

Número:

Nome:

## LEIC/LERC – 2011/12 - 1º Exame de Sistemas Operativos – 16/Janeiro/2012

Identifique todas as folhas. Responda no enunciado, apenas no espaço fornecido.  
Justifique todas as respostas. Duração: 2h30m

### Grupo I [4v]

1. Considere esta implementação de um trinco lógico, estudada nas aulas.

```
int t1_quer_entrar = FALSE, t2_quer_entrar = FALSE;

t1_fechar() {
    t1_quer_entrar = TRUE;
    while (t2_quer_entrar == TRUE) ;
}

t1_abrir() {
    t1_quer_entrar = FALSE;
}

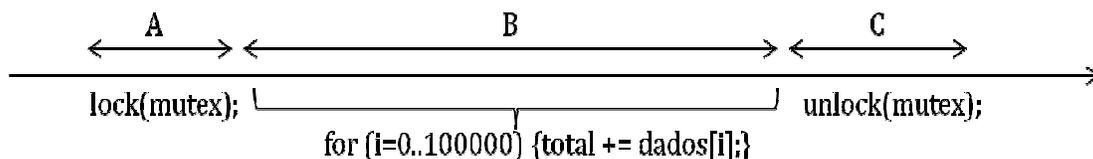
/* t2 -> simetrico */
```

- a. [0,7v] A solução sofre de espera activa? Se sim, justifique indicando a(s) linha(s) onde a espera activa ocorre. Se não, justifique.


- b. [0,7v] Porque razão a espera activa é prejudicial a um sistema multiprogramado? Justifique com um exemplo.


- c. [0,8v] Os trincos lógicos com suporte do núcleo conseguem minimizar consideravelmente os períodos de espera activa. Explique como tal é conseguido, relacionando com os estados de um processo.


2. Considere que um processo P1 está em execução num sistema operativo de tempo partilhado, com preempção, a executar o programa ilustrado de seguida:



Para cada período indicado na figura (A, B, C), descreva uma situação distinta que pode levar o processo P1 a perder o processador para outro processo.

Importante: na sua resposta, se descrever uma situação para um dos períodos, não a repita na resposta a outro período (mesmo que essa situação se aplique em ambos os períodos).

a. [0,6v] Exemplo de situação de perda de processador no Período A.


b. [0,6v] Exemplo de situação de perda de processador no Período B.


c. [0,6v] Exemplo de situação de perda de processador no Período C.


### Grupo II [4v]

Considere uma aplicação que gere um conjunto fixo de contas bancárias, e que oferece a função *transferir*, implementada da seguinte forma:

```

#define NUM_CONTAS = 100
typedef struct {
    unsigned int saldo;
    ...
} conta_t;

conta_t contas[NUM_CONTAS];

int transferir(unsigned int contaA,
              unsigned int contaB,
              unsigned int quantia)
{
    if (quantia > contas[contaA].saldo)
        return -1;

    contas[contaA].saldo -= quantia;
    contas[contaB].saldo += quantia;
    return quantia;
}
    
```

Assuma que inicialmente todas as contas têm saldo 1000.

1. [1v] Se a função *transferir* puder ser chamada concorrentemente por diferentes tarefas, o programa acima poderá levar a estados incorrectos. Descreva um exemplo de execução que ilustre a incorrecção.


2. [1v] Modifique a solução acima para que passe a ser correcta mesmo na presença de múltiplas tarefas que concorrentemente chamam a função *transferir*. Para tal, pode declarar e usar qualquer número de trincos lógicos (*mutexes*) e semáforos.

Na sua solução, assegure-se que:

- enquanto uma tarefa estiver a executar uma transferência entre as contas X e Y, nenhuma outra tarefa pode ler os saldos de X nem Y
- tarefas a trabalhar em contas distintas conseguem progredir em paralelo;

Declaração de novas variáveis:

```
int transferir(unsigned int contaA, unsigned int contaB, unsigned int quantia) {
```

3. [1v] A sua implementação da função `transferir` tem problemas de interbloqueio? Se sim, ilustre com um exemplo e proponha detalhadamente uma solução (de preferência, recorrendo a pseudo-código). Se não, justifique.

4. [1v] Estenda agora a sua solução à alínea 2 para que, no caso de não haver saldo suficiente em `contaA`, a função `transferir` espere até que o saldo em falta passe a estar disponível (assim que outra tarefa transfira a quantia suficiente para a `contaA`).

Ou seja, a função `transferir` deixará de retornar erro em caso de saldo insuficiente.

