UNIVERSIDADE DE SÃO PAULO - ICMC
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO
SSC-0140 – SISTEMAS OPERACIONAIS I
Profa: Kalinka Regina Lucas Jaquie Castelo Branco

Sistemas Operacionais I

Lista 01 - Listão de Exercícios – Processos

(os exercícios aqui expostos são de autoria de diversos profissionais, do livro do Tanenbaum, poscomp, entre outros)

1) Em sua opinião, qual foi a principal evolução de hardware e software que possibilitou o desenvolvimento dos Sistemas Operacionais que existem atualmente? Justifique sua resposta.

2) Descreva quais são as principais tarefas de um Sistema Operacional.

3) Defina e diferencie: programa, processo e *thread.*

4) No que diz respeito à implementação, qual a diferença entre *threads* de usuário (*user-level threads*) e *threads* de *kernel* (*kernel-level threads*)?

5) O custo (e conseqüentemente o tempo) de criação de uma *thread* (seja ela uma *thread* de usuário ou uma *thread* de *kernel*) é menor que o custo de criação de um processo. Por quê?

6) Um processo é caracterizado por ser um "programa em execução". Durante sua existência no sistema, um processo pode assumir basicamente três estados. Cite quais são esses estados e descreve os eventos que fazem com que um processo mude de um estado para o outro.

7) Qual a diferença entre processos *CPU-bound* e *I/O-bound*? Quais são os problemas que podem ocorrer caso o sistema tenha muitos processos *CPU-bound* ou muitos processos *I/O-bound*?

8) O que são chamadas de sistema e como elas podem ser implementadas? Simulando uma chamada de sistema, descreva o que o Sistema Operacional deve fazer quando uma chamada ocorre. (**dicas**: http://br.kernelnewbies.org/docs/syscall_artigo.html; http://www.tldp.org/LDP/lki/lki-2.html#ss2.11)

9) [POSCOMP] Com respeito às vantagens da arquitetura de micro-núcleo (*microkernel*) para sistemas operacionais em relação à arquitetura de núcleo monolítico, quais das seguintes afirmações são verdadeiras?
I. A arquitetura de micro-núcleo facilita a depuração do S.O.
II. A arquitetura de micro-núcleo permite um número menor de mudanças de contexto.
III. A arquitetura de micro-núcleo facilita a reconfiguração de serviços do S.O., pois a maioria deles reside em espaço de usuário.

(a) apenas I; (b) II e III; (c) I e III; (d) I e II; (e) todas são verdadeiras.

10) Cite pelo menos duas vantagens do uso de sistemas operacionais baseados na estrutura em Camadas.

11) Explique por que a tarefa do escalonador de processos é importante para o desempenho da CPU.

12) Descreva a execução dos seguintes algoritmos de escalonamento: *Round-robin*, Prioridades e Múltiplas Filas.

13) A maioria dos escalonadores *Round-robin* usa um *quantum* de tamanho fixo. O que pode acontecer se o *quantum* for muito pequeno? E se ele for muito grande?

14) Diferencie *Race Condition* de Exclusão Mútua.

15) Uma boa solução de Exclusão Mútua deve satisfazer quais condições? Por que?

16) Cinco processos (A, B, C, D, E) estão no estado de pronto para serem executados pela CPU. O tempo estimado de execução de cada processo é: 10, 6, 2, 4 e 8 segundos, respectivamente. Simule a execução dos seguintes algoritmos de escalonamento com esses processos e determine o *turnaround time* (tempo de retorno) de cada processo em cada algoritmo. Considere que o tempo de chaveamento entre os processos é de um segundo. Considere também o tempo para colocar o processo A na CPU.

- *Round-Robin* (*quantum*=3 segundos);
- SJF;
- FIFO.

17) Cinco processos estão prontos para serem executados. Os tempos de execução são: 9, 6, 3, 5 e X minutos. Qual deve ser a ordem de execução dos processos para que o *turnaround time* (tempo de retorno) de cada processo seja minimizado?

18) Quais das instruções a seguir devem ser realizadas somente pelo *Kernel* do Sistema Operacional? Justifique sua resposta.

a) Desabilitar interrupções;
b) Ler a hora do relógio;
c) Modificar hora do relógio;
d) Modificar o mapeamento da memória.

