

Teste de Sistemas de Operação — 30 de Maio de 2009

Duração: 30 min

B

NOME: _____

Indique apenas uma das alternativas. Respostas erradas <i>descontam</i> na nota.
--

1. Um pedido de entrada e saída sem buffering: (letra (c))
 - a. demora menos tempo a ser processado
 - b. demora mais tempo a ser processado
 - c. não possibilita a sobreposição de I/O com o uso de CPU

2. O programa abaixo deveria implementar em UNIX um processo que faz um append em um ficheiro. Este trecho de código não executa corretamente. Por que? (letra (a))
 - a. O ficheiro original é sobrescrito.
 - b. O ficheiro original é removido e um outro é criado em seu lugar.
 - c. O ficheiro não tem permissão para escrita.

```
if ( (fd = open (``filename``,O_RDONLY)) ! 0) err_sys(...);  
...  
if ( (lseek(fd,0,SEEK_END) ! 0)) err_sys(``lseek error``);  
if ( (write(fd,buffer,100) != 100)) err_sys(``write error``);
```

3. O mecanismo de interrupções (trap or software interrupt) é fundamental para (letra (a))
 - a. executar funções de sistema.
 - b. executar processos de sistema em modo-batch.
 - c. implementar uma TLB (Translation Lookaside Buffer).

4. A função de sistema `kill()` serve (letra (a))
 - a. para enviar sinais explícitos a um processo filho.
 - b. apenas para terminar um processo filho.
 - c. para terminar o processo que invoca a função.

5. Em um sistema que suporta multiprogramação e *time-sharing*, vários utilizadores podem compartilhar o sistema simultaneamente. Se este suporte não for bem implementado, esta situação pode gerar vários problemas de segurança. Um possível problema de segurança pode ser: (letra (a))

- a. processos podem invadir a área de memória de outros processos
- b. processos podem executar em modo não protegido
- c. processos podem utilizar a CPU de forma caótica

6. Suponha que, ao invés de implementar semáforos com uma fila FIFO, usa-se “last come, first served” (pilha). Este semáforo pode ser usado para implementar exclusão mútua “segura” com n processos cíclicos? (“segura” significa que nenhum processo fica bloqueado indefinidamente enquanto outros continuam executando). (letra (b))

- a. sim.
- b. não.
- c. não há dados suficientes para responder a esta questão.

7. A função de sistema `exec1 ()` permite substituir o programa do processo que executa a função por um outro; isto acontece porque

- a. se altera todo o contexto do processo.(letra (b))
- b. se modifica o segmento de dados e texto do processo.
- c. se cria um processo filho que vai executar o novo programa dado como argumento na função `exec1 ()`.

8. Considere a seguinte frase: "O Unix não é um sistema adequado para aplicações de tempo real porque um processo executando em modo kernel não pode ser interrompido". Esta afirmativa é falsa ou verdadeira? (letra (b))

- a. falsa.
- b. verdadeira.
- c. não há dados suficientes para responder a esta questão.

9. A instrução `fork ()` cria um processo filho. Suponha que executou o código abaixo. Podemos dizer que: (letra (a))

- a. foram criados 3 processos filho
- b. foram criados 5 processos filho
- c. foram criados 4 processos filho

```
for (i=0; i<2; i++)
    if (fork()==0) {
        escreve(i);
    }
```

- 10.** Quais destas instruções devem ser executadas em modo privilegiado? (letra (b))
- Modificar o valor do relógio, ler valor do relógio, desligar interrupções
 - Mudar de modo utilizador para modo kernel, desligar interrupções
 - Limpar a memória, modificar o valor do relógio
- 11.** A função de sistema `wait()` permite uma forma limitada de comunicação entre processos pai e filho, nomeadamente (letra (b))
- permite que o pai espere até que um filho em concreto termine.
 - permite que o pai espere até que um filho termine.
 - permite que o filho sincronize com o pai esperando que este termine.
- 12.** A instrução `fork()` cria um processo filho. Suponha que se executou o código abaxo. Podemos dizer que: (letra (a))
- o processo pai escreve o valor 2 e o processo filho escreve o valor 3
 - o processo pai escreve o valor 3 e o processo filho escreve o valor 2
 - apenas o processo pai escreve o valor da variável `x`.
- ```
if (fork()!=0) x=2; else x=3;
escreve(x);
```
- 13.** A implementação do redirecionamento de output de um comando para um ficheiro, requer o uso da função de sistema `dup2` para: (letra (b))
- associar o descriptor do ficheiro ao descriptor 0
  - associar o descriptor do ficheiro ao descriptor 1
  - duplicar o descriptor do ficheiro.
- 14.** Uma instrução que verifique e modifique uma posição de memória de forma atómica (letra (a))
- é fundamental para implementar sincronização entre processos
  - não é necessária para implementar sincronização entre processos
  - torna a implementação da sincronização entre processos mais fácil

