TE244 - Sistemas Operacionais
Lista Exercícios 1

Carlos Marcelo Pedroso

9 de setembro de 2025

Exercício 1: O IBM 360 modelo 75 é cerca de 50 vezes mais rápida do que o IBM 360 modelo 30. Todavia, o tempo de ciclo (frequência) apenas cinco vezes mais rápido. Quais fatores podem estar influindo nesta aparente discrepância? □

Exercício 2: Um computador que possui 8 vias no barramento de dados e 7 vias no barramento de endereços tem a capacidade de acessar quantos endereços de memória? □

Exercício 3: Certo computador pode ser equipado com 262.144 bytes em memória. Por que o fabricante escolheu um número tão peculiar, ao invés de um número fácil de lembrar, como 250.000? □

Exercício 4: Um computador tem um barramento com um ciclo de 250 ns, durante o qual ele pode ler ou escrever uma palavra de 32 bits em um determinado periférico. Um monitor de vídeo com 800x600 pixels, onde cada pixel necessita de 4 bytes, poderá ser atualizado 20 vezes por segundo neste sistema? Porque? □

Exercício 5: Suponha um periférico, por exemplo, o controlador de rede. Explique, neste caso específico, como é utilizada a interrupção de hardware (IRQ) na operação do sistema. □

Exercício 6: Explique porquê o *pipeline* é possível na arquitetura RISC e não na CISC. □

Exercício 7: Descreva o que é uma chamada ao sistema. Mostre como os sistemas Unix se beneficiam de uma padronização nesta área, possibilitando que aplicativos possam ser compilados em sistemas que executam sobre uma grande de plataformas de hardware. □

Exercício 8: Quando toma-se a decisão de utilizar um determinado sistema operacional, um dos pontos principais a serem considerados é a disponibilidade de *device drivers*.

a) O que é um *device driver* e qual a localização do módulo dentro do sistema operacional.
b) Porque existe tal preocupação sobre a disponibilidade de device drivers?

□

Exercício 9: O hardware possui normalmente dos modos de operação: modo kernel e modo usuário. Descreva porque isto é necessário para que o sistema operacional possa gerenciar os recursos do hardware. □

Exercício 10: Diagrama de estados de processos

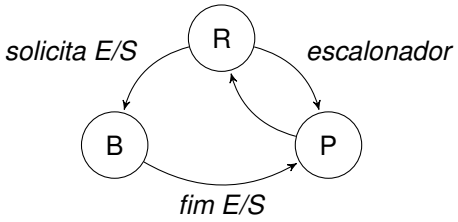

Figura 1: Diagrama de estados possíveis para processos

São três os estados básicos em que um processo pode encontrar-se dentro do sistema operacional:

R Rodando: processo está ocupando a CPU;

B Bloqueado: o processo solicitou uma operação de E/S e espera a sua conclusão;

P Pronto: o processo está pronto para a execução e aguarda o escalonamento.

Responda as seguintes perguntas:

a) Descreva o que é *salvamento de contexto* e quando isto ocorre.

b) Explique qual a diferença de um sistema com e sem *preempção*.

c) Explique qual a diferença de um sistema que opera com e sem *tempo compartilhado* (*timesharing*).

□

Exercício 11: Foi executado o comando "top" em um sistema Unix. O resultado é apresentado a seguir:

```
$ top

10:50:03 up 19 days, 12:10,  1 user,  load average: 0.56, 0.15, 0.12
Tasks: 108 total,   4 running, 104 sleeping,   0 stopped,   0 zombie
Cpu(s): 64.5%us, 18.9%sy,  0.0%ni, 16.3%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1843416k total,  1768176k used,    75240k free,   173400k buffers
Swap:        0k total,        0k used,        0k free,  1046796k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
13239 www-data  20   0  220m  36m 4068 S 30.7  2.0   0:00.76 apache2
13861 www-data  20   0  145m  16m 6332 S  7.7  0.9   0:00.14 php
13864 www-data  20   0  143m  14m 6260 R  5.0  0.8   0:00.12 php
13859 www-data  20   0  145m  15m 6180 S  4.7  0.9   0:00.08 php
2201  mysql     20   0  230m  47m 6104 S  1.0  2.6  14:38.43 mysqld
13841 pedroso   20   0 10684 1144  860 R  0.7  0.1   0:00.26 top
2798  root      20   0  191m  10m 6008 S  0.3  0.6   1:29.56 apache2
```

Responda:

a) Qual o número de processos prontos, executando e bloqueados?

b) Qual a prioridade base e dinâmica dos processos apresentados?

c) O que significa a linha Cpu(s) na saída do comando?

□

Exercício 12: Ao executar um processo no Unix foi acrescentado o símbolo "&" ao final do comando.

a) Qual o objetivo deste símbolo?

b) No Windows existe algo equivalente?