19) Por que é importante sincronizar processos em um sistema?

20) Como as primitivas *sleep* e *wakeup* resolvem problemas de acesso concorrente em regiões críticas?

21) Por que desabilitar interrupções pode não ser uma boa idéia para garantir exclusão mútua? Justifique sua resposta.

22) O que são os semáforos? Como eles funcionam?

23) Monitores podem ser implementados com o uso de semáforos. Explique como isso pode ser feito.

24) Considere o código abaixo:

```
Program produzconsome;
Const   tamdobuffer = 100;
Var     semaphore s = 1;
        semaphore n = 0;
        semaphore e = tamdobuffer;

procedure produtor;
begin
    repeat
        produz;
        down(e);
        down(s);
        entraitem;
        up(s);
        up(n);
    forever
end;
```

```
procedure consumidor;
begin
    repeat
        down(n);
        down(s);
        tiraitem;
        up(s);
        up(e);
        consomeitem
    forever
end;

begin /*programa principal*/
    begin
        produtor; consumidor
    end
end.
```

O significado do programa mudaria se fosse alterado o seguinte? O que aconteceria?
(a) down(e) por down(s);
(b) up(s) por up(n);
(c) down(n) por down(s);

25) Encontre situações da vida real que implementam soluções de exclusão mútua. Explique porque elas são necessárias.

26) Descreva sobre espera ocupada e como evitá-la. Em que situações ela é aceitável?

27) Quais as condições para a ocorrência de *deadlock*?

28) Cite e descreva quatro estratégias para tratamento de *deadlocks*.

29) Observe a figura 3.4 do livro Sistemas Operacionais Modernos – Tanenbaum. Suponha que no passo (o) *C* requisite *S* em vez de requisitar *R*. Isso levaria a uma situação de *deadlock*? E se ele requisitasse ambos, *S* e *R*?

30) Observe a Figura 3.11(b) do livro Sistemas Operacionais Modernos – Tanenbaum. Se *D* requisitar uma unidade a mais, isso levará a um estão seguro ou inseguro? O que acontecerá se a requisição chegar de *C* em vez de *D*?

31) Um sistema tem dois processos e três recursos idênticos. Cada processo precisa de no máximo dois recursos. É possível ocorrer *deadlock*? Justifique a sua resposta.

32) Um sistema tem quatro processos e cinco recursos alocáveis. A alocação atual e as necessidades máximas são as seguintes:

	Alocado	*Máximo*	*Disponível*
Processo A	1 0 2 1 1	1 1 2 1 3	0 0 ***x*** 1 1
Processo B	2 0 1 1 0	2 2 2 1 0	
Processo C	1 1 0 1 1	2 1 3 1 0	
Processo D	1 1 1 1 0	1 1 2 2 1	

Qual é o menor valor de ***x*** para que esse estado seja seguro?

33) Considere a tabela de processos abaixo:

Processos	**Tempo de Execução**	**Prioridade**
A	10	5
B	11	5
C	05	1
D	15	2
E	10	2
F	12	3
G	09	4

Considerações:

- A prioridade 5 é a mais alta e a prioridade 1 é a mais baixa;
- A ordem de chegada dos processos segue a tabela (o primeiro processo a chegar é o processo A e o último é o processo G e todos os processos estão prontos para execução;
- Desconsidere o tempo de espera dos processos por entrada/saída;

Tarefa:

- Simule (no papel) a execução dos seguintes algoritmos de escalonamento com esses processos e determine o tempo de retorno (*turnaround time* - tempo até finalizar sua execução) de cada processo em cada algoritmo
  a) *Round-Robin*: (*quantum*= 4 unidades de tempo);
  b) FIFO;
  c) Prioridades:
     a. *quantum* dos processos com prioridade 1, 2 e 3 = 4 unidades de tempo;
     b. *quantum* dos processos com prioridade 4 e 5 = 5 unidades de tempo;

Para o algoritmo de Prioridades, execute todos os processos da prioridade mais alta antes de passar para as outras prioridades. Ou seja, execute todos os processos de prioridade 5, depois os de 4 até chegar naqueles de prioridade 1.