INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ

ENGENHARIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS
SEMESTRE 2019.2 – Prova 1ª Etapa
PROF. FERNANDO PARENTE GARCIA

ALUNO: Mardonio Hilbert Marques de França

6,0

1) (2,0 Pontos) Sobre conceitos básicos de sistemas operacionais, responda:
   a) Qual o objetivo de estruturar um sistema de computação em níveis de camadas?
   b) Um processo que efetua um cálculo matemático complexo (por exemplo, inversão de matrizes) é *CPU-bound* ou *I/O-bound*? Justifique.

1,5

2) (2,0 Pontos) Sobre processos e threads, responda:
   a) Defina contexto de software e contexto de hardware. Qual deles pode ser compartilhado pelos threads de um mesmo processo?
   b) Qual a principal desvantagem de se definir um *timeslice* muito pequeno para os processos? Justifique.

1,8

3) (2,0 Pontos) Suponha que os seguintes processos chegaram para execução nos tempos indicados. Cada processo rodará a quantidade de tempo listada na tabela.

Processo	Tempo de Chegada (ms)	Tempo de Execução (ms)	Prioridade
A	5	15	1
B	10	7	2
C	12	9	2
D	14	5	2
E	30	14	1

1,3

Qual o tempo médio de espera para estes processos quando são utilizados os algoritmos de escalonamento abaixo. Considere que o sistema operacional gasta 1 ms para realizar a troca de contexto.
   a) SJF (*Short Job First*).
   b) Escalonamento circular com prioridade estática e *timeslice* de 4 ms.

4) (2,0 Pontos) A figura abaixo mostra a situação instantânea de um sistema em que é utilizado o algoritmo do banqueiro para evitação de deadlocks. Se uma requisição do processo P_1 chegar para (0,4,2,0), a requisição poderá ser concedida imediatamente? Justifique.

	Alocação				Máximo				Disponível			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

2,0

5) (2,0 Pontos) Problema *da busca/inserção/remoção:* Três tipos de threads compartilham acesso a uma lista encadeada: busca, inserção e remoção. Threads de busca meramente examinam a lista; Assim podem executar concorrentemente entre si. Threads de inserção adicionam itens no final da lista; inserções devem ser mutuamente exclusivas para impedir duas inserções de ocorrerem simultaneamente. No entanto, uma inserção pode ocorrer em paralelo com qualquer quantidade de buscas. Finalmente, remoções excluem um nó em qualquer local da lista, e pode ocorrer somente uma remoção por vez. Além disso, durante uma remoção não pode ocorrer nenhuma busca e nenhuma inserção. Os algoritmos abaixo propõem uma solução. Descreva detalhadamente os problemas encontrados na implementação proposta, e em seguida proponha alterações para solucionar estes problemas.

SEMAPHORE MUTEX1 = 1; SEMAPHORE MUTEX2 = 1; INT BUSCA = ~~0~~ 1

THREAD BUSCA:
```
WHILE (TRUE) {
 DOWN(MUTEX2);
 IF (BUSCA = = 0) DOWN(MUTEX1);
 BUSCA++;
 UP(MUTEX2);
 buscar_na_lista();
 DOWN(MUTEX2);
 BUSCA--;
 IF (BUSCA = = 0) UP(MUTEX1);
 UP(MUTEX2); }
```

THREAD INSERÇÃO:
```
WHILE (TRUE) {
 DOWN(MUTEX1);
 inserir_na_lista();
 UP(MUTEX1);
 }
```

THREAD REMOÇÃO:
```
WHILE (TRUE) {
 DOWN(MUTEX1); busca=0;
 remover_na_lista();
 UP(MUTEX1); busca=1;
}
```

BOA PROVA!

01. a) Possui como objetivo aumentar a segurança, pois desse modo as aplicações não possuem acesso direto aos recursos do hardware, existindo camadas do S.O. que servem como intermediadoras. 0,5

