UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática

SISTEMAS OPERATIVOS

Exame 1, 8 de Junho de 2016, 14.30 Horas Escala 0:20 Sem Consulta Sala 6.04/6.06 Duração: 2h30m

Nº/NOME _____

Observações: Não é permitida a consulta de livros ou de apontamentos. Não se esclarecem dúvidas durante a prova. Se tiver dúvidas, indique na folha de teste a sua interpretação. Utilize uma caligrafia legível.

Grupo A: História, Estrutura e Arquitectura dum Sistema Operativo (4 valores)

1. Explique os termos "multiprogramação" e "multiprocessamento" no contexto dum sistema operativo.
2. Para qual tipo de operações é o mecanismo conhecido como "DMA" útil? Exemplifique e explique.
3. Para que servem as *systems calls* (Chamadas ao Sistema)? Qual é o *Modo* em que são executadas? As funções seguintes utilizam ou não chamadas ao sistema : read(), getchar() e sqrt() definidas nos ficheiros <unistd.h>, <stdio.h> e <math.h>respetivamente ? Explique.
4. Compare a segurança dum sistema operativo que opere em "Dual Mode" com um outro que não tem este tipo de operação. A sua resposta deverá incluir o termo "instrução *privilegiada" (devidamente explicada) e fazer referência de como a operação em dual mode é implementada.*

Grupo B: Programas, Processos e Threads(4 valores)

5. Qual a relação entre "Processos", "Threads" e "Jobs" no sistema operativo Linux? Explique.
6. Explique a função execvp() e diga qual é o output do seguinte programa? (Deverá mostrar o funcionamento do programa usando um fluxograma

```
int main(){
   char * args[2]={"date",NULL}
   int  pid, x = 5;
   pid = fork();
   if (pid == 0) {
      fork();
      x++;
   }
   else { execvp( *args, args ); }
   printf("x=%d\n", x);
}
```

7. Considere um sistema de tempo real com 4 processos periódicos com períodos de 10,20,50 e 30 e tempos de execução 5,3,4 e 1 respectivamente. Explique se o sistema é escalonável ou Não ? Explique a sua resposta.

Grupo C: Gestão de Memória (4 Valores)

8. Qual o efeito de diminuir o tamanho de página em termos de tamanho de tabela de páginas e na fragmentação interna/externa?

9. Data *Execution Prevention* (DEP) é uma funcionalidade de segurança incluída em alguns sistemas operativos cujo objetivo é o de impedir a execução de instruções vindas de certas regiões da memória duma aplicação. Explique como é que este mecanismo poderia ser implementado usando um sistema de memória paginada e dê exemplos das áreas de memoria dum processo a proteger.

10. Um sistema de memória virtual tem um tamanho de página de 32 palavras, 6 páginas virtuais e 4 páginas físicas. Um endereço virtual neste sistema tem 8 bits, sendo que os primeiros 3 bits indicam o número de página. A tabela de páginas está inicialmente no estado apresentado em baixo:

a) Indique se o resultado dos endereços lógicos indicados em baixo e referenciados por esta ordem é uma page hit (sucesso), um page miss (falha) ou trap (uma interrupção devido um erro). Calcule o endereço físico (excepto no caso dum trap) em valor decimal.
Nota : Se for uma page-miss (falha) será **<u>invocado</u>** o algoritmo de substituição de páginas (LRU-Least Recently Used / Menor usada recentemente) – quer dizer que a tabela de paginas vai mudar.

b) Mostre a tabela de páginas no fim desta sequência de endereços.

Alínea	Endereço	Hit/Miss/Trap	Endereço Físico (Decimal)
i	001 00010		
ii	011 00001		
iii	010 00100		
iv	001 10000		
v	111 01111		
vi	000 01000		
vii	100 11100		

Tabela Inicial

Página Virtual	Página Física	*Valido/ Invalido*
0		*0*
1	*2*	*1*
2	*1*	*1*
3	*0*	*1*
4		*0*
5	*3*	*1*

Tabela Final

Página Virtual	Página Física	*Valido/ Invalido*
0		
1		
2		
3		
4		
5		

Grupo C: Concorrência, Sincronização e Bloqueio (4 valores)

11. Considere o seguinte programa e responde as perguntas seguintes baixo e tomando em conta que a instrução "x = x +k" onde "x" é uma variável inteira global e k um constante em assembler é a sequencia :
movl x, %eax ; addl $k, %eax ; movl %eax, x

```
int x=0;
void *maisx(void *args)   { int add = *(int*)args ;   x = x+add; }
int main() {
   pthread_t  th[3];
   int i, ids[3]={1,2,4};
   for (i=0; i<3; i++)
       pthread_create( &th[i], NULL, maisx, &ids[i]);
   for (i=0; i<3; i++)
            pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

- Explique detalhadamente como é que o output do programa poderá ser "x=5"?
- Usando a sintaxe do Posix explique como é que pode usar um trinco de exclusão mutua para garantir que o resultado deste programa seja "x=7"
- Usando um ou mais semáforos explique como é que se pode sincronizar as threads da maneira de executar pela seguinte ordem : primeiro "x=x+1", a seguir "x=x+4" e finalmente "x=x+2";

Grupo E: Exercício Prático (4 valores)

12. Utilizando pipetas de baixo nível (pipes) é possível criar uma comunicação bidireccional entre dois processos. O objectivo deste exercício é escrever **um** programa que criará dois processos (usando fork()), um cliente e um servidor, onde o processo servidor funcionará como calculador simples. O processo cliente deverá ler (a partir do teclado) dois operandos (números reais) seguido por um operador (+,-,*, /) e deverá enviar estes valores para o servidor usando um pipe. O servidor calculará o resultado enviando depois o resultado para o cliente usando um pipe, o cliente deverá mostrar no fim o resultado no ecrã: Exemplos de execução do programa

```
$./exame          $./exame
 2.5 3 +          3.0 2 /
 5.5              1.5
```

Nota: Não é necessário verificar o sucesso ou não das chamadas ao sistema.