Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus de Pato Branco
Curso de Engenharia de Computação
Professor(a) Dalcimar Casanova, Dr

Avaliação – 1ª Parcial
Disciplina: SO25CP – Turma: 7CP
27 de setembro de 2017

Acadêmico(a): _____

INSTRUÇÕES:

- Prova individual.
- A interpretação das questões faz parte da avaliação;
- Não é permitido o uso de material auxiliar de qualquer gênero;
- As respostas devem ser preenchidas a caneta, as respostas preenchidas a lápis não terão direito à revisão;
- As respostas poderão estar fora de ordem, desde que estejam numeradas;
- Valor da avaliação 10,0.
- Dentre as questões escolha:
- 3 questões com peso 1,5
- 3 questões com peso 0,5
- 4 questões com peso 1,0

Nota:

1. ( ) O que é multiprogramação? (V ou F) (Tanenbaum 1.3)
   a. ( ) Em um sistema de multiprogramação a CPU fica se alternando entre a execução de vários processos, cada um por dezenas ou centenas de milisegundos
   b. ( ) Em um sistema de multiprogramação temos frequentemente a situação onde vários processos estão prontos para serem executados
   c. ( ) É uma forma de evitar que processos que têm alta prioridade fiquem sendo executados indefinidamente
   d. ( ) Também conhecido como pseudo-paralelismo, a multiprogramação indica a qual processador o processo deve executar
   e. ( ) É uma forma de comunicação e sincronização de processos

2. ( ) Um arquivo, cujo proprietário tem UID=2 e GID=2, tem o modo acesso 760. Quais usuários abaixo poderão escrever no arquivo? (V ou F) (Tanenbaum 1.10)
   a. ( ) UID=2 e GID=3
   b. ( ) UID=3 e GID=2
   c. ( ) UID=4 e GID=1
   d. ( ) UID=5 e GID=2

3. ( ) Os pipes são um recurso fundamental? Alguma funcionalidade importante seria perdida se eles não estivessem disponíveis? (Assinale a alternativa correta) (Tanenbaum 1.16)
   a. ( ) Sim, sem eles não haveria nenhuma forma de comunicação entre processos
   b. ( ) Sim, pois sem os pipes processos não teriam como se comunicar na forma de pilha, ou seja, toda aplicação que utiliza-se um buffer-sequencial seria impossível
   c. ( ) Sim, sem named pipes a comunicação dentro do S.O. seria possível apenas entre processos descendentes diretos (pai-filho)
   d. ( ) Não, pipes são substituíveis em sua totalidade por outros mecanismos de comunicação
   e. ( ) Sim, pois é uma conexão entre dois processos, de modo que a saída padrão de um processo se torna a entrada padrão do outro processo. Sem os pipes esse redirecionamento de saídas e entradas seria mais complexo

4. ( ) Examine a lista de chamadas de sistema abaixo. Qual delas você acha que provavelmente será executada mais rapidamente? Explique sua resposta (assinale a alternativa correta) (Tanenbaum 1.20)
   a. ( ) fork: A criação de um processo é priorizada dentro de um sistema operacional, logo a chamada fork é a mais rápida de todas.
   b. ( ) getpid: É um chamada não bloqueante e não preempitiva, logo possui retorno imediato o que caracteriza uma chamada de sistema rápida
   c. ( ) read: A chamada read é não-bloqueante em todos os casos, e como a leitura é direto da memória ela é a chamada mais veloz do sistema
   d. ( ) wait: Um vez que o processo está rodando, chamar wait e bloquear é uma tarefa de preempção super veloz
   e. ( ) execv: Essa chamada não possui retorno, logo é muito rápida para ser executada

5. ( ) Cinco tarefas estão esperando para serem executadas. Seus tempos de execução esperados são 3, 8, 2, 5 e 10. Em que ordem elas devem ser executadas para minimizar o tempo de resposta médio? (assinale a alternativa correta) (Tanenbaum 2.27)
   a. ( ) 2, 3, 5, 8, 10
   b. ( ) 10, 8, 5, 3, 2
   c. ( ) 5, 3, 2, 10, 8
   d. ( ) 8, 3, 10, 2, 5
   e. ( ) Nenhuma das alternativas

6. ( ) O algoritmo de envelhecimento com alpha=1/2 está sendo eusado para prever tempos de execução. As quatro execuções anteriores, da mais antiga para a mais recente, foram de 30, 20, 40 e 40ms. Qual é a previsão do próximo tempo? (Tanembaum 2.30)

