GABARITO - P2 - Sistemas Operacionais I
Professor: Leandro Marzulo
2013-1

Nome:

Instruções: Esta prova é composta de 3 questões totalizando 12 (doze) pontos, sendo a nota máxima 10 (dez). Responda as questões de forma sucinta e clara. O uso de lápis é permitido, no entanto, pedidos de revisão serão considerados apenas para questões respondidas a caneta. BOA PROVA!

1) (6,0) Considere três processos com as características descritas na tabela a seguir:

Processo	Tempo da Rajada de CPU 1	Tempo da Rajada de I/O	Tempo da Rajada de CPU 2	Prioridade	Instante de criação
P1	5	12	3	2	2
P2	8	5	5	1	3
P3	4	8	2	3	0

Considere que cada processo faz I/O em um dispositivo independente (todos os I/Os são paralelos) e que o tempo de troca de contexto é insignificante. Saiba que, para cada processo:

Tempo de Turnarround = Tempo de Execução no processador + Tempo de I/O + Tempo de Espera

Repare que o Tempo de Execução no Processador e o Tempo de I/O de cada processo é independente do mecanismo de escalonamento. Além disso, o tempo de turnarround de cada processo pode também ser definido como o tempo decorrido entre o instante de criação do processo e o instante do seu término.

Para cada um dos mecanismos de escalonamento a seguir, desenhe o diagrama de Gantt ilustrando o escalonamento dos processos, além de calcular seus respectivos tempos de turnaround e tempo médio de espera, segundo as políticas especificadas a seguir.

Vamos calcular, para cada processo, o "Tempo de Execução no processado" e o "Tempo de I/O", pois estes são independentes do mecanismo de escalonamento. O Tempo de Execução no processado" de um processo é a soma dos tempos de rajada de CPU do mesmo. Já o "Tempo de I/O" é soma dos tempos de raja de I/O do mesmo. Temos, portanto:

Processo	Tempo de Execução no processador	Tempo de I/O
P1	5+3 = 8	12
P2	8+5 = 13	5
P3	4+2 = 6	8

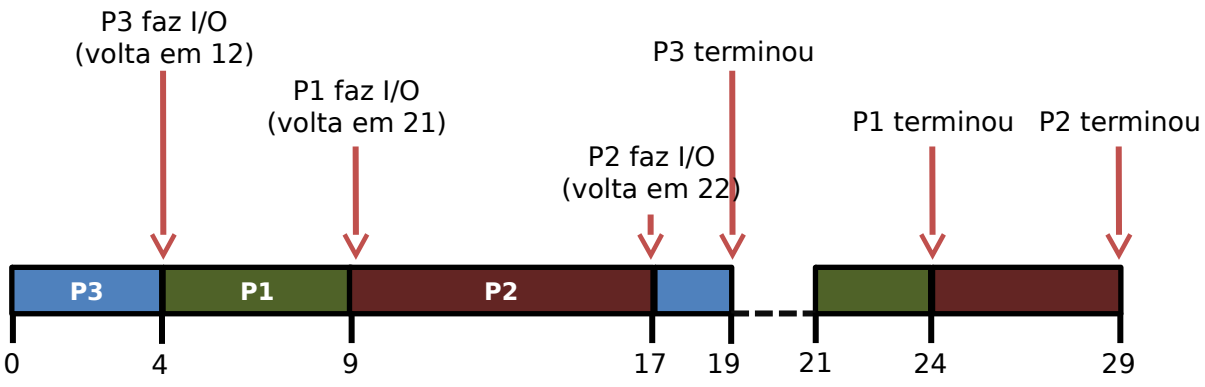
Além disso, sabemos que:

Tempo de turnaround = Instante do término - Instante de criação

e que, portanto:

