UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática : SISTEMAS OPERATIVOS
**VERSÃO : 2**

Frequência I, 3 de Abril de 2019, 14.00Horas Escala 0:20 Sem Consulta Duração: 1h45m

Nº_____ Nome_____

Pontuação: A (0.25) Cada. B (0.5) Cada

**(A) Assinale na folha do enunciado as afirmações que se seguem conforme sejam verdadeiros(V) ou falsos(F)**

1. ____ Cada operação de I/O requer que o software e o hardware coordenem suas operações usando Polling ou Interrupções.
2. ____ As interrupções não são permitidas numa arquitetura estrita/rigorosa de von Neumann.
3. ____ A CPU é responsável por salvar todos os valores dos registos da CPU quando ocorre uma interrupção.
4. ____ Num computador convencional com apenas um CPU vários programas podem executar simultaneamente.
5. ____ Se um processo individual não puder tirar proveito da sobreposição de operações de CPU e I/O, o sistema operativo pode se sobrepor a execução da CPU de um processo com a operação de I/O de outros processos.
6. ____ Comutação de contexto pode ocorrer sempre que o sistema operativo obtém o controle do processador.
7. ____ A função printf() da biblioteca padrão da C invoca sempre a chamada ao sistema write().
8. ____ Operação em Dual-Mode permite a execução de dois sistemas operativos.
9. ____ Um Computador Moderno "Multi-Core" não exibe multi-programação mas apenas multi-processamento.
10. ___ A comutação do contexto requere a guarda do estado dum processo.
11. ___ Durante a execução dum processo (C/Linux) instâncias duma função, variáveis e valor do retorno, são criados na área de memória do processo conhecido como Pilha (stack).
12. ___ A chamada ao sistema exec() em Linux irá criar um novo processo para executar um novo programa.

**(B) Na resposta a cada um dos seguintes itens seleciona inequivocamente a única opção adequada assinalando-a neste enunciado**

1. O que Não é um exemplo de software de sistema?
   a. Interpretador de linha de comando (CLI)
   b. Sistema de Gestão de Processos
   c. Software de processamento de texto
   d. Sistema de gestão de janelas

2. Um sistema operativo faz cada um dos seguintes Exceto
   a. Aloca os componentes do computador para diferentes programas
   b. Sincroniza as atividades dos programas individuais
   c. Garante que os programas encerrem a sua execução
   d. Fornece os mecanismos gerais que são necessários para que os programas executem em perfeita harmonia

3. Um sistema operativo é distinguido de outro software do sistema por cada um dos seguintes Exceto
   a. Ele interage diretamente com o hardware para fornecer uma interface usada por outro software de sistema e softwares de aplicação.
   b. Ele permite que diferentes aplicativos compartilhem os recursos de hardware por meio de suas políticas de gerenciamento de recursos
   c. Ele pode ser usado para oferecer suporte a uma ampla gama de domínios de aplicativo
   d. As abstrações de recursos de hardware que ele fornece são convenientes, mas o seu uso por aplicativos é opcional.

4. O que Não é um exemplo de um recurso que é habitualmente multiplexado por espaço?
   a. CPU
   b. RAM de vídeo
   c. Disco rígido
   d. Memória principal

5. O que Não é um exemplo de um recurso que é habitualmente multiplexado por tempo?
   a. Interface de rede
   b. CPU
   c. Acelerador de gráficos
   d. Memória principal

6. Qual dos seguintes Deve ser implementado como software confiável no Kernel ?
   a. Sistema de Gestão de Base de Dados
   b. Gerente de multiprogramação
   c. Compilador
   d. Interpretador de comandos

7. Qual declaração sobre DMA é sempre verdadeira?
   a. DMA existe em todos os computadores com sistema operativo.
   b. Exige sempre um Hardware especial/dedicado.
   c. A CPU está sempre preocupada com a transferencia dos dados
   d. O DMA é usado em Linux nos dispositivos de tipo Bloco e do tipo Caracter

8. Qual declaração sobre o tratamento das interrupções pela CPU está incorreta?
   a. O processador cessa de executar a sequência de instruções "atuais"
   b. A CPU começa a executar uma nova sequência de instruções
   c. O hardware guarda a localização do PC (program counter) antigo
   d. Quando a CPU determina que a nova sequência de instruções é concluída, a CPU restaura o PC antigo e a execução da sequência original prossegue.

9. Considere o seguinte programa escrito em C. `main() { printf("ola\n"); asm("cli"); }`
Qual declaração está correta?
   a. O Programa Compila Sem erro e Execute Sem Error imprimindo “ola”
   b. O Programa Compila Com Avisos e Execute Sem Error imprimindo “ola”
   c. O Programa Compila Com Avisos e Execute com Error imprimindo “ola”
   d. O Programa Compila Com Avisos e Execute com Error sem imprimindo “ola”

10. Um processo (modelo de 5 estados) pode fazer as transições seguintes exceto qual ?
   a. Transitar do estado Running (Execução) para Terminated (Terminado)
   b. Transitar do estado Running (Execução) para Waitiing (Espera)
   c. Transitar do estado Ready (pronto) para Running (Execução)
   d. Transitar do estado Ready (pronto) para Waiting (Espera)
   e. Transitar do estado Waiting (Espera) para Running (Execução)

**Secção C** Responder na Folha de Teste

1. Um sistema de tempo real tem 3 processos onde o processo 1 tem período 12 e tempo de execução 4, o processo 2 tem período 6 e tempo de execução 1 e o processo 3 tem período 8 e tempo de execução 3. O sistema é escalonável ? (deve apresentar quaisquer cálculos)