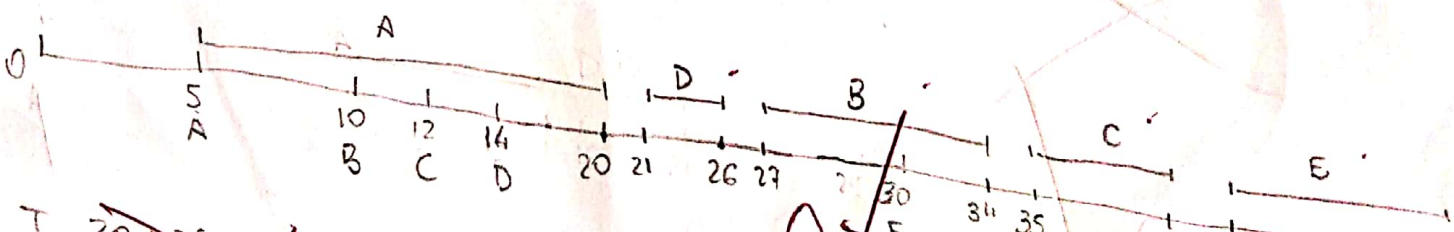
b) É CPU-bound, pois é uma tarefa que passa a maior parte do tempo em execução. Para ser I/O-bound deveria passar a maior parte do tempo bloqueada esperando uma ação do usuário. 1,0

02. a) Contexto de software são informações ~~do~~ e prioridade de execução. Contexto de hardware são informações sobre registradores gerais e específicos da cpu como PC e registrador de status. O contexto de software é compartilhado pelas threads de um mesmo processo, no entanto cada thread possui seu próprio contexto de hardware. 0,9

b) Ao definir um timeslice muito pequeno degrada-se a performance, pois a cpu passará a gastar muito tempo com trocas de contexto entre os processos. 0,9

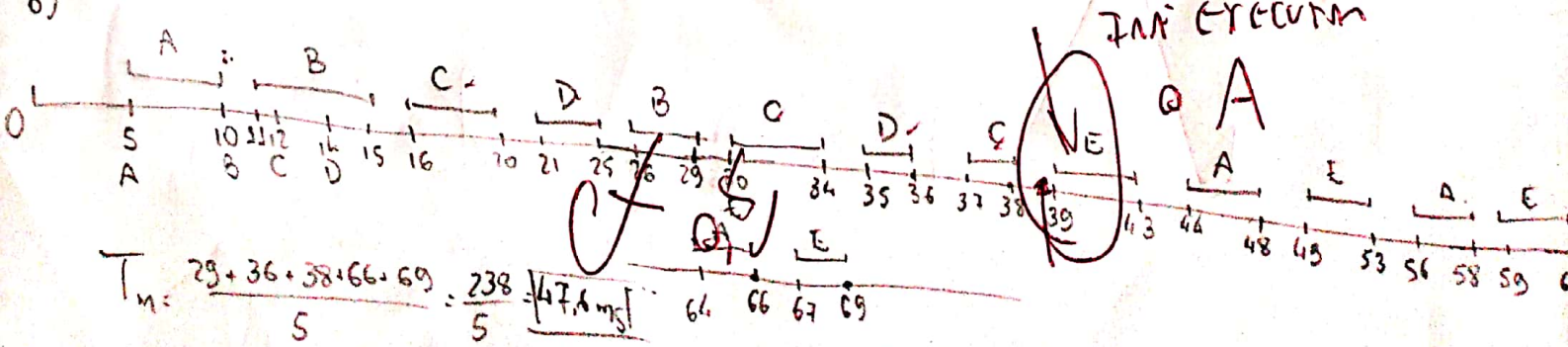
03.

a)



$T_m = \frac{20 + 26 + 34 + 44 + 59}{5} = \frac{183}{5} = 36,6\ ms$

b)



$T_m = \frac{29 + 36 + 58 + 66 + 69}{5} = \frac{238}{5} = 47,6\ ms$

Scanned by CamScanner

04. A = (1, 5, 2, 0) ; C =

P_0	0	0	1	2
P_1	1	0	0	0
P_2	1	3	5	4
P_3	0	6	3	2
P_4	0	0	1	4

;

P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

R.: Sim, pois

Se concedermos a requisição de P_1:

Temos um novo vetor de recursos disponíveis

$A = (1, 1, 0, 0)$. P_0 pode executar, com isso temos: $A = (1, 1, 1, 2)$

Com esse novo vetor de disponíveis P_2 executa. Temos: $A = (2, 4, 6, 6)$

Com isso P_3 pode executar, temos: $A = (2, 10, 9, 8)$, e com esses recursos podemos executar P_1 e P_4.

novo R após requisição de P_1 =

P_0	0	0	0	0
P_1	0	3	3	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

05.