Declaração de novas variáveis:

```
int transferir(unsigned int contaA, unsigned int contaB, unsigned int quantia) {
```

**Grupo III [4v]**

1. [0.3v] Considere um disco magnético para suporte à memória secundária num sistema operativo como é o caso do Unix. No que diz respeito ao tempo de acesso aos dados no disco, diga quais as 3 operações principais que devemos ter em conta.


2. Considere um sistema de ficheiros do tipo FAT.
- a. [0.3v] Diga se concorda com a afirmação seguinte. (Atenção: responda SIM ou NÃO e justifique).  
*“Um dos problemas dos sistemas de ficheiros do tipo FAT é a elevada dimensão da tabela de alocação quando os discos têm dimensões muito grandes”.*


- b. [0.5v] Complete a Figura 1 que ilustra a organização do sistema de ficheiros do tipo FAT para a situação seguinte:
- existem 2 ficheiros no directório raiz, “F1” e “F2”;
  - F1 tem 3 blocos de dados, 1, 3 e 5;
  - F2 tem 2 blocos de dados, 2 e 4.

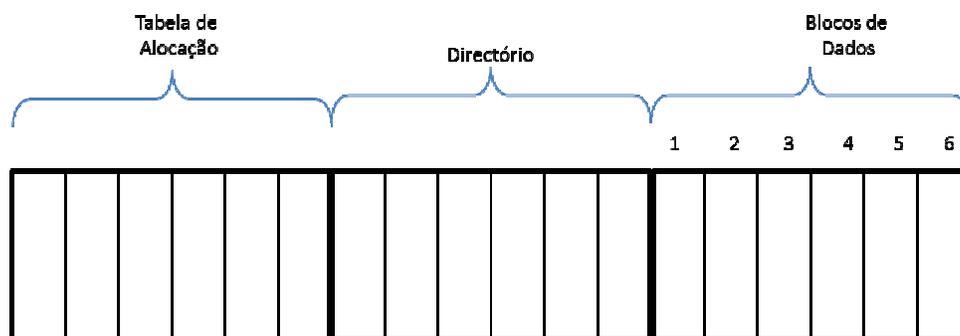


Figura 1 – Estrutura de um sistema de ficheiros do tipo FAT.

3. Considere um sistema de ficheiros no qual a dimensão máximo de um qualquer ficheiro é dado pelo fórmula seguinte (na qual B é a dimensão em bytes de um bloco de dados, e R é a dimensão em bytes de uma referência para um bloco):

$$\text{dimensão máxima} = B \times (4 + B/R + (B/R)^2)$$

- a. [0.5v] Faça uma figura que ilustre as estruturas de dados do sistema de ficheiros em causa.

--

- b. [0.4v] Diga quantos blocos são usados para guardar um ficheiro com 15 Kbytes quando  $B=4$  Kbytes e cada referência tem 4 bytes.


- c. [0.4v] Diga quantos blocos são usados para guardar um ficheiro com 8 Mbytes quando  $B=4$  Kbytes e cada referência tem 4 bytes.


4. Considere o sistema de ficheiros EXT2.

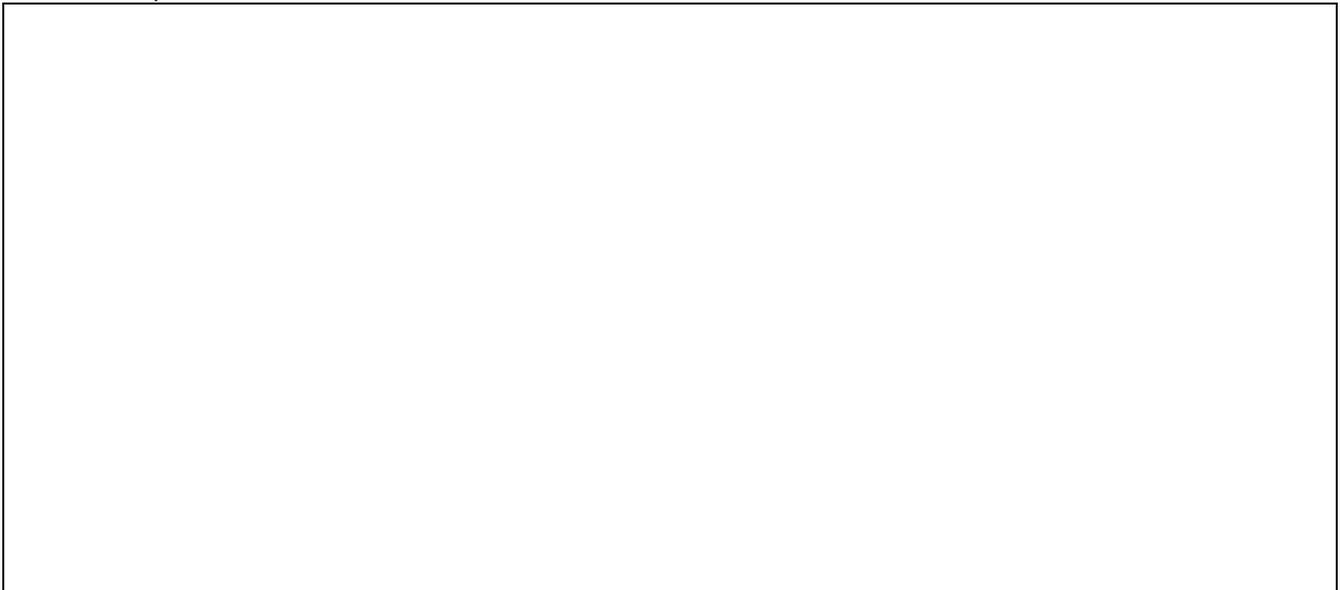
- a. [0.3v] Qual o factor limitador do número máximo de ficheiros?


