determine quanto espaço em disco será utilizado por esses arquivos, considerando que o sistema de arquivos aloca espaço por bloco.

10 (b) **Capacidade Máxima de Armazenamento**: Discuta se o disco poderia armazenar um *arquivo único* de 1.5 GB e explique por quê.

a) 100 arquivos com bloco lógico de 32k Bytes

$S = 100 \cdot 32k \therefore$ **S = 3,2 M Bytes**

b) Tamanho máximo: 2^{16} blocos. Bloco de $\frac{32k \text{ Bytes}}{2^{15}}$

Máximo = $2^{16} \cdot 2^{15} = 2^{31} = 2$ G Bytes

Logo, é possível

20 7. Considere dois processos P_1 e P_2 que devem imprimir mensagens na tela em uma sequência específica. O processo P_1 imprime a mensagem "A" e o processo P_2 imprime a mensagem "B", em loop infinito. Deseja-se que a saída dos processos seja na seguinte ordem:

A B A A B A B A A B ...

Para alcançar essa sequência, utilize semáforos para sincronizar os processos. Considere que os semáforos são representados por *sem*1, *sem*2, e *sem*3, que são inicializados com valores 1, 0 e 0, respectivamente.

- Escreva um pseudocódigo para os processos P_1 e P_2, utilizando as operações `UP` e `DOWN` para garantir que a saída ocorra na sequência desejada.

Solução 1

Variáveis globais:

```
Semáforo S1=1, S2=0, M=1;
inteiro c=0;
```

Processo P_1

```
While (true)
{
    down(S1);
    down(M);
    print("A");
    c++;
    up(M);
    up(S2);
}
```

Processo P_2

```
While (true)
{
    down(S2);
    down(M);
    if (c%2)
        print("B");
    up(M);
    up(S1);
}
```

Solução 1

Variáveis globais:

```
Semáforo S1=1, S2=0, S3=0;
```

Processo P_1

```
While (true)
{
    down(S1);
    print("A");
    up (S2);
    up (S3);
}
```

Processo P_2

```
While (true)
{
    down(S2);
    print ("B");
    down (S3);
    up (S1);
    up (S1);
    down (S3);
    UP (S2);
}
```