

Exercícios e Provas Antigas!

p/ a 1^a prova de SO

Sophia Carrazza

Avaliação SO - Mostre seu raciocínio
NÃO serão aceitas respostas que apresentem apenas valores!
Não serão respondidas perguntas durante a prova!

- 1) Um sistema possui 3 processos que estão prontos para serem executados. Os tempos de execução (burst time) e os tempos de chegada (arrival time) dos processos são os seguintes:

- **Processo A:** Burst Time = 8 ms, Arrival Time = 0 ms
- **Processo B:** Burst Time = 4 ms, Arrival Time = 1 ms
- **Processo C:** Burst Time = 6 ms, Arrival Time = 2 ms

O sistema utiliza um quantum de 3 ms para o escalonamento Round Robin. Responda as seguintes perguntas:

- a) Calcule o tempo de resposta de cada processo.
- b) Calcule o tempo de espera de cada processo.
- c) Calcule o tempo de retorno de cada processo.

- 2) Suponha que os seguintes processos cheguem para execução nos tempos indicados abaixo (para responder as questões use um escalonamento não-preemptivo):

| Processo | Tempo de Chegada | Tempo em Execução |
|----------|------------------|-------------------|
| P1 | 0.0 | 8 |
| P2 | 0.5 | 4 |
| P3 | 1.0 | 1 |

- a) Calcule o tempo de retorno de cada processo para uma política que use escalonamento SJF.
- b) Calcule o tempo de retorno considerando que a CPU irá ficar inativa por 1 unidade assim que o primeiro processo chegar e a política SJF for então aplicada.

Este algoritmo é conhecido como: *future-knowledge scheduling*.

- 3) Sabe-se que um sistema é escalonável se diferentes tipos de eventos, digamos A, B, C, e etc. ocorrem de tal maneira que o evento A1 terá sido completamente processado antes de A2 ocorrer novamente. Da mesma forma, se B2 e C2 ocorrem tem-se a certeza que B1 e C1 já encerraram suas respectivas execuções - B2 começa com a certeza que B1 foi concluído. Raciocínio análogo segue para C2. Considere 5 processos periódicos com duração de 60, 100, 20 e 10 ms. Sabendo-se que o período de cada processo é de 240, 200, 100 e X ms respectivamente, qual o menor valor que X pode assumir para que o sistema seja escalonável?

- 4) O IBM 360/370 possuía uma função TST (Test and Set). O código desta função é dado a seguir (considere passagem de parâmetro por referência):

```
TST (int aux) { aux = C; C = 1; }
```

Onde C é uma variável global de sistema (Condition Code). Esta função é garantida "atômica". Mas o que aconteceria no programa a seguir se a função TST fosse substituída pela sequência equivalente (corpo da função)?

```
void p1( )
{
    int v1;
    repeat
        comando1;
        repeat
            TST1 (v1);
        until v1 = 0;
Região Crítica 1
        C = 0;
        comando2;
    forever;
}
```

```
void p2( )
{
    int v2;
    repeat
        comando3;
        repeat
            TST2 (v2);
        until v2 = 0;
Região Crítica 2
        C = 0;
        comando4;
    forever;
}
```

- 5) Considere a situação em que 4 processos: P1, P2, P3 e P4 concorrem por recursos em um sistema que possui 2 unidades de USB, 2 unidades de disco e 2 unidades de fita. Os processos estão na seguinte situação:

- O processo P1 está de posse de uma USB e requisita uma unidade de disco.
- O processo P2 está de posse de uma unidade de disco e requisita uma unidade de fita.
- O processo P3 está de posse de uma unidade de fita e requisita a outra unidade de disco.
- O processo P4 está de posse de uma USB e requisita uma unidade de fita.

Esta situação leva a um deadlock? Por quê? Mostre o grafo de alocação de recursos!

Altera-se a situação se o processo P2 pudesse requisitar uma USB e o processo P1 pudesse requisitar uma unidade de fita? Por quê?

