UNIVERSIDADE FEDERAL DO CEARÁ
Campus de Quixadá
Prof. Thiago Werlley Bandeira da Silva
QXD0013- Sistemas Operacionais

Lista
2022.2

Nome: _____ Matrícula: _____

1. Marque verdadeiro (V) ou falso (F) nas questões abaixo.

   ( ) Sistema operacional é o primeiro software a ser carregado pelo computador e desempenha várias funções importantes.

   ( ) Sistema operacional é o software básico do computador, sendo responsável por gerenciar o hardware e os programas, exceto a interação entre os programas e o hardware.

   ( ) A comunicação entre processos é uma forma de evitar que processos concorrentes acessem o mesmo endereço de memória ao mesmo tempo.

   ( ) O kernel é um conjunto de rotinas (Procedimentos) que oferecem serviços aos usuários do sistema e suas aplicações.

   ( ) O sistema que permite a execução de vários programas simultaneamente é conhecido como multitarefa.

   ( ) O sistema operacional monotarefa não suporta mais de um programa rodando.

   ( ) A ideia básica do escalonamento de prioridade: a cada processo é designada duas prioridades, e o processo executável com a prioridade mais baixa é autorizado a executar primeiro.

   ( ) Uma versão preemptiva da tarefa mais curta primeiro (shortest job first) é o tempo restante mais curto em seguida (shortest remaining time next).

   ( ) Um algoritmo de escalonamento preemptivo escolhe um processo para ser executado e então o deixa ser executado até que ele seja bloqueado (seja em E/S ou esperando por outro processo), ou libera voluntariamente a CPU.

   ( ) Um algoritmo de escalonamento não preemptivo escolhe um processo e o deixa executar por no máximo um certo tempo fixado. Se ele ainda estiver executando ao fim do intervalo de tempo, ele é suspenso e o escalonador escolhe outro processo para executar (se algum estiver disponível).

2. Marque a alternativa correta com respeito ao conceito de sistemas de compartilhamento de tempo e de multiprogramação.

   a) Em um sistema de multiprogramação, vários usuários podem acessar e executar cálculos em um sistema de computação simultaneamente usando seus próprios terminais.

   b) Todos os sistemas de compartilhamento de tempo são sistemas de multiprogramação, mas nem todos os sistemas de multiprogramação são sistemas de compartilhamento de tempo.

   c) Nem todos os sistemas de compartilhamento de tempo são sistemas de multiprogramação, mas todos os sistemas de multiprogramação são sistemas de compartilhamento de tempo.

   d) Os sistemas de compartilhamentos de tempo permitem ao usuário executar vários programas simultaneamente usando o mesmo terminal.

3. Marque a alternativa correta com respeito ao algoritmo de escalonamento Round Robin.

   a) Com respeito a seu funcionamento, cada processo recebe um intervalo de tempo, chamado quantum, durante o qual ele pode executar. Se o processo ainda estiver executando ao final do quantum, o sistema operacional interrompe a sua execução e passa a UCP outro processo.

   b) Um dos mais recentes, complexos, justos, e menos utilizados algoritmo de escalonamento é o Round Robin.

   c) Round Robin é complexo de implementar. O escalonador tem que manter os processos que desejam executar na memória secundária.

Nota: _____

d) Com respeito a seu funcionamento, se um processo é liberado ou iniciado antes do início do quantum, a troca da UCP para outro processo é obviamente feita assim que o processo é liberado ou iniciado.

4. Como podemos realizar a exclusão mútua?

a) Existe somente uma proposta para realizar a exclusão mútua, de maneira que enquanto um processo está ocupado atualizando a memória compartilhada em sua região crítica, nenhum outro entrará na sua região crítica para causar problemas.

b) Existem várias propostas para realizar a exclusão mútua, de maneira que enquanto um processo está ocupado atualizando a memória cache em sua região de memória, nenhum outro entrará na sua região para causar problemas.

c) Existe somente uma proposta para realizar a exclusão mútua, de maneira que enquanto um processo está ocupado atualizando a memória compartilhada em sua região crítica, nenhum outro entrará na sua região crítica para causar problemas. Para evitar exclusão mútua não pode-se utilizar: desabilitando interrupções, variáveis do tipo trava, chaveamento obrigatório, solução de Peterson, a instrução TSL, entre outros.

d) Existem várias propostas para realizar a exclusão mútua, de maneira que enquanto um processo está ocupado atualizando a memória compartilhada em sua região crítica, nenhum outro entrará na sua região crítica para causar problemas. Para obtenção de exclusão mútua pode-se utilizar: desabilitando interrupções, variáveis do tipo trava, chaveamento obrigatório, solução de Peterson, a instrução TSL, entre outros.

