

Aula 15-04 - Teste Avaliativo 1

Entrega 15 abr em 12:20

Pontos 35

Perguntas 10

Disponível 15 abr em 10:40 - 15 abr em 12:20 aproximadamente 2 horas

Limite de tempo 100 Minutos

Instruções

Prezado aluno,

bom dia. Responda ao que pede.

[]'s

Mark

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	59 minutos	35 de 35

⚠ As respostas corretas estão ocultas.

Pontuação deste teste: **35** de 35

Enviado 15 abr em 11:39

Esta tentativa levou 59 minutos.

Pergunta 1

2,5 / 2,5 pts

Em um sistema operacional, um processo pode, em um dado instante de tempo, estar em um de três estados: em execução, pronto ou bloqueado. Considere as afirmativas abaixo sobre as possíveis transições entre estes estados que um processo pode realizar.

- I. Do estado em execução para o estado bloqueado
- II. Do estado em execução para o estado pronto
- III. Do estado pronto para o estado em execução

IV. Do estado pronto para o estado bloqueado

V. Do estado bloqueado para o estado em execução

VI. Do estado bloqueado para o estado pronto

Quais são as afirmativas verdadeiras?

- ☐ Todas as afirmativas são verdadeiras.
- ☒ Somente as afirmativas I, II, III e VI são verdadeiras.
- ☐ Somente as afirmativas I, III, IV e VI são verdadeiras.
- ☐ Somente as afirmativas I, III, IV e V são verdadeiras.
- ☐ Somente as afirmativas I, II e são verdadeiras.

Pergunta 2

2,5 / 2,5 pts

Alguns dos objetivos dos algoritmos de escalonamento de processos são comuns a todos os tipos de sistemas operacionais. Outros, entretanto, variam de acordo com o tipo de sistema. Qual dos objetivos abaixo NÃO se aplica a algoritmos de escalonamento de processos?

- ☐ Manter a CPU ocupada o tempo todo.
- ☒ Manter os dispositivos de E/S ocupados o máximo de tempo possível.
- ☐ Atender às requisições dos usuários o mais rápido possível.
- ☐ Maximizar o número de tarefas processadas por unidade de tempo.

- ☐ Minimizar o tempo entre a submissão e o término de um job.

Pergunta 3**2,5 / 2,5 pts**

Starvation ocorre quando:

- ☒ Pelo menos um processo é continuamente postergado e não executa.
- ☐ Dois ou mais processos são forçados a acessar dados críticos alternando estritamente entre eles.
- ☐ O processo tenta mas não consegue acessar uma variável compartilhada
- ☐ Pelo menos um evento espera por outro evento que não vai ocorrer.
- ☐ A prioridade de um processo é ajustada de acordo com o tempo total de execução do mesmo.

Pergunta 4**4,5 / 4,5 pts**

Considere o seguinte código que implementa exclusão mútua entre dois processo i e j:

Processo Pi

while (1) {

while (turn != i) ; // entrada seção

crítica

```
                seção crítica
                turn = j;                // saída da seção
crítica
                código restante
            }
```

Processo Pj

```
                while (1) {
                    while (turn != j) ;    // entrada seção
crítica
                    seção crítica
                    turn = i;                // saída da
seção crítica
                    código restante
                }
```

De acordo com o código acima pode-se afirmar, exceto:

- ☐ A solução garante exclusão mútua.
- ☐ Os processos fazem espera ativa.
- ☐ Um processo bloqueia o outro mesmo não estando na seção crítica.
- ☐ Existe alternância estrita.
- ☒ A solução garante progresso.

Pergunta 5

4,5 / 4,5 pts

O escalonamento de CPU trata do problema de decidir qual dos processos na fila de prontos deve ser entregue à CPU. Considere que o algoritmo de escalonamento Round-Robin esteja sendo utilizado e que o conjunto de processos abaixo chegue no momento 0, com a extensão do tempo de burst de CPU indicada em milissegundos. Dado: P1 é o primeiro processo na fila de prontos, P2 é o segundo e P3 é o terceiro.

Processo	Tempo de burst
P1	10
P2	3
P3	4

Se for utilizado um quantum de 4 milissegundos, o tempo de RETORNO de P1 será de:

☐ 13

☐ 21

☐ 24

☐ 18

☒ 17

Pergunta 6

2,5 / 2,5 pts

Nos conceitos de ciência da computação, um processo é um módulo executável que pode conter *threads*. Um conceito importante sobre

threads que estão contidas no mesmo processo é que:

- ☒ podem compartilhar a memória do processo.
- ☐ controlam o acesso da memória dinâmica (heap) dos processos externos.
- ☐ evitam que processos concorrentes acessem recursos do processo principal.
- ☐ possuem proteção contra problemas comuns de processos, como vazamento e acessos inválidos da memória.
- ☐ acessam a memória estática ao contrário de processos convencionais.