7. ( ) Na solução do problema da janta dos filósofos (abaixo), por que a variável de estado é configurada como HUNGRY na função take_forks? (V ou F) (Tanenbaum 2.21)

```
#define N 5 /* number of philosophers */
#define LEFT (i+N−1)%N /* number of i's left neighbor */
#define RIGHT (i+1)%N /* number of i's right neighbor */
#define THINKING 0 /* philosopher is thinking */
#define HUNGRY 1 /* philosopher is trying to get forks */
#define EATING 2 /* philosopher is eating */
typedef int semaphore; /* semaphores are a special kind of int */
int state[N]; /* array to keep track of everyone's state */
semaphore mutex = 1; /* mutual exclusion for critical regions */
semaphore s[N]; /* one semaphore per philosopher */

void philosopher(int i) /* i: philosopher number, from 0 to N−1 */
{
    while (TRUE) { /* repeat forever */
        think( ); /* philosopher is thinking */
        take_forks(i); /* acquire two forks or block */
        eat( ); /* yum-yum, spaghetti */
        put_forks(i); /* put both forks back on table */
    }
}
```

```
void take_forks(int i) /* i: philosopher number, from 0 to N−1 */
{
    down(&mutex); /* enter critical region */
    state[i] = HUNGRY; /* record fact that philosopher i is hungry */
    test(i); /* try to acquire 2 forks */
    up(&mutex); /* exit critical region */
    down(&s[i]); /* block if forks were not acquired */
}

void put_forks(i) /* i: philosopher number, from 0 to N−1 */
{
    down(&mutex); /* enter critical region */
    state[i] = THINKING; /* philosopher has finished eating */
    test(LEFT); /* see if left neighbor can now eat */
    test(RIGHT); /* see if right neighbor can now eat */
    up(&mutex); /* exit critical region */
}

void test(i) /* i: philosopher number, from 0 to N−1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        up(&s[i]);
    }
}
```

   a. ( ) É apenas uma variável didática, não faz diferença no código, pois uma vez que o processo entrou em take_forks (região crítica), ele irá comer e liberar o hungry em seguida
   b. ( ) É utilizada para controlar a região crítica
   c. ( ) Essa variável é o semáforo que controla a região crítica
   d. ( ) É utilizada, essencialmente, numa espécie de sinalizador, pois quando um filósofo termina de comer ele verifica se algum dos filósofos da mesa estão com fome
   e. ( ) É utilizada, essencialmente, numa espécie de sinalizador, pois quando um filósofo termina de comer ele verifica se algum dos filósofos vizinhos a ele estão com fome
   f. ( ) É utilizada para bloquear o acesso a variável semáforo up(&s[i]), que é efetivamente quem libera um filósofo para comer

8. ( ) Cinco tarefas de lote, de A a E, chegam em um centro de computação quase ao mesmo tempo. Elas têm tempos de execução estimados de 6, 10, 4, 2 e 8 segundos. Suas prioridades (determinadas externamente) são 1, 3, 2, 1 e 2, respectivamente, sendo 3 a prioridades mais alta. Para cada um dos algoritmos de escalonamento a seguir, determine o tempo de retorno médio dos processos. Ignore a sobrecarga da comutação de processo (Tanenbaum 2.28).
   a. Round-robin
   b. Escalonamento pro prioridade
   c. Primeiro a chegar, primeiro a ser servido
   d. Tarefa mais rápida primeiro

9. ( ) Desenhe um esquema para explicar a transição de estados dentro de um sistema operacional simples. Adicionando dois novos estados: Novo e Terminado. (Tanenbaum 2.5)

10. ( ) Quais as diferenças entre um processo e uma thread? (V ou F) (Tanenbaum 2.7)
    a. ( ) Threads possuem espaço de endereçamento próprio, logo as variáveis precisam ser explicitamente compartilhadas. Processos podem compartilhar seu espaço de endereçamento usando memória compartilhada
    b. ( ) Processos finalizados serão removidos do escalonamento pelo sistema operacional, já as threads finalizadas com continuam existindo até o processo terminar
    c. ( ) A troca de contexto de um processo é total e a troca de contexto de uma thread é parcial
    d. ( ) A comunicação entre processos é feita no modo com auxílio de recursos extras do kernel e entre threads essa comunicação é feita diretamente no espaço de endereçamento do processo a qual pertencem
    e. ( ) Threads tem seu próprio PCB