□

Exercício 13: Suponha um sistema com 4 processos, P_1, P_2, P_3 e P_4. Os tempos de execução são respectivamente de 5, 2, 3 e 4 unidades de tempo.

a) Calcule o tempo médio de execução caso seja utilizado o algoritmo FIFO sem preempção para escalonamento;

b) Calcule o tempo médio de execução caso seja utilizado o algoritmo *Shortest Job First* (ou SJF), sem preempção;

c) Utilizando o algoritmo Round Robin para sistemas com compartilhamento de tempo:
   - Calcule o tempo médio de execução caso seja utilizado o algoritmo Round Robin, com Quantum=1;
   - Calcule o tempo médio de execução caso seja utilizado o algoritmo Round Robin, com Quantum=1 e com um tempo de salvamento de contexto de 0.1;
   - Explique porque um quantum de tempo pequeno prejudica o desempenho do sistema quando o tempo de salvamento de contexto é considerado.
   - Explique porque um quantum de tempo muito grande prejudica processos interativos.

d) Utilizando o algoritmo Round Robin com 4 níveis de prioridade; suponha que a cada quantum consumido, a prioridade será decrementada e a cada quantum bloqueado a prioridade ser incrementada (até o máximo da prioridade base). Suponha que a prioridade 1 é a menor e a 4 é a maior. As prioridades base são $P_1 = 3$, $P_2 = 1$, $P_3 = 2$ e $P_4 = 4$. Considere um quantum de uma unidade de tempo.
   - Considere o Quantum=1 unidade de tempo. Calcule o tempo médio de resposta sabendo-se que o processo P4 faz uma operação de E/S em seu terceiro quantum e fica bloqueado durante quatro quantum. Qual a influência no tempo de resposta de P4?

□

Exercício 14: Descreva porque o algoritmo Round Robin com prioridades normalmente é implementado com um sistema de incremento/decremento de prioridade dinâmica. □

Exercício 15: Considere um sistema Linux que utiliza o escalonador Completely Fair Scheduler (CFS). O sistema possui um único núcleo de processamento e três processos (P1, P2 e P3) que estão prontos para serem executados. Suponha que o valor de nice para cada processo seja o seguinte:

- Processo P1: nice = -5
- Processo P2: nice = 0
- Processo P3: nice = +10

O tempo de CPU exigido de cada processo é de 500 ms.

Responda:

a) Explique o conceito fundamental do CFS. Como ele busca alcançar a justiça (fairness) na distribuição do tempo de CPU entre os processos? Diferencie-o dos escalonadores tradicionais baseados em quantum fixo.

b) Qual o papel do valor nice no CFS? Como o CFS utiliza esse valor para determinar a prioridade de um processo? Calcule o peso (weight) de cada um dos três processos (P1, P2, P3) e explique como o peso afeta a alocação do tempo de CPU.

c) Descreva a execução dos processos. Com base nos valores de nice e nos pesos calculados, como o tempo total de CPU de 500 ms será dividido entre os três processos? Justifique sua resposta detalhando a alocação do tempo de execução para cada processo.

d) Imagine que um novo processo (P4) com nice = -20 é criado e se torna executável. Explique o que acontece com a alocação de tempo de CPU para os processos P1, P2 e P3. Como o CFS se adapta a essa nova carga de trabalho?

□

Exercício 16: Descreva o funcionamento do algoritmo de escalonamento RATE MONOTONIC, para sistemas em tempo real. □

Exercício 17: Considere um sistema de tempo real com um único processador que executa quatro tarefas periódicas (T1, T2, T3, T4). O sistema utiliza o algoritmo de escalonamento Rate Monotonic (RM). As características de cada tarefa são as seguintes:

- Tarefa T1: Tempo de Execução (C1) = 10 ms, Período (P1) = 50 ms
- Tarefa T2: Tempo de Execução (C2) = 20 ms, Período (P2) = 100 ms
- Tarefa T3: Tempo de Execução (C3) = 10 ms, Período (P3) = 75 ms

- Tarefa T4: Tempo de Execução (C4) = 15 ms, Período (P4) = 200 ms
- Todas as tarefas têm seu deadline igual ao seu período.

Responda:

a) Determine a prioridade de cada uma das quatro tarefas (T1, T2, T3, T4), do mais alto para o mais baixo.

b) Calcule a utilização total do processador.

c) Avalie se o conjunto de tarefas é escalonável pelo algoritmo Rate Monotonic.

d) Descreva a execução das tarefas utilizando o diagrama de Grantt. Trace a linha do tempo de execução para as tarefas no intervalo [0, 200] e determine se os deadlines realmente podem ser cumpridos.

□

Exercício 18: Repita o exercício anterior utilizando o algoritmo *Early Deadline First* □

Exercício 19: Um problema enfrentado pela sonda Mars Pathfinder, enviado para Marte, foram constantes reinicializações do sistema causados por um *watch dog* que detectava que um processo importante não estava sendo executado. O sistema operava sem tempo compartilhado e com o conceito de preempção. O diagnóstico apontou que um processo de baixa prioridade em execução fazia a reserva do recurso R1 e era preemptado por processos mais prioritários, atrasando a sua execução. Um processo prioritário também necessitava realizar a reserva de R1. O *watch dog* detectava que este processo prioritário não estava sendo executado até o seu *deadline* e concluía que alguma coisa tinha dado muito errado, comandando então um reboot no sistema. Explique qual a causa do problema e indique uma possível solução. □

Exercício 20: Observe a figura a seguir, retirada do livro Sistemas Operacionais Modernos de A. Tanenbaum, e responda as questões.