15. O que entende por *buffering*, *caching* e *spooling*? (letra (b))
- a. Buffering permite sobrepor comunicação com operações de entrada e saída, caching permite armazenar dados que serão utilizados no futuro, spooling é o processo de enfileirar tarefas de impressão
  - b. Buffering permite sobrepor computação com operações de entrada e saída, caching permite armazenar dados que serão utilizados no futuro, spooling permite processar tarefas sobrepondo computação com entrada e saída
  - c. Buffering permite armazenar dados que serão utilizados no futuro, caching é uma tecnologia de acesso rápido para armazenamento e recuperação de dados, spooling é o processo de enfileirar tarefas de impressão
16. O uso de DMA (Direct Memory Access) tem o efeito de? (letra (b))
- a. Reduzir o número de vezes que os dados passam no bus do sistema.
  - b. Reduzir a intervenção do SO na operação de I/O
  - c. Facilitar o acesso directo dos processos à memória da máquina.
17. Considere a implementação abaixo. A função deste programa é: (letra (c))
- a. criar um histórico de comandos.
  - b. comparar um comando com “gwc”.
  - c. ler e interpretar linhas de comando inseridas pelo utilizador.

```
main() {
 char *linha;
 COMMAND com;

 while (1) {
 if ((linha = readline("my_prompt$ ")) == NULL)
 exit(0);
 if (strlen(linha) != 0) {
 add_history(linha);
 com = parse(linha);
 ...
 if (strcmp("gwc", com->cmd) == 0) {
 int input_fd = open(com->argv[2], O_RDONLY);
 gwc(com->argv[1], input_fd);
 }
 ...
 }
 free(linha);
 }
}
```

**18.** Qual dos seguintes códigos resolve o problema de exclusão mútua, entre 2 processos, de forma correta? (letra (a))

- a. SOLUTION 1
- b. SOLUTION 2
- c. SOLUTION 3

```
SOLUTION 1
int turn;
proc(i)
int i;
{
 while (TRUE)
 {
 compute;
 while (turn <> i);
 CS;
 turn <- i+1 mod 2;
 }
 turn <- 1;
 proc(0) AND proc(1);
}
```

```
SOLUTION 2
boolean flag[2];
proc(i)
int i;
{
 while (TRUE)
 {
 compute;
 while (flag[i+1 mod 2]);
 flag[i] <- TRUE;
 CS;
 flag[i] <- FALSE;
 }
 flagp[0] <- flag[1] <- FALSE;
 proc(0) AND proc(1);
}
```

```
SOLUTION 3
boolean flag[2];
proc(i)
int i;
{
 while (TRUE)
 {
 compute;
 flag[i] <- TRUE;
 while (flag[i+1 mod 2]);
 CS;
 flag[i] <- FALSE;
 }
 flagp[0] <- flag[1] <- FALSE;
 proc(0) AND proc(1);
}
```

**19.** Suponha que existem 5 tarefas para processamento com um tempo estimado de execução de 30, 12, 20, 10 e 2 segundos e que têm as seguintes prioridades 3, 1, 4, 5 e 2, respectivamente, em que valores mais baixo de prioridade significam maior prioridade de execução. Ignorando o tempo de troca de contexto e considerando as estratégias “o mais curto primeiro”, “prioridades” e “ordem de chegada” temos, respectivamente, que o tempo médio de **execução** por processo (*turnaround*) é (assuma que uma tarefa uma vez selecionada corre até terminar) (letra (a))

- a. 16.4, 26.8 e 41.2 segundos
- b. 26.8, 41.2 e 16.4 segundos
- c. 41.2, 16.4 e 26.8 segundos

**20.** Para que serve um mailbox? (letra (b))

- a. para guardar mensagens de email.
- b. para implementar comunicação entre processos.
- c. para receber mensagens de email