UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Teste de Sistemas Operativos

11/janeiro/2019

Nome: Dmytro Ostapchuk Nº mec. 84801

NOTE BEM: Justifique todos os passos das suas respostas. Respostas não justificadas não serão consideradas.

I. Considere o seguinte código:

```
1.  #include <stdio.h>
2.  #include <unistd.h>
3.
4.  int main(int argc, char *argv[])
5.  {
6.      printf("A\n");
7.      if(fork()!=0) {
8.          fork();
9.          printf("B\n");
10.         printf("C\n");
11.     } else {
12.         printf("D\n");
13.         execlp("echo", "echo", "sop", NULL);
14.         printf("E\n");
15.     }
16. }
```

a) Quantos processos, incluindo o inicial, são executados pelo programa anterior? Desenhe um diagrama que descreva a relação entre os vários processos que são executados.

b) Indique, para cada linha do programa anterior, qual(is) o (ou os) processo(s) que executam essa linha de código.

c) Apresente duas saídas (com conteúdo distinto) que podem resultar do código anterior.

d) Indique os campos principais do *Process Control Block*. Refira explicitamente quais os campos que têm valor distinto para os processos que são executados pelo programa anterior.

e) Descreva, por palavras suas, o funcionamento das primitivas **`fork()`** e **`execlp()`**.

f) Desenhe o diagrama de estados de um processo e indique, para cada linha do código anterior que o processo pai executou, qual o estado (ou estados) em que esse processo pode estar. Considere que não nada é conhecido sobre os outros processos a executar no sistema.

II. Considere que várias *threads* executam o seguinte código. O semáforo **`mutex`** é inicializado a 1, o semáforo **`barrier`** é inicializado a 0, a variável partilhada **`count`** tem o valor inicial 0 e o valor de **`n`** é 4.

```
1. codeBefore()

2. mutex.down()
3. count=count+1
4. if (count == n) barrier.up()
5. mutex.up()
6. barrier.down()
7. barrier.up()

8. codeAfter()
```

a) Considerando que 4 *threads* (T1 a T4) executam o código anterior, indique uma possíve sequência de operações, indicando claramente todas as situações de bloqueio e desbloquei Qual o valor do semáforo **`barrier`** no final?

V.S

b) O código apresentado estabelece uma barreira entre `codeBefore()` e `codeAfter()`, no entanto, esta barreira não funciona corretamente no caso de `codeBefore()` e `codeAfter()` serem executados em *loop*. Justifique a frase anterior, apresentando uma situação que demonstre que a barreira não funciona corretamente neste caso.

c) Escreva o código de uma barreira entre `codeBefore()` e `codeAfter()` que funcione corretamente mesmo que estas operações sejam executadas em *loop*.

d) Indique as diferenças principais entre as primitivas de sincronização `up()` e `down()` dos semáforos e as primitivas de sincronização `signal()` e `wait()` das variáveis de condição.

III. Considere um sistema de *spooling* de impressão, em que vários utilizadores enviam ficheiros para o servidor, que os guarda em disco (tamanho fixo). A impressão apenas se inicia quando o ficheiro estiver completamente transferido. Dependendo da forma como os utilizadores realizam as impressões e como estas são tratadas pelo servidor, existe o perigo de ocorrência de *deadlock*.

a) Apresente um exemplo de execução de 2 processos de impressão em que ocorra *deadlock*. Relacione este *deadlock* com as condições necessárias à existência de *deadlock*.

b) Proponha um algoritmo de *spooling* de impressão que garanta que o *deadlock* nunca ocorre. Relacione a sua solução com as condições necessárias à existência de *deadlock*.

IV. Considere que os processos P1 a P5 têm as caracteristicas apresentadas na tabela seguinte.

Processo	Tempo de execução	Tempo de chegada
P1	40 ms	0 ms
P2	20 ms	0 ms
P3	4 ms	10 ms
P4	10 ms	20 ms
P5	25 ms	20 ms

a) Apresente o diagrama temporal do escalonamento destes processos usando o algoritmo *Shortest Job First*, para as seguintes condições

i. Sem preempção
ii. Com preempção

CPU intensivos
Real time

b) Quais as características e garantias dadas por estes algoritmos de escalonamento?

V. O conteúdo da memória e a tabela de página de cada processo ativo de um sistema de memória virtual paginada são apresentados a seguir (de forma simplificada, considerando que cada página tem o tamanho de 1 byte e que apenas existem 2 processos)

P1

	Memória	Tabela de página
0:	A	1
1:	B	3
2:	C	5
3:	D	8

P2

	Memória	Tabela de página
0:	E	11
1:	F	4
2:	G	0
3:	H	9

a) Apresente o conteúdo da memória física entre os endereços 0 e 15.

b) Num sistema com memória virtual é possível garantir que um processo não consegue alterar o conteúdo da memória dos outros processos. Explique quais os mecanismos que garantem este nível de segurança.

c) Qual o objetivo do *Translation Lookaside Buffer*? Explique o seu funcionamento.

d) Quais os principais objetivos da Memória Virtual?