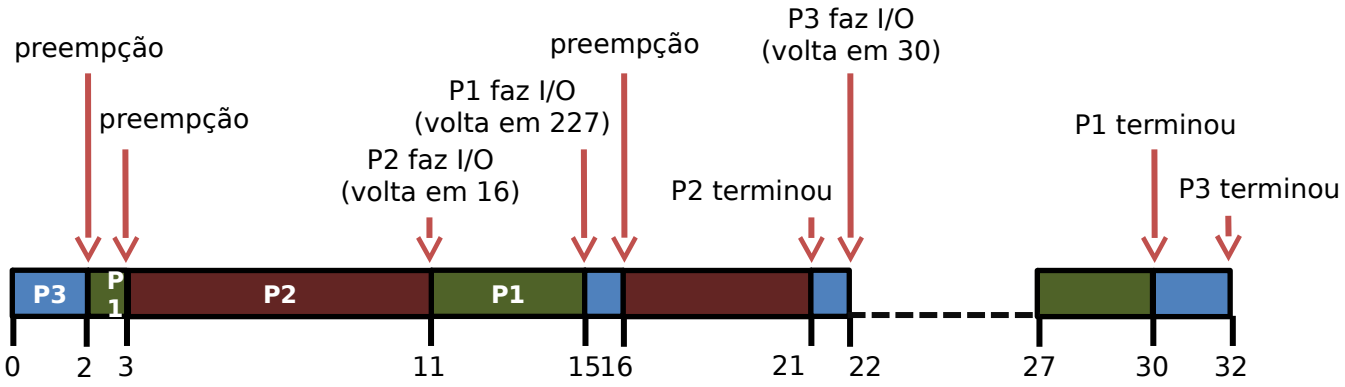
Tempo de Espera = Tempo de turnaround - Tempo de Execução no processador - Tempo de I/O

a) (2,0) FIFO



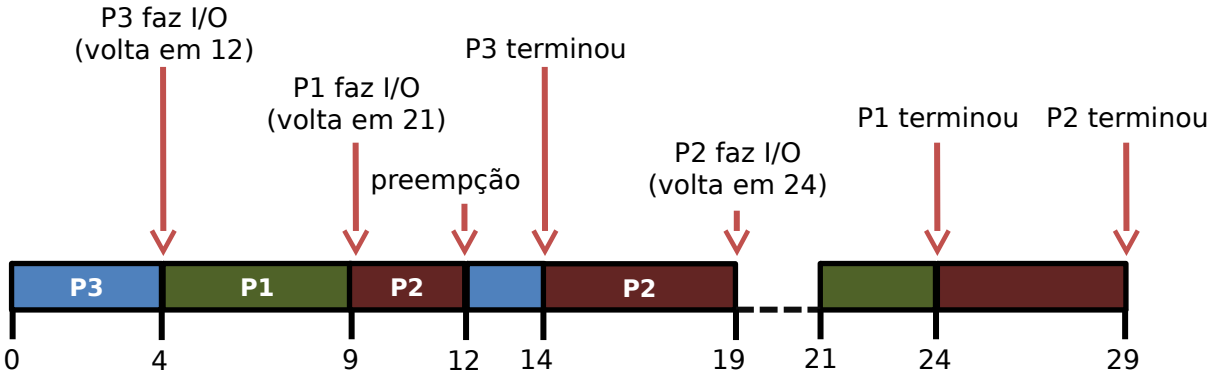
Processo	Tempo de Turnaround	Tempo de Espera
P1	24-2 = 22	22-12-8 = 2
P2	29-3 = 26	26-5-13 = 8
P3	19-0 = 19	19-8-6 = 5
Média	-	5

b) (2,0) Prioridade com preempção (número menor implica prioridade maior)



Processo	Tempo de Turnaround	Tempo de Espera
P1	30-2 = 28	28-12-8 = 8
P2	21-3 = 18	18-5-13 = 0
P3	32-0 = 32	32-8-6 = 18
Média	-	8,66

c) (2,0) Trabalho restante mais curto primeiro



Processo	Tempo de Turnaround	Tempo de Espera
P1	24-2 = 22	22-12-8 = 2
P2	29-3 = 26	26-5-13 = 8
P3	14-0 = 14	14-8-6 = 0
Média	-	3,33

2) (3,0) Considere um sistema com um total 8 instâncias de um determinado recurso. Temos 3 processos A, B e C que executam concorrentemente. A já alocou 2 instâncias e precisa de mais 3 para terminar a sua execução; B alocou 1 e precisa de mais 6; C alocou 4 e precisa de mais 1. Considerando o conceito de sequência de segurança e estado de segurança, responda:

a) (1,0) Mostre todas as sequências de segurança válidas (se houver alguma) e explique o que torna as sequência válidas.