- 1) Um sistema possui 3 processos que estão prontos para serem executados. Os tempos de execução (burst time) e os tempos de chegada (arrival time) dos processos são os seguintes:

- Processo A: Burst Time = 8 ms, Arrival Time = 0 ms
- Processo B: Burst Time = 4 ms, Arrival Time = 1 ms
- Processo C: Burst Time = 6 ms, Arrival Time = 2 ms

O sistema utiliza um quantum de 3 ms para o escalonamento Round Robin. Responda as seguintes perguntas:

- a) Calcule o tempo de resposta de cada processo.
- b) Calcule o tempo de espera de cada processo.
- c) Calcule o tempo de retorno de cada processo.

| Processo | Tempo de texec. | Chegada | Quantum = 3ms |
|----------|-----------------|---------|---------------|
| A | 8 ms | 0 ms | |
| B | 4 ms | 1 ms | |
| C | 6 ms | 2 ms | XBCA |

$$\begin{aligned}
 A &\rightarrow 3/8 \rightarrow \text{falta } 5 \quad 0-3 \ A \ \text{m+2} \ m \ L)_4 \\
 B &\rightarrow 3/4 \rightarrow \text{falta } 1 \quad 3-6 \ B \ \text{m } 3)_4 \\
 C &\rightarrow 3/6 \rightarrow \text{falta } 3 \quad 6-9 \ C \ \text{m } \\
 A &\rightarrow 3+3/8 \rightarrow \text{falta } 2 \quad 9-12 \ A \ M) + 6 + 4 \\
 B &\rightarrow 3+2/4 \rightarrow \text{acebar} \quad 12-13 \ B \\
 C &\rightarrow 3+3/6 \rightarrow \text{acebar} \quad 13-16 \ C \\
 A &\rightarrow 3+3+2/8 \rightarrow \text{acebar} \quad 16-18 \ A
 \end{aligned}$$

a) $A = 0 \text{ ms}; B = 3-1 = 2 \text{ ms}; C = 6-2 = 4 \text{ ms}$

b) $A \rightarrow \text{esperou } B \text{ e } C \text{ por } 6 \text{ ms e depois por } +4 \text{ ms} \Rightarrow = 10 \text{ ms}$

$B \rightarrow \text{quando chegou, esperou } 2, \text{ depois } +6 \Rightarrow = 8 \text{ ms}$

$C \rightarrow \text{esperou } 4 \text{ depois que chegou, depois } +4 \Rightarrow = 8 \text{ ms}$

c) $A = 18 \text{ ms} (\text{terminou em } 18 \text{ ms e chegou em } 0 \text{ ms})$
 $B = 12 \text{ ms } (13 - 1)$
 $C = 14 \text{ ms } (16 - 2)$

2) Suponha que os seguintes processos cheguem para execução nos tempos indicados abaixo (para responder as questões use um escalonamento não-preemptivo):

| Processo | Tempo de Chegada | Tempo em Execução |
|----------|------------------|-------------------|
| P1 | 0.0 | 8 |
| P2 | 0.5 | 4 |
| P3 | 1.0 | 1 |

- a) Calcule o tempo de retorno de cada processo para uma política que use escalonamento SJF.
- b) Calcule o tempo de retorno considerando que a CPU irá ficar inativa por 1 unidade assim que o primeiro processo chegar e a política SJF for então aplicada. Este algoritmo é conhecido como: *future-knowledge scheduling*.

2) Como é não-preemptivo, não processa de rodar um processo até que ele termine
 P1 → chegou primeiro → 8ms ← nem 8ms, P2 e P3 já chegaram
 P3 → (menor que P2) → 1ms (8+1)
 P2 → ultmo → 4ms (9+4)

a) Tempos de Retorno:

$$P1 \Rightarrow 8\text{ms} (8-0)$$

$$P3 \Rightarrow 8,5\text{ms} (9-0,5)$$

$$P2 \Rightarrow 12\text{ms} (13-1)$$

b) 1ms inativa → P1, P2 e P3 já chegaram. Logo, a ordem é dada pelo tempo de execução dos maiores burst times.