**(2 valores)**

2. Mostrar o trace (diagrama temporal de execução) do funcionamento do seguinte programa. Num sistema operativo que utilize escalonamento FCFS (First Come First Served/primeiro a chegar, primeiro a ser servido") não preemptivo determine o output

```
int  pid, x = 2;
pid = fork();
if (0==pid) {
   pid = fork();
   if (0 != pid)
      x--;
}
x=x+1;
printf("%d ", x);
```

**(2 valores)**

3. (Responder na Folha de Teste) Faça o diagrama temporal do processamento dos processos indicados na tabela abaixo seguindo o algoritmo de escalonamento por prioridades fixas com preempção e calcule o tempo médio de circulação (Turnaround).

| processo | tempo de chegada | prioridade | duração |
|---|---|---|---|
| P1 | 1 | 3 | 3 |
| P2 | 2 | 4 | 2 |
| P3 | 2 | 2 | 3 |
| P4 | 3 | 1 | 2 |

**(2 valores)**

4. Escreva a função removerChar (char *fonte, char *destino, char a, char b, int blockSize)

   Esta função irá fazer uma cópia do ficheiro cujo nome é o parâmetro fonte para o ficheiro com nome destino com todos os caracteres especificados pela parâmetro "a" substituídos pelo parâmetro "b"
   A função deverá ser eficiente no sentido de ler e escrever em blocos dum tamanho especificado (parâmetro *blockSize*)

   *Exemplo Utilização*: removerChar(in.txt, out.txt, 's ', 'c ', 512);

   O ficheiro com o string "são" → "cão"

   Não é necessário especificar as bibliotecas padrão (#include <fcntl.h> etc.).

**(3 valores)**

5. O ficheiro /etc/passwd é um ficheiro de texto com um registo por linha, cada linha descreve a conta dum utilizador. Cada registo consiste em sete campos, separados por dois pontos. Os 7 campos são: nome do utilizador, password, id, grupo, comentário, diretório raiz e o caminho do programa do shell por defeito
   Exemplos de registos são :

```
ruisantos:*:11:123:Rui Santos, Sala 62:/home/rsantos:/bin/bash
ruisantosneves:*:12:123:Rui Neves, Sala 63:/home/rsneves:/bin/sh
```

   Criar um bash script (freq.sh) para

   (i) Indicar que o interpretador do Bash é o ficheiro /etc/bin/bash
   (ii) Criar um ficheiro "passwd.bck" com apenas o nome do utilizador e id (ordenado pelo valor do "id" )
   (iii) Pedir o nome dum utilizador e elimine este registo, se existir, do ficheiro /etc/passwd

**(3 valores)**

## Normas, Bibliotecas de linguagem C e Linux

**Notas:** CPU=central Processing nit : I/O=Input/Output – entrada/saída

**void * malloc( size_t size );** Allocates size bytes of memory. returns a pointer to the allocated memory, NULL if there is an error.

**pid_t fork ( void** ); creates a new process.

**int execvp( const char *file, char *const argv[]);** The **exec** family of functions replaces the current process image with a new process image.

**pid_t wait(int *status);** The *wait*() function shall suspend execution of the calling thread until status information for one of the terminated child processes of the calling process is available. If *wait*() returns because the status of a child process is available, the function shall return a value equal to the process ID of the child process for which *status* is reported

**int open( char *filename, int access );** Opens a file. On success, open returns a nonnegative integer, called the file handle or file descriptor. On error, returns -1. Access Examples : O_RDONLY .

**int read ( int handle, void *buf, unsigned len );** read attempts to read len bytes from the file associated with handle into the buffer pointed to by buf. On successful completion, read returns an integer indicating the number of bytes placed in the buffer.

**int write ( int handle, void *buf, unsigned len );** write writes a buffer of data to the file or device named by the given handle write returns the number of bytes written. write returns the number of bytes written.

## LINUX

**grep** prints lines that contain a match for a pattern. Options

-e pattern Use pattern as the pattern. -f file Obtain patterns from file, one per line.

-I Ignore case distinctions, so that characters that differ only in case match each other.

-v Invert the sense of matching, to select non-matching lines. (-v is specified by POSIX.)

**Sort** merge, or compare all the lines from the files given (or standard input.) Options

-n Sort numerically:

-t SEPARATOR Use character SEPARATOR as the field separator when finding the sort keys in each line. By default, fields are separated by the empty string between a non-whitespace character and a whitespace character.

-k POS1[,POS2] The recommended, POSIX, option for specifying a sort field.

**cut** Print selected parts of lines from each FILE to standard output. Options

-d DELIM use DELIM instead of TAB for field delimiter

-f LIST select only these fields;

**wc** - print newline, word, and byte counts for each file. With no FILE, read standard input. Options

-m, --chars print the character counts -l, --lines print the newline counts -w, --words print the word counts

**BASH IF Syntax**

```
if [ expression ]; then
<commands>
else
<commands>
fi
```

**BASH FOR Syntax**

```
for variable in List_OF_Values
do
<commands>
done
```

**Regular Expressions**

| | | | |
|---|---|---|---|
| abc… | Letters | {m} | m Repetitions |
| 123… | Digits | {m,n} | m to n Repetitions |
| \d | Any Digit | * | Zero or more repetitions |
| \D | Any Non-digit character | + | One or more repetitions |
| . | Any Character | ? | Optional character |
| \. | Period | \s | Any Whitespace |
| [abc] | Only a, b, or c | \S | Any Non-whitespace character |
| [^abc] | Not a, b, nor c | ^…$ | Starts and ends |
| [a-z] | Characters a to z | (…) | Capture Group |
| [0-9] | Numbers 0 to 9 | (a(bc)) | Capture Sub-group |
| \w | Any Alphanumeric character | (.*) | Capture all |
| \W | Any Non-alphanumeric character | (abc\|def) | Matches abc or def |