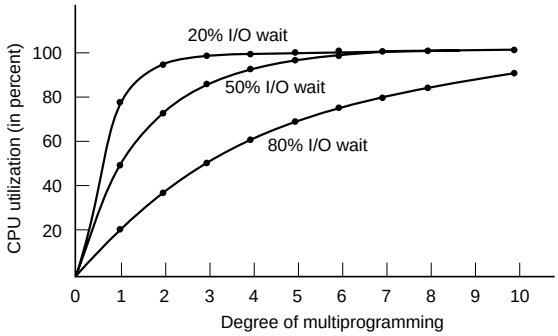

Fig. 4-3. CPU utilization as a function of the number of processes in memory.

Suponha um computador com uma CPU.

a) Em um servidor de banco de dados (grande nível de espera por E/S) é interessante atender as requisições de forma paralela (cada requisição cria uma thread) ou uma por vez? *fundamente a sua resposta*

b) Responda a mesma pergunta anterior, mas desta vez considere um computador dedicado a jogos (por exemplo, um simulador de vôo).

□

Exercício 21: Considere dois Processos, P_A e P_B. Suponha que os processos compartilham as variáveis y e z. Considere que o sistema operacional utiliza um escalonador *round robin* com compartilhamento de tempo (*time sharing*).

Processo A:	Processo B:
// inicializações inteiro x; $x = y + z$; //... *Processo A continua*	// inicializações $z = 2$; $y = 1$; //... *Processo B continua*

a) Quais os possíveis valores finais para x?

b) Mostre como seria possível resolver o problema utilizando semáforos.

□

Exercício 22: Crie um exemplo em pseudo-código de 3 processos que usam semáforos de tal forma que, em uma certa ordem de execução, eles podem:

- Terminar normalmente; ou
- Entrar em *deadlock* (impasse).

Lembre-se de indicar como os semáforos devem ser inicializados. □

Exercício 23: Suponha os Processos A e B listados a seguir:

Variáveis compartilhadas

```
semaphore mutex=1;
semaphore a=2;
semaphore b=0;
```

Processo A:	Processo B:

while true **do**
down(a);
down(mutex);
print(A);
print(B);
up(mutex);
up(b);
end while
	while true **do**
	down(b);
	down(mutex);
	print(C);
	print(D);
	up(mutex);
	up(a);
	end while

Realize a implementação em linguagem C do pseudo-código acima em sistemas Unix. A execução concorrente dos Processos A e B produz uma sequência infinita de impressão dos caracteres “A” e “B”. Assinale a única sequência possível para execução simultânea dos processos:

(a) ABCDABABCDABCD...

(b) ABCDABCDCDABAB...

(c) ABABCDABCDABCD...

(d) ACBDACBDACBDAC...

(e) ABABABCDCDABAB...

□

Exercício 24: Suponha os Processos A e B listados a seguir:

Processo A:	Processo B:

print(A);
print(B);
print(C);
	print(D);
	print(E);
	print(F);

Mostre como podem ser utilizados semáforos para satisfazer as seguintes condições:

- imprimir A antes de imprimir E; e,
- imprimir E antes de imprimir C.

Lembre-se de indicar como os semáforos devem ser inicializados. □

Exercício 25: Suponha os Processos A e B listados a seguir:

Processo A:	Processo B:

while true **do**
  print(A);
end while
	while true **do**
	  print(B);
	end while

a) Utilize semáforos para fazer com que seja impressa a sequência "*ABABABABAB...*".

b) Mostre uma possível solução para o problema do item anterior utilizando flags e indique qual o problema desta solução.

c) Seria possível ter o mesmo resultado utilizando somente a instrução *sleep(n)* (que faz o processo bloquear por *n* microsegundos)? Justifique sua resposta.

□

Exercício 26: Considere 3 processos, chamados A, B e C. O processo A produz itens e os armazena em um vetor de 10 posições chamado VETA. O processo B faz uma cópia do VETA e processa os itens, produzindo como saída uma única variável chamada ITEMB. O processo C faz uma cópia de ITEMB e usa a resposta para produzir sua saída, chamada ITEMC. Escreva em pseudo código os três processos, fazendo uma sincronização entre eles de tal forma que:

(a) Depois de preencher os 10 itens em VETA o processo A deve aguardar até que o processo B realize a cópia do vetor.

(b) O processo B deve aguardar que o processo A complete o vetor VETA para iniciar seu processamento. *Além disso, o processo B só pode produzir uma saída após o processo C ter feito uma cópia do ITEMB.* O processo B não pode produzir um novo item até que o processo C consuma o ITEMB produzido.

(c) O processo C deve aguardar a saída do Processo B para fazer a cópia do ITEMB e produzir o ITEMC.

□

Exercício 27: Considere o problema do *deadlock*

(a) Descreva em que situações ocorre o problema.

(b) Descreva como a solução do *spool* resolve este problema.

(c) Considere as soluções possíveis para o problema do *deadlock* e indique os problemas práticos para implementação destas soluções.

□