$$P3 \Rightarrow 1\text{ms}$$

$$P2 \Rightarrow 4\text{ms} (1+4) = 5\text{ms}$$

$$P1 \Rightarrow 8\text{ms} (5+8) = 13\text{ms}$$

Tempos de Retorno:

$$P3 \Rightarrow 1\text{ms} (\text{atraso}) + 1\text{ms} (\text{execucao}) - 1\text{ms} (\text{tempodeinatividade}) = 1\text{ms}$$

$$P2 \Rightarrow 5\text{ms} + 1\text{ms} - 0,5\text{ms} = 5,5\text{ms}$$

$$P1 \Rightarrow 13 + 1 - 0 \rightarrow 14\text{ms}$$

Escalarabilidade de processos periódicos

mais preemptivo

3) Sabe-se que um sistema é escalonável se diferentes tipos de eventos, digamos A, B, C, e etc. ocorrem de tal maneira que o evento A1 terá sido completamente processado **antes de A2 ocorrer novamente**. Da mesma forma, se B2 e C2 ocorrem tem-se a certeza que B1 e C1 já encerraram suas respectivas execuções - B2 começa com a certeza que B1 foi concluído. Raciocínio análogo segue para C2. Considere 5 processos periódicos com duração de 60, 100, 20 e 10 ms. Sabendo-se que o período de cada processo é de 240, 200, 100 e X ms respectivamente, qual o menor valor que X pode assumir para que o sistema seja escalonável?

| | Tempo Exec. | Período |
|----------------|-------------|---------|
| P ₁ | 60 | 240 |
| P ₂ | 100 | 200 |
| P ₃ | 20 | 100 |
| P ₄ | 10 | X |

menor valor de X p/ que o sistema seja escalonável

* Um sistema é escalonável se todos os processos puderem ser executados dentro dos seus respectivos períodos.

- ↳ cada processo deve finalizar execução antes do final de seu período
- ↳ o tempo total necessário p/ executar todos os processos não pode exceder o intervalo de tempo em que todos os períodos se repetem (o MMC dos períodos).

$$\begin{array}{r} 2 \\ 18 \\ 19 \\ \hline 54 \end{array}$$

$$\begin{array}{r} 240 \\ 60 \\ 30 \\ 15 \\ 5 \\ \hline 2 \end{array} \quad \left. \begin{array}{l} 4 \\ 2 \\ 2 \end{array} \right\} 2^4$$

$$\begin{array}{r} 200 \\ 50 \\ 25 \\ 5 \\ \hline 2 \end{array} \quad \left. \begin{array}{l} 4 \\ 2 \\ 5 \end{array} \right\} 2^3 \cdot 5^2$$

$$\begin{array}{r} 100 \\ 25 \\ 5 \\ \hline 2 \end{array} \quad \left. \begin{array}{l} 4 \\ 5 \\ 5 \end{array} \right\} 2^2 \cdot 5^2$$

MMC (240, 200, 100) → menor intervalo de tempo em que os três se repetem.

↳ fatoração dos números } MMC é dado pelo produto dos maiores fatores primos:

$$\left. \begin{array}{l} 240 \Rightarrow 2^4 \cdot 3 \cdot 5 \\ 200 \Rightarrow 2^3 \cdot 5^2 \\ 100 \Rightarrow 2^2 \cdot 5^2 \end{array} \right\} 2^4 \cdot 3 \cdot 5^2 = 1200$$

OU: (+ fácil)

$$\begin{array}{r} 240, 200, 100 \\ 60, 50, 25 \\ 30, 25, 25 \\ 15, 25, 25 \\ 3, 5, 5 \\ 3, 1, 1 \\ 1, 1, 1 \end{array} \quad \left| \begin{array}{r} 4 \\ 2 \\ 2 \\ 5 \\ 5 \\ 3 \\ 1 \end{array} \right\} \begin{array}{r} 16 \\ 8 \\ 8 \\ 400 \\ 3 \\ 1200 \\ 1200 \end{array}$$

★ Determinaremos a carga total do sistema (o tempo necessário para executar todos os processos).

