UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Frequência I, 18 de Abril de 2018, 14.10Horas Escala 0:20 Sem Consulta Duração: 1h30m

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

1. Explique o conceito dum "Interpretador de comandos". **(1).**

2. Para qual tipo de operações é o mecanismo conhecido como "DMA" útil? Exemplifique e explique. **(1).**

3. Qual é a diferença entre multiprogramação e multiprocessamento? **(1)**

4. Explique porque é que atualmente cada computador usa um sistema operativo, sabendo nós que houve tempos em que ele não era usado. **(1)**

5. Considere o seguinte programa escrito em C. Este será compilado com o compilador gcc que permite embutir código assembler dentro dum programa em C através da palavra chave asm (*the gcc inline assembly feature).* Explique detalhadamente os três eventos do processo de compilação e execução mostrados em baixo **(2)**

```
//cli.c
main() {
    printf("ola mundo\n");
    asm("cli");
    printf("Dono da maquina\n");
}
```

i.[fedora]$ gcc -o cli cli.c
cli.c: warning: implicit declaration of built-in function 'printf'
ii.[fedora]$ cli
bash: cli: command not found
iii.[fedora]$ ./cli
Ola mundo
Segmentation fault

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

6. Qual a relação entre "Programas", "Processos", e "Jobs" no sistema operativo Linux? Explique. **(1)**

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

```
int  pid, x = 2;
if ( (pid = fork()) ==0) {
    fork();
    pid=fork();
    if (0==pid) x--;
}
else {
   execl("/bin/date", "date", 0 );
   x=x+2;
}
printf("x=%d\n",x);
```

8. Mostre as transições e desenhe o diagrama de 7(sete) estados dum processo - este é uma simples extensão do modelo de 5 estados apresentados nas aulas. Neste modelo existem mais dois estados, para indicar processos que já não se encontram em memória principal (RAM) estão em memória secundária/temporária (disco) e podem ser bloqueados ou desbloqueados. Justifique as transições *de* e *para* estes dois novos estados. **(2)**

9. No contexto de escalonamento de tempo real : **(3)**
   - **a.** o que são "processos periódicos" e "prazo" (deadline) de execução" ?
   - **b.** considere um sistema de 2 processos onde :
     o processo 1 tem período 5 e tempo de execução 2,
     o processo 2 tem período 8 e tempo de execução 4
     Este sistema é escalonável ou não? Explique.
   - **c.** Explique o funcionamento do algoritmo dinâmico, EDF, de escalonamento do tempo real e exemplifique com o sistema em cima dum tempo inicial zero até 15 unidades do tempo .

Grupo C: Exercícios Práticos (6 valores 4 + 2)

10. Antes de carregar para a memoria o código dum controlador de dispositivo o sistema operativo deverá verificar a integridade do ficheiro (para verificar se o ficheiro não tenha sido alterado por nenhum vírus).
    Por isto utiliza-se um código de verificação (chamado MAC) que é concatenado depois do fim do ficheiro.
    A sua tarefa é escrever uma função (VerifyIntegrity) que será usado pelo sistema operativo neste processo de verificação, embora o cálculo do MAC será feito pelo modulo/chip do chamado TPM (Trusted Platform Module) do Computador que armazena as chaves criptográficos e um cripto-processador seguro.
    Implemente a função seguinte que especifica o algoritmo necessário:

```
int VerifyIntegrity( char * fileName , int size)
{
```

- Abrir o ficheiro : devolve -1 caso de erro
- Ir para o fim do ficheiro (reposicionar o offset !) e verificar o tamanho do ficheiro é pelo menos 1024bytes
- Avançar 4 bytes.
- O Tamanho (T) do Mac é o valor do próximo byte (Verificar se o seu valor é 16, 32 ou 64)
- Faça a leitura do MAC (os próximos 32 bytes)
- Chamar a função do verificação TpmHMacVerify ver o seu sintaxe em baixo
- Return true(1) or false (0)

```
}
```

int TpmHMacVerify (const char *mac, int size, int fd);

Returns true(1) or false (0) if the HMAC-SHAx algorithm, where x is specified by size, for the file with open file descriptor, fd, verifies or not with the given mac code, mac.

Devolve verdade(1) ou falso (0) se o algoritmo HMAC-SHAx, onde x diz respeito ao tamanho, size, para o ficheiro aberto com um descritor de ficheiro, fd, verifica-se ou não com o código, mac, dado

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

11. Considere os comandos seguintes no bash shell de linux.

```
shell$ ls -l -type a.out
-rw-r--r--  1 a12345  alunos  12672 Apr 18 14:00 a.out    :type   executable i386
shell$ a.out
-bash: a.out: command not found
shell$ ./a.out
-bash: ./a.out: Permission denied
```

- Explique os problemas encontrados pelo Shell.
- Para correr o programa escrevendo apenas o nome do programa (*shell$ a.out*) quais são os comandos necessários para efetuar ?

Normas, Bibliotecas de linguagem C

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.

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.

off_t lseek (int fd, off_t offset, int whence); lseek repositions the file offset of the open file associated with the file descriptor fd to the argument offset according to the directive whence: SEEK_SET→The file offset is set to offset bytes. SEEK_CUR→The file offset is set to its current location plus offset bytes. SEEK_END →The file offset is set to the size of the file plus offset bytes. Upon successful completion, lseek() returns the resulting offset location as measured in bytes from the beginning of the file.