Nome:		Número:	

LEIC/LERC – 2008/09

2º Exame de Sistemas Operativos

10 de Fevereiro de 2009
Duração: 2h30m

Identifique o cabeçalho de todas as folhas da prova.

O teste é resolvido no espaço dedicado ao efeito após cada pergunta.

O número de linhas reservado para o efeito não pode ser excedido, havendo lugar a uma penalização para quem responder num número de linhas superior.

Em caso de engano, poderá usar em alternativa o espaço da última página para responder à questão, devendo indicá-lo claramente, e respeitar o limite de linhas da questão.

Nas perguntas de escolha múltipla, na ausência de outra indicação, cada resposta errada desconta 1/3 da cotação.

Grupo I [1,8 valores]

1. [0,6 v.] O núcleo do sistema operativo pode ter uma organização baseada num micronúcleo.
   A. Unix não tem um micronúcleo.
   B. Um micronúcleo obriga à utilização de *hardware* dedicado no processador.
   C. A subdivisão existente no Windows designada *kernel* é o micronúcleo desse sistema operativo.
   D. A vantagem de um micronúcleo é permitir não ter de utilizar chamadas sistema.

2. [0,6 v.] Como é possível ultrapassar a barreira de protecção?
   A. Pode-se saltar a execução de nível utilizador para um endereço do código do núcleo mas o problema é conhecer esses endereços.
   B. Instalar um *device driver* que responda a comandos do utilizador.
   C. Modificar os registos da gestão da memória.
   D. Em vez de interrupções de *software* usar interrupções de *hardware* com vectores definidos pelos utilizadores.

3. [0,6 v.] Para que serve o valor passado com a interrupção de *software* (ou *trap*) utilizada nas chamadas sistema? (Por exemplo, em Intel xx86, o valor num da instrução "`INT num`")?
   A. É o endereço dentro do núcleo onde está a função.
   B. Permite endereçar uma tabela de indirecção que tem os endereços das funções do núcleo.
   C. Nas chamadas sistema não há parâmetros. É só no *hardware* que existem vectores.
   D. É para passar um parâmetro na invocação.

Número:	

Grupo II [3,2 valores]

```
int MAX = 10;  int[] tampao = new int[MAX];
int contador = 0;  int indPor = 0;  int indTirar = 0;
void enviar () {
        monitor.enter();
        while (contador == MAX) monitor.wait();
        tampao[indPor] = mensagem;
        indPor++;
        if (indPor == MAX) indPor = 0;
        contador ++;
        monitor. signalAll();
        monitor.exit();
}
void receber () {
        monitor.enter();
        while (contador == 0) monitor.wait();
        mensagem = tampao[indTirar] ;
        indTirar++;
        if (indTirar == MAX) indTirar = 0;
        contador --;
        monitor.signalAll();
        monitor.exit
}
```

Procure nas perguntas seguintes programar a função enviar e receber utilizando semáforos em vez de monitores, seguindo os passos que lhe são pedidos e preocupando-se em evidenciar a sincronização. Não se esqueça de definir a inicialização das variáveis ou objectos que introduzir

1. [0,6 v.] Resolva apenas o problema da exclusão mútua.

```
int MAX = 10;  int[] tampao = new int[MAX];
int contador = 0;  int indPor = 0;  int indTirar = 0;
```

```
void enviar () {


   tampao[indPor] = mensagem;
   indPor++;
   if (indPor == MAX) indPor = 0;
   contador ++;


}
```

```
void receber () {


   mensagem = tampao[indTirar] ;
   indTirar++;
   if (indTirar == MAX) indTirar = 0;
   contador --;


}
```

Número:	

2. [0,7 v.] Resolva a sincronização associada à condição de tampão cheio.

```
int MAX = 10;  int[] tampao = new int[MAX];
int contador = 0;  int indPor = 0;  int indTirar = 0;
```

```
void enviar () {


   tampao[indPor] = mensagem;
   indPor++;
   if (indPor == MAX) indPor = 0;
   contador ++;


}
```

```
void receber () {


   mensagem = tampao[indTirar] ;
   indTirar++;
   if (indTirar == MAX) indTirar = 0;
   contador --;


}
```

3. [0,7 v.] Resolva a sincronização associada à condição de tampão vazio.

```
int MAX = 10;  int[] tampao = new int[MAX];
int contador = 0;  int indPor = 0;  int indTirar = 0;
```

```
void enviar () {


   tampao[indPor] = mensagem;
   indPor++;
   if (indPor == MAX) indPor = 0;
   contador ++;


}
```

```
void receber () {


   mensagem = tampao[indTirar] ;
   indTirar++;
   if (indTirar == MAX) indTirar = 0;
   contador --;


}
```

4. Como sabe a existência de exclusão mútua e de funções de sincronização pode dar origem a interblocagem (*deadlock*).

4.1. [0,6 v.] Em particular, é possível combinar as soluções às alíneas anteriores dando origem a um programa que sofre de interblocagem. Apresente tal solução com interblocagem e dê um exemplo de uma execução onde aconteça interblocagem.