$$\rightarrow P_1 \Rightarrow 60$$

$$n^{\circ} \text{ de repetição} \Rightarrow \frac{1200}{240} = 5$$

$$\text{tempo total necessário} = 60 \cdot 5 = 300$$

$$\rightarrow P_2 \Rightarrow 100$$

$$n^{\circ} \text{ de repetição} \Rightarrow \frac{1200}{200} = 6$$

$$\text{tempo total necessário} = 100 \cdot 6 = 600$$

$$\rightarrow P_3 \Rightarrow 20$$

$$\rightarrow \frac{1200}{100} = 12$$

$$\rightarrow \text{tempo total necessário} = 20 \cdot 12 = 240$$

$$\text{tempo total} = 1200 = 300 + 600 + 240 + T_T$$
$$1200 = 1140 + T_T$$

$$T_T = 60 = \text{tempo total necessário de } P_4$$

$$\frac{1200}{X} \leq 60$$

$$X \geq 200$$

- 4) O IBM 360/370 possuía uma função TST (Test and Set). O código desta função é dado a seguir (considere passagem de parâmetro por referência)

TST (int aux) { aux = C; C = 1; }

Onde C é uma variável global de sistema (Condition Code). Esta função é garantida "atômica". Mas o que aconteceria no programa a seguir se a função TST fosse substituída pela sequência equivalente (corpo da função)?

```
void p1( )
{
    int v1;
    repeat
        comando1;
        repeat
            TST1 (v1);
            until v1 = 0;
Região Crítica 1
            C = 0;
            comando2;
    forever;
}
```

```
void p2( )
{
    int v2;
    repeat
        comando3;
        repeat
            TST2 (v2);
            until v2 = 0;
Região Crítica 2
            C = 0;
            comando4;
    forever;
}
```

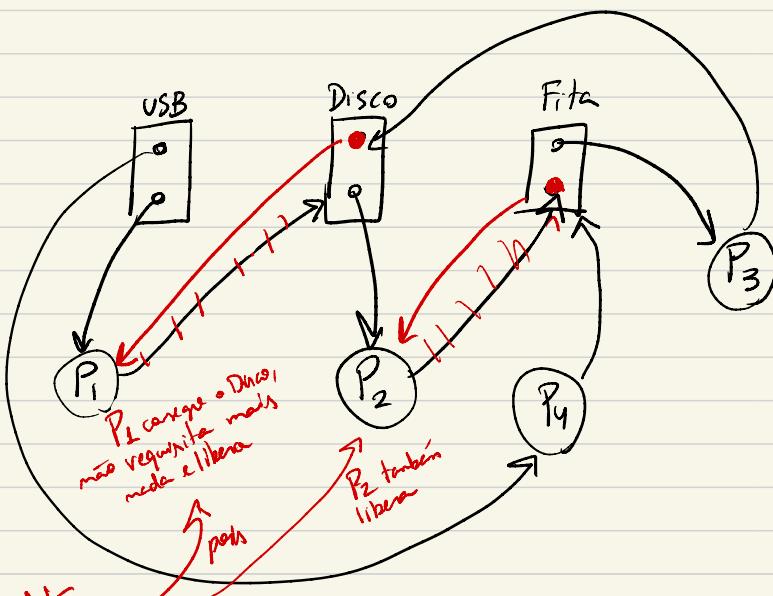
$TST(\text{int aux}) \{ \text{aux} = C; C = 1 \}$

5) Considere a situação em que 4 processos: P1, P2, P3 e P4 concorrem por recursos em um sistema que possui 2 unidades de USB, 2 unidades de disco e 2 unidades de fita. Os processos estão na seguinte situação:

- O processo P1 está de posse de uma USB e requisita uma unidade de disco.
- O processo P2 está de posse de uma unidade de disco e requisita uma unidade de fita.
- O processo P3 está de posse de uma unidade de fita e requisita a outra unidade de disco.
- O processo P4 está de posse de uma USB e requisita uma unidade de fita.

