UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Frequência I, 31 de Março de 2016, 14Horas Escala 0:20 Sem Consulta Duração: 1h30m

Grupo A: História, Estrutura e Arquitectura dum Sistema Operativo (SO) (6 valores)

1. Indique cinco funcionalidades/componentes principais dum Sistema Operativo. **(1)**

2. Explique o papel do "temporizador" dum sistema computacional. **(1)**

3. Nos sistemas operativos dos computadores *Multi-Core* actuais que contêm múltiplas unidades de processamento faz sentido falar de multi-programação ou apenas de multi-processamento ? Explique. **(1)**

4. Para que servem as *systems calls* (Chamadas ao Sistema)? Qual é o *Modo* em que são executadas? **(1.5)**

5. As funções seguintes podem invocar ou não chamadas ao sistema Linux : read(), strcmp(), clock(), scanf() e sqrt() definidas nos ficheiros <unistd.h>, <string.h> <time.h> <stdio.h> e <math.h>respectivamente? Explique. **(1.5)**

Grupo B: Programas, Processos e Escalonamento (8 valores)

6. Desenhe o diagrama de 6 (seis) estados dum processo – mostrando as transições possíveis no estado dum processo, este é uma simples extensão do modelo de 5 estados apresentados nas aulas. Neste modelo existe mais um estado, para indicar processos que estão em espera prolongada e que assim não necessitam de ocupar memória principal (RAM)**(1)**

7. Define e explique, mostrando as diferenças entre eles, os conceitos de *programa* e *processo*. **(1)**

8. Faça o diagrama temporal da execução dos processos indicados na tabela abaixo seguindo o algoritmo de escalonamento por prioridades com preempção. Calcule o tempo médio de circulação (turnaround) **(2)**

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

9. No contexto de escalonamento de tempo real considere um sistema de 2 processos periódicos onde :
   o processo 1 tem período 5 e tempo de execução 3,
   o processo 2 tem período 7 e tempo de execução 2
   (i) Este sistema é escalonável ou não? Explique.
   (ii) Exemplifique o algoritmo de escalonamento de tempo real EDF (Earliest Deadline First) com o sistema em cima a partir dum tempo inicial zero até 20 unidades do tempo **(2)**

10. Qual é o output do seguinte programa? (Deverá mostrar o *trace* do funcionamento do programa) **(2)**

```
{
    int  pid, x = 0;
p1: pid = fork();
    if (0 == pid ) {
        x++;
        if (1==x) goto p1;   /* goto : instrução de salto incondicional */
        x++;
    }
    else {
       x--;
    }
    printf("x=%d\n",x);
}
```

Grupo C: Exercícios Práticos (6 valores)

(i) Escreva a função *bmpFile()* que

(a) verificará se um ficheiro é do tipo bmp (bitmap image file) – se não a função devolverá 1 e

(b) se for um ficheiro BMP irá a seguir ler o tamanho do ficheiro e imprimirá no ecrã "BMP SIZE z" onde z é o tamanho do ficheiro bmp em bytes e depois devolverá zero.

Para saber se um ficheiro é do tipo BMP e ver o seu tamanho deverá seguir as instruções em baixo
A função terá de usar apenas as funções I/O de baixo nível e printf(). Poderá usar como base o código dado em baixo.

O seu programa deverá efectuar a verificação de erros e a função *bmpFile*() deverá devolver -1 caso que haja erro e 0 ou 1 caso a função execute com sucesso onde 1 indique que o ficheiro não é do tipo bmp e 0 caso que seja.

```
/* ficheiro  bmpFile.c */
int bmpFile( char * fileName )
{
   //abrir ficheiro : devolve -1 caso de erro
   //faça leitura do cabeçalho dum ficheiro BMP : isto são os primeiros 14 Bytes
   //se não conseguir ler 14 bytes então o ficheiro de certeza não é um bmp.
   //verificar se os primeiros dois bytes do cabeçalho são 0x42 0x4D hexadecimal
            isto indique se um ficheiro é do tipo BMP ou não
   //se é um ficheiro BMP então os próximos 4 quatro bytes contêm o tamanho como um inteiro (int)
 }
```

Não é necessário especificar as bibliotecas padrão (#include <fcntl.h> etc.) no seu programa.

(iii) Um programa completo escrito no ficheiro *main.c* pretende chamar e utilizar esta função.

- Apresente o comando (*cc*) para compilar os ficheiros *main.c* e *bmpfile.c* para ficheiros objecto.
- Apresente o comando (*cc*) necessário para fazer a linkagem dos ficheiros *main.o* e *bnmpfile.o* e criar um ficheiro executável cujo nome é *freq2016.*
- *Explique para que serve a variável do Shell "PATH".*
- O valor da variável PATH do utilizador é /usr/local/bin:/bin:/usr/bin e o utilizador não se encontra em nenhuma destas directórias - Apresente e explique o comando necessário para executar o programa *freq2016* escrevendo apenas o nome do programa.

Normas e Bibliotecas de linguagem C

goto : instrução de salto incondicional de c

Norma de Escalonamento com Prioridades Fixas – Prioridade mais alta = Número de prioridade mais baixa (>=1)

Cli: The assembler instruction to turn off the interruptions flag, disabling maskarable interruptions.

int open(char *filename, int access); Opens a file. If successful, returns a non-negative integer, -1 on failure.
Read Only Acces Mode, the flag: 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 success, returns an integer indicating the number of bytes placed in the buffer.
On error, -1 is returned..
void * malloc(size_t size); The malloc() function allocates size bytes of memory and returns a pointer to the allocated memory, malloc() returns a NULL pointer if there is an error.

```
int bmpFile( char * fileName )
{
   int fd = open(fileName, O_RDONLY)
   if (fd < 0 ) return -1;
   char buf[14];
   int lidos = read(fd,buf,14);
   if (lidos<14) return 1;
   if  ( ! ( buf[0]== 0x42 && buf[1]== 0x4D ) ) return 1;

   int size;
   read(fd,&size,4)
   printf("BMP SIZE %d", size);

   int *size = &buf[2];
   printf("BMP SIZE %d", *size);

   int size;
   memcpy(&size,&buf[2],4);
   printf("BMP SIZE %d", size);


   return 0;
 }
```