Número:	

4.2. [0,6 v.] Porque razão não surge este problema de interblocagem nos monitores?

Grupo III [3,5 valores]

Considere o seguinte programa, em que a chamada sistema `getppid()` retorna o pid do processo pai (do processo chamador da função).

```
ApanhaSig() {
   printf("PID %d Signal tratado \n", getpid());
   signal(SIGINT, ApanhaSig);
}

main () {
int ppid;

   signal(SIGINT, ApanhaSig);
   if (fork() ==0) {
       ppid = getppid();
       for(;;)
             if (kill(ppid, SIGINT) == -1)exit();
   }
   nice (10);
   for (;;);
}
```

1. [0,5 v.] Explique qual a função da chamada sistema `signal`.

2. [0,5 v.] Justifique por que razão a chamada aparece em duas situações no programa.

3. [0,5 v.] Considere a chamada sistema `nice (10)`. O que faz concretamente esta chamada sistema?

Número:	

4. [0,5 v.] Relacione esta chamada sistema com a fórmula da prioridade dos processos em Unix.

5. [0,5 v.] Se algum processo executar a função `nice`, que processo será?
   A. O processo pai.
   B. O processo filho.
   C. Ambos os processos.
   D. É impossível algum dos processo executar `nice` devido ao ciclo infinito.

6. [0,5 v.] `kill`:
   A. Função que permite terminar um outro processo.
   B. Só termina um processo se o signal for `SIGKILL` de outro modo é ignorada.
   C. Se o processo destinatário tratar o signal e se este não for `SIGKILL` não mata o processo.
   D. Só serve para enviar o `SIGKILL` e o `SIGINT` a outro processo.

7. [0,5 v.] Qual é o resultado deste programa?
   A. O programa coloca dois processos em execução que nunca terminam.
   B. O programa termina depois do processo pai receber o 1º signal.
   C. O programa não é determinista; depende do escalonamento.
   D. O processo pai nunca termina mas o filho pode terminar se o kill retornar um erro.

Justifique a sua opção.

Número:	

Grupo IV [4,3 valores]

1. Considere uma arquitectura elementar de gestão de memória paginada, com endereçamento virtual de 31 bits e páginas de 4 Kbytes. Considere um processo, P1, cuja tabela de páginas está presente em memória primária, começando no endereço real **0x60050000**, e que o seu conteúdo é o seguinte:

Página	Presente	Protecção	Base
0	0	RW	0x00001
1	1	R	0x00001
2	0	R	0x00001
3	1	RW	0x00002
4	1	RW	0x00000
5	0	E	0x00000

Considere um outro processo, P2, cuja tabela de páginas também está presente em memória primária, começando no endereço real **0x80000010**, e que o seu conteúdo é o seguinte:

Página	Presente	Protecção	Base
0	0	R	0x00001
1	0	R	0x00001
2	0	R	0x00001
3	0	W	0x00002
4	0	W	0x00000

1.1. [0,6 v.] Assuma que P1 está em execução e que faz os seguintes acessos às seguintes posições de memória:
Leitura da posição `0x00001A90`
Leitura da posição `0x00003400`
Escrita na posição `0x00004888`

Para cada acesso, indique o respectivo endereço real. Caso não disponha de dados suficientes para calcular algum(ns) desse(s) valor(es), responda "Indefinido".

a. Leitura da posição `0x00001A90`:	
b. Leitura da posição `0x00003400`:	
c. Escrita na posição `0x00004888`:	

1.2. [0,6 v.] Indique, para cada endereço virtual traduzido na alínea anterior, se essa tradução implicou a passagem ao modo núcleo. Justifique.

Número:	

1.3. [0,6 v.] Considere apenas os acessos para os quais conseguiu identificar, na alínea 1.1, o seu endereço real (i.e. ignore os acessos que identificou como "indefinidos").
Indique qual o estado das seguintes estruturas hardware, imediatamente após os acessos considerados anteriormente. Assuma que a TLB estava vazia antes dos acessos.