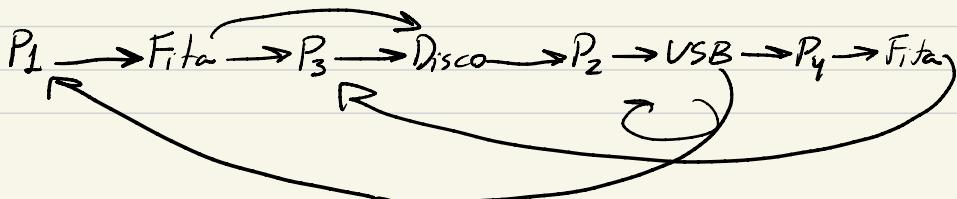
Esta situação leva a um deadlock? Por quê? Mostre o grafo de alocação de recursos!

Altera-se a situação se o processo P2 pudesse requisitar uma USB e o processo P1 pudesse requisitar uma unidade de fita? Por quê?



a) ~~Sx~~ Não, há deadlock pois, já atendendo exclusão mútua e não-preempção, todos os processos estão esperando por um recurso e aguardando entre si, formando uma apera circular (e põe-aqua).

b) Não, muda muda, ainda há um deadlock:



Questão 3. Um computador tem 6 fitas, com N processos competindo pelas mesmas. Cada processo necessita de 2 fitas. Para quais valores de N o sistema é livre de deadlocks?

Para $n < 5$ o sistema será livre de deadlocks, pois respectivamente 1 fita a mais se ocorrer a necessidade de um processo precisar de duas fitas

- 10) Um sistema de tempo real tem 4 eventos com períodos de 50, 100, 200, 250 msegms cada. Considere que cada evento demande 35, 20, 10 e x msegms de CPU respectivamente. Qual o maior valor de x para o sistema continue escalonável?

Obs: em um sistema de tempo real um evento ei deve ser completado antes que o mesmo evento ocorra novamente. Se isto não acontecer o sistema não será escalonável.

| | Tep. fxx. | Rm | |
|-------|-----------|-----|--|
| P_1 | 35 | 50 | $\left\{ \begin{array}{l} MML = 50, 100, 200, 250 \\ 25, 50, 100, 125 \\ 25, 25, 50, 125 \\ 25, 25, 25, 125 \end{array} \right.$ |
| P_2 | 20 | 200 | |
| P_3 | 20 | 200 | |
| P_4 | x | 250 | |

$$P_1 \frac{\frac{MMC}{P_{min}}}{P_{max}} = \frac{1000}{80} = 20$$

$$\text{tempo necessário} = 20 \cdot 35 = 700$$

$$P_2 \frac{1000}{100} = 10 \rightarrow 20 \cdot 10 = 200$$

$$P_3 \frac{1000}{200} = 5 \rightarrow 5 \cdot 10 = 50$$

$$P_4 = \frac{1000}{250} = 4 \rightarrow 4 \cdot x = 4x$$

$$1000 \geq 700 + 200 + 50 + 4x$$

$$50 \geq 4x$$

$$x \leq 12,5$$

$$\frac{1000}{250} \leq \frac{12,5}{x}$$

$$4x \leq 12,5 \Rightarrow x = 3,125$$

Questão 1. Considere o seguinte código:

```
n : integer;  
s, delay : semaforo; // binário
```

```
produtor {  
    loop {
```

*N S for igual a 0,
não entre no
nível critico e
nunca.
Se não produzir.*

```
        wait(s);  
        critico 1;  
        n = n + 1;  
        if (n = 1) then signal (delay);  
        signal(s);
```

```
    } }
```

*se algum processo P tiver n=0,
acorda P.
se n > 0 libera S*

```
} }  
/// execução principal ///
```

```
begin n = 0; s = 1; delay = 0; cobegin produtor, consumidor; coend end.
```

```
consumidor {  
    wait(delay);  
    loop {
```

```
        wait(s);  
        critico 2;
```

```
        n = n - 1;
```

```
        if (n = 0) then wait(delay);
```

```
        signal(s);
```

*if (n=0){signal(s);
wait(delay);}*

```
else signal(s)
```

*P → 0 L → L
C → 1 0 0*

?



Discuta a correção da solução apresentada.

*quando o produtor produz
1 e o consumidor já come-
re, o n=0, e ele fica
esperando o delay mas nunca
liberava a região critica
n' produzir mais.*