Teste Modelo de Sistemas Operativos

Eduardo Fernandes

2024/2025

Questões Teóricas

Questão 1

O escalonador de processos procura manter uma mistura equilibrada de processos intensivos em CPU e em I/O porque isso permite uma utilização mais eficiente dos recursos do sistema. Enquanto processos intensivos em I/O aguardam a conclusão de operações em dispositivos externos, o CPU pode ser utilizado por processos que requerem muita computação. Isto permite estabelecer paralelismo entre processos, maximizando a utilização dos recursos do sistema. Este paralelismo permite aumentar o desempenho do sistema, pois temos mais processos a executar concorrentemente.

Questão 2

Para o sistema operativo apresentado, escolheria o algoritmo de escalonamento **MLFQ** (Multi Level Feedback Queue). Este algoritmo distribui a execução de processos em filas de espera com prioridade, sendo a primeira fila a que tem maior prioridade. A cada processo é atribuída uma porção de tempo em cada fila, e periodicamente cada processo é elevado para a fila de cima.

Uma das **vantagens** deste algoritmo é o baixo turn around time e response time, pois processos rápidos com baixa duração executam rapidamente e processos com alta prioridade são frequentemente trocados, aproximando-se do algoritmo Round Robin. Uma **preocupação** para este algoritmo é determinar o número de filas de espera, a porção de tempo para cada processo em cada fila e quando cada processo deve ser elevado, pois é necessário efetuar uma configuração cuidadosa e saber estatísticas relativas à utilização do sistema.

Questão 3

1. Combinar paginação com mecanismos de swapping é uma técnica muito **vantajosa**, pois permite aos programadores desenvolver programas, sem se preocupar com o facto de as suas estruturas de dados cabem ou não na memória fisica, isto é, permite ter programas que ocupam um espaço maior que a própria memória principal. Uma **preocupação** que o sistema operativo deve ter é a substituição de páginas quando a área de swap está cheia. Se o sistema operativo remover páginas que são frequentemente usadas, perde eficiencia (muitos acessos a disco), logo é necessário ter uma boa política de substituição para minimizar page faults.

2. É preferivel usar uma **partição de disco**, pois evita a sobrecarga associada à gestão do sistema de ficheiros (alocação de blocos, verificação de permissões, ...) tornando o acesso à swap mais rápido e eficiente.

Questão 4

Uma possível razão para o mau desempenho das aplicações é o facto de os pedidos que se situam mais longe dos cilindros não chegarem a ser realizados, caso sejam feitos pedidos que estejam mais próximos da cabeça do disco, que têm maior prioridade. Um algoritmo de escalonamento capaz de melhorar a desempenho destas aplicações é o algoritmo **Circular SCAN**, que percorre o disco, da track mais exterior para a mais interior e vice-versa. Este algoritmo de elevador é justo para qualquer pedido a qualquer track.

Questão 5

[B] Shortest Job First (SJF).

[C] First-Come First-Served (FCFS).

Questão 6

[D] A paginação de memória pode gerar fragmentação interna mas não externa.

Questão 7

[B] Sim, tempos de acesso aleatório são mais altos do que tempos de acesso sequenciais em discos HDD.

[D] Sim, já que tende a diminuir o tempo médio de seek do disco HDD.

Questões Práticas

Questão 8

Resolução apresentada no ficheiro `question-8.c`

Questão 9

Resolução apresentada no ficheiro `question-9.c`

Questão 10

Resolução apresentada nos ficheiros `SOGPT.c` e `servidor.c`

Se a abertura do pipe com nome `fifoc_name` (linha 15) fosse feita logo após a criação do mesmo (linha 8), o programa não iria funcionar devido ao facto de o cliente bloquear na abertura do pipe `fifoc_name`, que consequentemente iria bloquear o servidor na abertura do pipe `fifo_server`, pois todos os clientes ficam bloqueados no `open("fifoc_name", O_WRONLY)`. Por outras palavras, o programa `cliente` iria bloquear na abertura do pipe com nome `fifoc_name`, e como o servidor apenas abre esse canal após a abertura do pipe `fifo_server`, este fica bloqueado, pois todos os clientes abrem o canal `fifo_server` após a abertura do `fifoc_name`.

Questão 11

C A operação `op(...)` no máximo executa 10 vezes.

Questão 12

A O programa cria processos que executam a `func1()` sequencialmente.

D O output esperado é:

```
processo 0
processo 0 terminou
processo 1
processo 1 terminou
processo 2
processo 2 terminou
```

Questão 13

C `lseek(fd, sizeof(Matricula) * 11, SEEK_SET);`

Questão 14

A Após um cliente terminar a sua execução, e caso não exista mais nenhum cliente a executar, o servidor termina também o seu programa e não atende mais clientes.

B Após um cliente terminar a sua execução, caso estejam outros programas clientes a executar e escrever mensagens para o servidor, o servidor termina também o seu programa, não atendendo estes clientes.

Questão 15

B O seguinte código está em falta (linha 25)

```
1   close(pfd[0][0]);
2   close(pfd[1][1]);
```

D O programa emula o comando `cat etc/passwd | sort | wc -l`