5. Marque verdadeiro (V) ou falso (F) nas questões abaixo quanto a definição de processo de thread?

( ) Um processo é uma unidade básica de execução na CPU. Uma thread é equivalente a uma tarefa, um único processo é uma tarefa executando dentro da thread.

( ) Um processo é uma unidade básica de execução na CPU. Uma thread é equivalente a várias tarefa, em que vários processos são executados dentro de uma thread.

( ) Um thread pode ser definido como uma subrotina de um programa que pode ser executada de forma assíncrona, ou seja, executada paralelamente ao programa chamador.

( ) Uma thread é um processo em execução na CPU, é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrencialmente. Um processo é a uma thread, ou seja, é uma abstração de espaço de memória.

( ) Um processo pode ser definido como o ambiente onde um programa é executado. Este ambiente, além das informações sobre a execução, possui também o quanto de recursos do sistema cada programa pode utilizar, como o espaço de endereçamento, tempo de processador e área em disco.

6. O que é uma região crítica?

a) Seção do programa onde são efetuados acessos (para leitura e escrita) a recursos partilhados por dois ou mais processos. É necessário assegurar que dois ou mais processos não se encontrem simultaneamente na região crítica.

b) Seção da memória onde são efetuados acessos (para leitura e escrita) a recursos partilhados por dois ou mais ponteiros.

c) Seção do programa onde são efetuados acessos (para leitura e escrita) a recursos partilhados por um único ponteiro.

d) Seção da memória onde são efetuados acessos (para leitura e escrita) a recursos partilhados por um único processo. É necessário assegurar que processos se encontrem simultaneamente na região crítica para processarem em conjunto.

7. Marque verdadeiro (V) ou falso (F) nas questões abaixo quanto a definição de deadlock e starvation.

( ) Deadlock é um problema que está presente em todos os sistemas operacionais atuais, que leva ao travamento de processos.

( ) A situação de starvation está presente quando o sistema operacional provê prioridades a processos, que não atualizados fazem com que os processos de menor prioridade nunca sejam executados causando assim, deficiência em servidores de impressão e etc.

( ) Starvation é um problema que está presente em todos os sistemas operacionais atuais, que leva ao travamento de processos.

( ) A situação de deadlock está presente quando o sistema operacional provê prioridades a processos, que não atualizados fazem com que os processos de menor prioridade nunca sejam executados causando assim, deficiência em servidores de impressão e etc.

8. Cinco tarefas em lote, A até E, chegam a um centro de computadores quase ao mesmo tempo. Elas têm tempos de execução estimados de 10, 6, 2, 4 e 8 minutos. Suas prioridades (externamente determinadas) são 3, 5, 2, 1 e 4, respectivamente, sendo 5 a mais alta. Para cada um dos algoritmos de escalonamento a seguir, determine o tempo de retorno médio do processo. Ignore a sobrecarga de chaveamento de processo.

Processo	Tempo de Execução	Prioridade
A	10	3
B	6	5
C	2	2
D	4	1
E	8	4

   a) Circular
   b) Escalonamento por prioridade.
   c) Primeiro a chegar, primeiro a ser servido (siga a ordem 10, 6, 2, 4, 8).
   d) Tarefa mais curta primeiro.

   Para (a), presuma que o sistema é multiprogramado e que cada tarefa recebe sua porção justa de tempo na CPU. Para (b) até (d), presuma que apenas uma tarefa de cada vez é executada, até terminar. Todas as tarefas são completamente limitadas pela CPU.