Pergunta 7

4,5 / 4,5 pts

A concorrência entre processos ocorrem em diversas situações no âmbito dos sistemas operacionais e outros tipos de processos. Alguns destes problemas foram estudados e se tornaram clássicos pela sua ocorrência e aplicabilidade. Um destes é o problema do produtor consumidor, onde temos um ou mais processos “produzindo” uma informação e um ou mais processos “consumindo” a informação. Neste problema o mais importante é controlar o acesso dos processos envolvidos à porção de memória compartilhada de forma a não permitir que dois processos não “consumam” a mesma informação ou uma informação não possa ser consumida enquanto ela ainda está sendo produzida. A implementação mais segura para este cenário é com o uso de dois semáforos contadores, geralmente chamados de: Cheio e Vazio. O semáforo Cheio controla o quão cheio está o buffer de memória e o Vazio controla o tanto que o mesmo buffer está vazio e admite mais itens serem produzidos.

Considere o código abaixo que implementa a solução para o

problema do produtor/consumidor:

```
void produtor ()
```

```
{
```

```
    int item;
```

```
    while (TRUE) {
```

```
        item = produzir();
```

```
        1. _____
```

```
        2. _____
```

```
        insere(item);
```

```
        3. _____
```

```
        4. _____
```

```
    }
```

```
}
```

```
void consumidor()
```

```
{
```

```
    int item;
```

```
    while(TRUE) {
```

```
        5. _____
```

```
        6. _____
```

```
        item = remover();
```

```
        7. _____
```

```
        8. _____
```

```
        consumir(item);
```

```
    }
```

```
}
```

Assinale a alternativa que preenche corretamente as lacunas.



1. wait(&cheio); 2. wait(&mutex); 3. signal(&mutex); 4. signal(&vazio);
5. wait(&vazio); 6. wait(&mutex); 7. signal(&mutex); 8. signal(&cheio);



1. wait(&vazio); 2. wait(&mutex); 3. signal(&mutex); 4. signal(&cheio);
5. wait(&cheio); 6. wait(&mutex); 7. signal(&mutex); 8. signal(&vazio);



1. wait(&mutex); 2. wait(&vazio); 3. signal(&cheio); 4. signal(&mutex);
5. wait(&mutex); 6. wait(&cheio); 7. signal(&vazio); 8. signal(&mutex);



1. wait(&mutex); 2. wait(&cheio); 3. signal(&vazio); 4. signal(&mutex);
5. wait(&mutex); 6. wait(&vazio); 7. signal(&cheio); 8. signal(&mutex);

Pergunta 8

2,5 / 2,5 pts

Um processo é apenas uma instância de um programa em execução, incluindo os valores atuais do contador do programa, registradores e variáveis. Além destas informações, todo processo possui um PID (número de identificação do processo), uma prioridade (que exerce influência no escalonamento), entre outras informações, como a credencial responsável pela execução do processo (UID – User ID).

Ao iniciar um processo, qual o nome da estrutura criada para armazenar estas informações referentes ao processo e que são utilizadas pelo sistema operacional para gerenciar o respectivo processo?



Ponteiro para o processo pai, conhecido como Parent PID.



Descritor de processo, conhecido como PCB (Process Control Block).



Contador de programa, que também determina qual instrução o processo deve executar em seguida.



Identificador do processo, conhecido como PID (Process Identifier).

Pergunta 9**4,5 / 4,5 pts**

Considerando um escalonamento SJF não-preemptivo:

Processo	Tempo chegada	Tempo de burst
P1	0,0	7
P2	2,0	4
P3	4,0	1
P4	5,0	4

Qual o tempo médio de retorno dados os processos acima?

☐ 6☒ 8☐ 10☐ 7☐ 9**Pergunta 10****4,5 / 4,5 pts**

Considerando um escalonamento Round-Robin com quantum = 20 e overhead = 5:

Processo	Tempo de burst
P_1	53

P_2

17

 P_3

68

 P_4

24

Qual a taxa de utilização da CPU, aproximadamente, dados os processos acima?

☐ 82☐ 84☐ 78☐ 76☒ 80

Pontuação do teste: **35** de 35