UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Exame, 15 de Junho de 2021, 9.30 Horas Escala 0:20 Sem Consulta Duração: 2.30H

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

Pontuação
1: 2
2: 2
3: 2
4 : 3
5 : 4
6 : 3
7 : 4

1. Explique a diferença entre "User Mode" e "Kernel Mode" (Modo de Utilizador e Modo de Kernel) e as diferenças entre as funçoes sbrk() printf(), malloc(), sqrt(), strcmp(), write() (ver folha anexo para os sintaxes)

2. Faça o diagrama temporal do processamento dos processos indicados na tabela abaixo seguindo o algoritmo de escalonamento por prioridades com preempção. Justifique as decisões tomadas.

processo	tempo de chegada	prioridade	duração
P1	1	3	4
P2	2	1	3
P3	4	2	2
P4	3	4	1
P5	1	2	2

3. Considere o programa seguinte:

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

(a) O que faz a chamada ao sistema execl()?

(b) Considere um SO cujo algoritmo de escalonamento é FCFS (first com, first served) sem preempção, Qual é o output deste programa ?

4. Um sistema de memória virtual tem um tamanho de página de 32 palavras, 8 páginas virtuais e 3 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:

Página virtual	Página física	*Valido/Invalido*
0	1	*1*
1	2	*1*
2		*0*
3		*0*
4		*0*
5		*0*
6		*0*
7	0	1

i. 111 00010
ii. 000 00011
iii. 001 01100
iv. 010 10000
v. 000 01111
vi. 011 01000
vii. 100 11100

a) Indique se o resultado dos endereços lógicos indicados em cima e referenciados por esta ordem decrescente é uma *page* hit (sucesso), uma *page* miss (falha) ou uma *trap* (uma interrupção devido um erro).
Se for uma *page*-miss (falha) será **invocado** o algoritmo de substituição de paginas (LRU-Least Recently Used / Menor usada recentemente) – quer dizer que a tabela de paginas poderá **mudar**!

b) Calcule o endereço físico resultante (exceto no caso dum trap) em valor decimal.

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

5. Responde as perguntas seguintes considerando o seguinte programa em baixo. Nota que a instrução "x = x + k" onde "x" é uma variável inteira global e k um constante em *assembler* (x86 GNU) é a sequencia de três instruções: *movl x, %eax ; addl $k, %eax ; movl %eax, x*

```
int x=0;
pthread_mutex_lock  trinco;
sem_t   S;
void *maisx(void *args)
{
    int id = *(int*)args ;
    while (0==x);
    x=x+id;
}
main()
{
   pthread_t  th[3];
   int i,ids[3]={1,3,5};
   pthread_mutex_init(&trinco,NULL);
   for (i=0; i<3; i++) pthread_create( &th[i],NULL, maisx, &ids[i]);
   getchar();
   x=1;
   for (i=0; i<3; i++) pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

a) Quais são os outputs possíveis do programa ?

b) Explique detalhadamente (faça uso dum *program trace*, pseudo-código, fluxograma etc) como é que o output poderá ser "x=7"

c) Usando a sintaxe de POSIX Threads explique como é que poderia usar a variável de exclusão mutua para garantir que o resultado deste programa seja "x=10"

d) Usando um semáforo pode-se evitar a utilização do *splinlock* na função maisx(). Usando a variável global, S, do tipo semáforo para este efeito indicar claramente quais as partes do programa que necessita de ser modificadas - Deverá indicar qual o valor de inicialização do semáforo e onde é feita, e onde as funções do sem_wait() e sem_post() são chamadas.

6. Considere um sistema que gere transferências numa "blockchain" entre entidades identificados através dum número inteiro único. Estes efetuam pedidos de transferência de valores entre múltiplas contas do mesmo sistema numa tentativa de aumentar a privacidade. Tais pedidos podem ocorrer concorrentemente, sendo cada pedido servido por uma ***thread*** diferente. Cada transferência envolve um fonte e dois destinos. Considere agora a seguinte implementação da função *transfere* (que cada thread executa para servir um pedido de transferência):

```
sem_t trincos[MAX_NUMERO_CONTAS]; (inicializados  a 1)
pthread_mutex_lock globalLock; (initializado a Null)

transfere(int de, int para1, int para2, int v1, int v2)
{
1   sem_wait( &trincos[de] )
2    pthread_mutex_lock( &globalLock )
3     sem_wait( &trincos[para1] )
4     sem_wait( &trincos[para2] )
5       saldo[de] -= (v1+v2);
6       saldo[para1]  += v1;
7       saldo[para2]  += v2;
8     sem_post( &trincos[para1] )
9     sem_post( &trincos[para2] )
10   pthread_mutex_unlock( &globalLock )
11  sem_post( &trincos[de] )
}
```

- Discute e Explique para cada cenário em baixo se o sistema possa ou não entrar em "Deadlock" com as duas transferências concorrentes. Se o sistema possa entrar em Deadlock deve Identificar os recursos naõ-prempteveis, fazer um grafo de alocação de recursos e indicar os passos necessários para Deadlock acontecer.

  Cenário1. transfere(1, 3, 4, ○, ○) **e** transfere (2, 4, 3, ○, ○) (○ É qualquer valor positivo)

  Cenário2. transfere(1, 2, 3, ○, ○) **e** transfere (3, 2, 1, ○, ○)

7. Utilizando pipetas de baixo nível (pipes) é possível criar uma comunicação bidirecional entre dois processos!

   Neste exercício escreva um programa tipo "cliente-servidor" para ajudar os milhões de pessoas que querem apostar no Euro2020. Todos querem adivinhar os resultados dos jogos portanto vamos perguntar ao Mourinho! No inicio de cada etapa o Mourinho envia um ficheiro com os seus palpites para os jogos – o ficheiro "palpites.txt". Este tem o seguinte formato : nome da equipa A nome da equipa B golos da equipa A e golos da equipa B
   Exemplo duma linha no ficheiro de texto: portugal germany 1 1

   - O seu programa deverá criar 2 processos. O processo "progénito" (*child*) será o cliente e pedirá ao utilizador do programa duas strings – os nomes das duas equipas – de máximo 30 caracteres que serão enviadas para o servidor através duma pipeta de baixo nível (pipe). O processo "progenitor" (*parent*) será o servidor. Este depois de receber os dois nomes abrirá o ficheiro fornecido pelo Mourinho, encontrará a linha respetiva e enviará o resultado do jogo para o processo cliente que imprimirá o palpite do Mourinho no ecrã. Se o utilizador introduzir um par de nomes inválidos o servidor enviará o resultado -1 -1 para o cliente.

     Exemplos dumas Execuções do Programa

     Euro 2020 – Introduza os nomes das equipas! *turkia italia* ↵
     Mourinho acha que o resultado será turkia 0 italia 3

     Euro 2020 – Introduza os nomes das equipas! *hungria portugal* ↵
     Mourinho acha que o resultado será *hungria* 1 *portugal* 1

     Ask Mourinho - Introduza os nomes das 2 equipas! *grecia malta* ↵
     Não se brinca com Mourinho!

   Notas: Pode assumir que os caracteres são sempre minúsculas. As sintaxes das funções necessárias são dadas numa folha anexa ao teste.