9. Quatro processos *A*, *B*, *C* e *D* chegam na ordem apresentada na tabela abaixo e, aproximadamente, ao mesmo tempo com os seguintes tempos de execução e prioridades (menor valor significa maior prioridade):

Processo	Tempo de Execução	Prioridade
A	3	3
B	7	4
C	5	1
D	4	2

   Determine o tempo de resposta de cada processo e o tempo de espera médio para os algoritmos *First Come First Served* (FCFS) e *Shortest Job First* (SJF).

10. Considere o seguintes conjunto de processos, com o tamanho do tempo de burst de CPU dado em minutos:
    Presume-se que os processos tenham chegado na ordem A, B, C, D, E, todos no momento 0.

    a) Desenhe quatro gráficos de Gantt que ilustrem a execução desses processos usando os algoritmos de escalonamentos a seguir: FCFS, SJF, por prioridades sem preempção (um número de prioridade menor implica uma prioridade mais alta) e Circular (quantum = 1).
    b) Qual é o tempo de resposta de cada processo para cada um dos algoritmos de escalonamento da parte a?

Processo	Tempo de Execução	Prioridade
A	10	3
B	1	1
C	2	3
D	1	4
E	5	2

c) Qual é o tempo de espera de cada processo para cada um desses algoritmos de escalonamento?

d) Qual dos algoritmos resulta no menor tempo médio de espera (para todos os processos)?

11. Na solução para o problema do jantar dos filósofos (Código abaixo), por que a variável de estado está configurada para HUNGRY na rotina take_forks?

```
#define N               5               /* numero de filosofos */
#define LEFT            (i+N–1)%N       /* numero do vizinho a esquerda de i */
#define RIGHT           (i+1)%N         /* numero do vizinho a direita de i */
#define THINKING        0               /* o filosofo esta pensando */
#define HUNGRY          1               /* o filosofo esta tentando pegar garfos */
#define EATING          2               /* o filosofo esta comendo */

typedef int semaphore;                  /* semaforos sao um tipo especial de int */
int state[N];                           /* arranjo para controlar o estado de cada um */
semaphore mutex = 1;                    /* exclusao mutua para as regioes criticas */
semaphore s[N];                         /* um semaforo por filosofo */

void philosopher(int i)                 /* i: o numero do filosofo, de 0 a N–1 */
{
    while (TRUE) {                      /* repete para sempre */
        think( );                       /* o filosofo esta pensando */
        take_forks(i);                  /* pega dois garfos ou bloqueia */
        eat( );                         /* hummm, espaguete! */
        put_forks(i);                   /* devolve os dois garfos a mesa */
    }
}

void take_forks(int i)                  /* i: o numero do filosofo, de 0 a N–1 */
{
    down(&mutex);                       /* entra na regiao critica */
    state[i] = HUNGRY;                  /* registra que o filosofo esta faminto */
    test(i);                            /* tenta pegar dois garfos */
    up(&mutex);                         /* sai da regiao critica */
    down(&s[i]);                        /* bloqueia se os garfos nao foram pegos */
}

void put_forks(i)                       /* i: o numero do filosofo, de 0 a N–1 */
{
    down(&mutex);                       /* entra na regiao critica */
    state[i] = THINKING;                /* o filosofo acabou de comer */
    test(LEFT);                         /* ve se o vizinho da esquerda pode comer agora */
    test(RIGHT);                        /* ve se o vizinho da direita pode comer agora */
    up(&mutex);                         /* sai da regiao critica */
}

void test(i)/* i: o numero do filosofo, de 0 a N–1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        up(&s[i]);
    }
}
```

a) Se um filósofo bloquear, os vizinhos não poderão ver mais tarde que ele está com fome verificando seu estado, em teste, para que eles possam dormir até os garfos estarem disponíveis.

b) Se dois filósofos bloquearem, os vizinhos não poderão ver mais tarde que eles estão com fome verificando seu estado, em teste, para que eles possam ser despertados quando os garfos estiverem disponíveis.

c) Se um filósofo bloquear, os vizinhos poderão ver mais tarde que ele está com fome verificando seu estado, em teste, para que ele possa ser despertado quando os garfos estiverem disponíveis.

d) Se dois filósofos bloquearem, um vizinho poderá ver mais tarde que eles estão com fome verificando seu estado, em teste, para que eles possam dormir até os garfos estarem disponíveis.