- b. [0.3v] Como é que o sistema de ficheiros sabe quais as posições que estão livres e ocupadas na Tabela de Inodes de uma dada partição ?


5. Considere as tabelas de ficheiro abertos por processo, a tabela de ficheiros abertos global e a tabela de inodes no Unix/Linux.
- a. [0.5v] Apresente as estruturas de dados acima indicadas depois de um processo P1 ter aberto um ficheiro F1 em modo de escrita+leitura.



- b. [0.5v] Actualize a figura (que desenhou acima) na sequência da abertura do ficheiro F1 por um processo P2, em modo leitura.



**Grupo IV [4v]**

Um dado processo P1 executa-se **isoladamente** numa máquina com memória virtual paginada de 16 bits, com páginas de 4Kbytes.

Durante um dado período de tempo, observou-se o seguinte comportamento do programa no que diz respeito aos seus acessos à memória:

Endereço indicado pelo programa de P1	0x0002	0x0003	0x2010	0x1A10	0x2020	0x9002
Endereço acedido na memória física	0x5002	0x5003	0xB010	0x2A10	0xB020	0x5002

1. [0,5v] “Esta máquina usa endereçamento real, e não virtual”. Com base em dados da figura acima, indique uma prova de que a afirmação é falsa.


2. [0,5v] Nesta arquitectura, um endereço virtual é decomposto em duas componentes: nº da página e deslocamento. Indique quantos bits cada componente tem, justificando.


3. Considere o momento indicado pela linha a tracejado (ou seja, imediatamente após o 5º acesso ser completado).
- a. [0,7v] Indique qual o conteúdo das seguintes colunas da tabela de páginas do processo. Nota: apresente apenas as linhas que a figura permite perceber.

Índice da linha	Bit de Presença	Base

- b. [0,3v] Assumindo que não existe TLB, assinale directamente sobre a figura quais os momentos em que a tabela de páginas do processo seria lida. Indique cada momento com a sigla “TP”.
- c. [0,3v] Agora assumo que existe uma TLB com 8 entradas, que está vazia no início da execução da figura. Indique quais dos acessos marcados com “TP” na alínea anterior deixariam de acontecer. Responda marcando um círculo à volta de cada acesso evitado ( $\textcircled{\text{TP}}$ ).
4. Considere agora o 6º (e último) acesso na figura.
- a. [0,6v] Explique detalhadamente os passos que ocorrem até que este acesso seja finalmente completado, indicando: i) mudanças entre modos utilizador e núcleo que aconteçam; ii)

estruturas de dados envolvidas; iii) acessos ao disco, caso ocorram; iv) páginas substituídas, caso ocorram.


- b. [0,5v] Indique qual o conteúdo das seguintes colunas da tabela de páginas do processo. No final do 6º acesso. Mais uma vez, apresente apenas as linhas que a execução na figura permite perceber.

Índice da linha	Bit de Presença	Base

5. [0,6v] “No sistema operativo Windows, um processo só é colocado em execução se existir um número mínimo de páginas físicas que estão livres nesse momento.”  
 Esta estratégia aparenta ser prejudicial para o desempenho de alguns processos, pois pode atrasar o momento em que alguns processos são colocados em execução. Explique a vantagem desta estratégia, relacionando com a gestão de memória e o conceito de *working set* (espaço de trabalho mínimo).