A sequência de segurança válida é C,A,B. Ela é válida pois:

- Temos 7 recursos alocados e apenas 1 disponível;
- Com o recurso disponível só possível atender a necessidade de C (que, por este motivo, aparece então em primeiro na sequência);
- Quando C é atendido plenamente, ele libera 5 recursos;
- Com esses 5 recursos só é possível atender a necessidade de A, usando apenas 3 recuros (por este motivo, A aparece em segundo na sequência);
- Quando A é atendido plenamente ele libera 5 recursos, deixando 7 recursos disponíveis no sistema;
- Apenas com 6 recursos já é possível atender a necessidade de B plenamente (B aparece por último na sequência);
- Portanto, seguindo a sequência de segurança é possível atender a necessidade máxima de todos os processos sem deadlock.

b) (1,0) Se A fizesse a requisição de um recurso e fosse atendido, ainda teríamos alguma sequência de segurança válida? Em caso afirmativo, quais? Em caso negativo, o que aconteceria no sistema?

Não temos mais sequências válidas, pois com 0 recursos remanecentes não podemos atender plenamente a nenhum dos processos. O sistema entraria em deadlock.

c) (1,0) Considerando ainda o cenário original do enunciado, se B fizesse a requisição três recursos e fosse atendido, ainda teríamos alguma sequência de segurança válida? Em caso afirmativo, quais? Em caso negativo, o que aconteceria no sistema?

Anulada

3) (3,0) Suponha que tenhamos três regiões de memória, R1, R2 e R3, compartilhadas por três processos, A, B e C. O processo A deposita dois valores, um em R1 e o outro em R2, os quais são consumidos pelo processo B. O processo B, por sua vez, após ler os valores de R1 e de R2, coloca a soma deles em R3 para ser consumida pelo processo C. Finalmente, o processo C somente lê a soma de R3. Como os semáforos binários podem ser usados para garantir o correto funcionamento dos processos A, B e C, supondo que eles repitam o procedimento descrito acima indefinidamente? Porque não são necessários semáforos de contagem para resolver este problema?

Precisamos de quatro semáforos binários, dadojalido12 e dado disponivel12, para as regiões R1 e R2, e dadojalido3 e dadodisponivel3 para a região R3**. O semáforo dadodisponivel12 irá bloquear o processo B se o processo A ainda não tiver colocado novos valores em R1 e R2. Já o semáforo dadojalido12 irá bloquear o processo A caso o processo B ainda não tenha lido os últimos valores colocados por A em R1 e em R2. De modo similar, o semáforo dadodisponivel3 irá bloquear o processo C se o processo B ainda não tiver colocado um novo valor em R3, e o semáforo dadojalido3 irá bloquear o processo B se o processo C ainda não tiver lido o último valor colocado por B em R3. Como inicialmente os processos não estarão em execução, então não existirão valores válidos em R1, R2 e R3 e, com isso, os valores dos semáforos dadodisponivel12 e dadodisponivel3 serão ambos 0, e os valores dos semáforos dadojalido12 e dadojalido3 serão ambos 1. A seguir damos um esboço (em C) dos procedimentos ProcessoA, ProcessoB e ProcessoC que devem ser executados, respectivamente, pelos processos A, B e C. A função GerarValor gera um novo valor para ser colocado em R1 ou em R2 e a função UsarSoma usa a soma (de R1 e R2) lida de R3. Repare que os semáforos funcionam como os semáforos cheia e vazia do exemplo de produtor/consumidor visto em sala de aula. Eles podem ser binários pois não estamos lidando com um buffer, mas apenas com variáveis simples. Não precisamos do semáforo de acesso para cada processo pois com o uso desses semáforos binários só um processo por vez pode acessar as variáveis. No caso do produtor-consumidor com buffer mais de um produtor poderia acessar o buffer e deveria incrementar os índices de acesso; daí a necessidade de ter o semáforo de exclusão mútua além dos semáforos de cheia a vazia.**

```
void ProcessoA(void)
{
      int Valor1, Valor2;
      while (1)
      {
            Valor1 = GerarValor();
            Valor2 = GerarValor();
            P(dadojalido12);
            R1 = Valor1;
            R2 = Valor2;
            V(dadodisponivel12);
      }
}


void ProcessoB(void)
{
      int Soma;
```

```
	while (1)
	{
		P(dadodisponivel12);
		Soma = R1 + R2;
		V(dadojalido12);
		P(dadojalido3);
		R3 = Soma;
		V(dadodisponivel3);
	}
}

void ProcessoC(void)
{
	int Soma;
	while (1)
	{
		P(dadodisponivel3);
		Soma = R3;
		V(dadojalido3);
		UsarSoma(Soma);
	}
}
```