**Grupo V [4v]**

1. [0.6v] A implementação do canal de comunicação entre processos pode ser feita usando duas técnicas alternativas: memória partilhada ou transferência através do núcleo do sistema operativo. Diga como compara ambas no que diz respeito à facilidade de programação e à eficiência na transferência de dados.


2. [0.4v] Considere o mecanismo de *pipes* sem nome usado para suportar a comunicação entre os processos P1 e P2. Explícite de forma clara, usando apenas as chamadas sistema *pipe*, *close*, *dup* e *fork*, o pseudo-código que é executado por cada um dos processos, de forma a que o output (stdout) de P1 fique re-direccionado para o input (stdin) de P2. Ilustre o efeito do pseudo-código nas tabelas de descritores dos processos em causa.. (Atenção: deve considerar que, na situação inicial, apenas existe o

processo P1 e que a variável *fd* está declarada como um *array* com duas posições destinadas a guardar os descritores do *pipe*, tal como indicado em baixo.)

```
int fd[2];
```

3. Diga se concorda com as frases seguintes, justificando a sua resposta. (Atenção: responda SIM ou NÃO e depois justifique).

- a. [0.4v] Os pipes são um mecanismo que permite a interligação entre os mecanismos de entradas/saídas, a comunicação entre processos, e o sistema de ficheiros.


- b. [0.4v] Os pipes sem nome são unidireccionais.


4. Diga se concorda com as frases seguintes, justificando a sua resposta. (Atenção: responda SIM ou NÃO e depois justifique).

- a. [0.4v] A utilização de *sockets* permite a adopção dos modelos de comunicação **de canal com ligação** ou **de caixas de mensagens**, dependendo do tipo de *socket* escolhido.


- b. [0.4v] Os *sockets* do tipo *stream* não suportam o modelo de interacção do tipo diálogo.


c. [0.4v] Os *sockets* do tipo *datagram* suportam o modelo de interação do tipo correio.


d. [0.4v] A primitiva *accept* tem uma semântica que suporta implementação do modelo de interação de diálogo.


5. [0.6v] Considere o mecanismo de comunicação entre dois processos baseada em *sockets stream*. Usando apenas as funções sistema necessárias e suficientes, preencha os espaços em branco dentro de cada caixa na Figura 2 que ilustra a interação entre dois processos Servidor e Cliente.

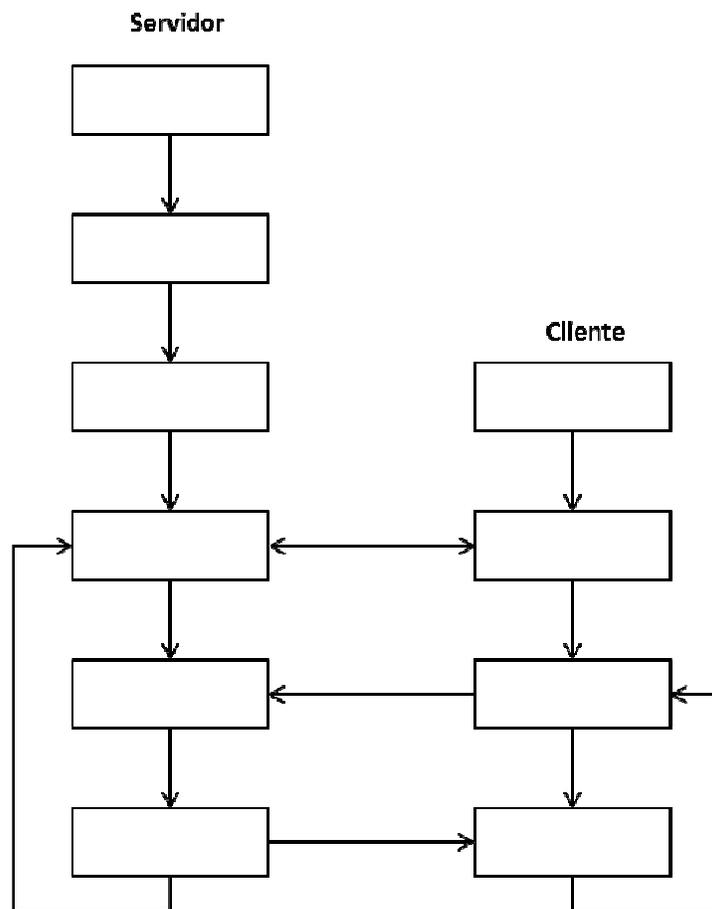


Figura 2 - Comunicação baseada em sockets stream.