BTP:

LTP:

TLB:

2. Considere, no mesmo sistema das alíneas anteriores, a seguinte tabela de páginas de P1, estendida com os campos Idade (que tem o significado habitual de tempo desde o ultimo acesso) e Carregada (indica o tempo de permanência em memória).
Considere (apenas por motivos académicos) que a memória física do computador só tem 3 páginas úteis para o espaço de endereçamento dos processos (não se preocupe com o espaço para o sistema ou para as tabelas de páginas). Considere que se partia da situação inicial descrita na tabela abaixo e que o sistema operativo iria tratar as faltas de páginas carregando para memória as páginas em falta.

Página	Presente	Protecção	Idade	Carregada	Base
0	0	RW	0		0x00001
1	1	R	2	100	0x00001
2	0	R	0		0x00001
3	1	RW	1	200	0x00002
4	1	RW	0	50	0x00000
5	0	E	0		0x00000

2.1. [0,6 val.] Com base na tabela de páginas na tabela acima complete o preenchimento da situação inicial. Considere que só o processo cuja tabela está descrita se encontra em execução

Página Física	Página virtual	Base
0	4	0x00000
1		
2		

2.2. [0,7 val.] Preencha a evolução da memória se a política de substituição for FIFO

Página virtual acedida : **0**	Página Física	Página virtual	Base
	0		
	1		
	2		

Página virtual acedida : **1**	Página Física	Página virtual	Base
	0		

Número:	

	1		
	2		

Página virtual acedida : **5**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Página virtual acedida : **4**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Página virtual acedida : **0**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

2.3. [0,7 val.] Preencha se for LRU

Página virtual acedida : **0**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Página virtual acedida : **1**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Página virtual acedida : **5**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Página virtual acedida : **4**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

Número:	

Página virtual acedida : **0**	**Página Física**	**Página virtual**	**Base**
	0		
	1		
	2		

2.4. [0,5 val.] As diferenças indicam um comportamento diferente do algoritmo. Qual lhe parece melhor e porquê? Indique um exemplo que ilustre a vantagem com base nas alíneas anteriores.

Grupo V [3,2 valores]

1. A dimensão do bloco é um parâmetro de grande impacto num sistema de ficheiros. Nas alíneas seguintes, compare um sistema de ficheiros *ext2*, montado numa mesma partição, quando configurado com dimensões de blocos diferentes: **(A) blocos de 512 bytes**; **(B) blocos de 8 Kbytes**. (Respostas erradas descontam 1/2 da cotação.)

1.1. Indique, para cada alínea, qual das opções (**A ou B**) se aplica a cada frase, ou **NENHUMA** se nenhuma se aplicar.

a) [0,4 val.] Maior volume de meta-dados necessário para descrever igual conjunto de ficheiros.

b) [0,4 val.] Maior número de acessos ao disco para ler o conteúdo de um mesmo ficheiro.

c) [0,4 val.] Suporte a ficheiros de maior dimensão.

d) [0,4 val.] Maior fragmentação interna.

1.2. Considere os seguintes sistemas de ficheiros, assumindo igual dimensão de blocos para todos:

A – FAT16
B – ext2 (i-node com 12 entradas directas + 1 indirecta de nível-1 + 1 indirecta de nível-2 + 1 indirecta de nível-3)
C – variante de B em que i-node tem 1000 entradas exclusivamente directas

Assuma que todas as caches estão inactivas.
Indique, para cada alínea seguinte, qual/quais dos sistemas de ficheiros **(A, B, C)** em que a frase se aplica.

a) [0,4 val.] Determinar o número do 15º bloco de um ficheiro demora mais que determinar o nº do 1º bloco do mesmo ficheiro.

Número:	

b) [0,4 val.] Determinar o número do 16º bloco de um ficheiro demora mais que determinar o nº do 15º bloco do mesmo ficheiro.

c) [0,4 val.] Uma partição com 10 ficheiros pequenos (<4 blocos) ocupa menos meta-dados que uma partição com 10 ficheiros grandes (>500 blocos).

d) [0,4 val.] Tendo carregado o i-node de um ficheiro em memória, determinar o nº de um dos seus blocos é sempre possível sem qualquer acesso ao disco.

Número:	

Grupo VI [4 valores]

1. [1,8 val.] Programe um utilitário chamado *broadcast* que lê dados do stdin e escreve o que lê para os stdin de 2 comandos que lhe são passados como argumentos. Por exemplo, se se chamar "broadcast wc sort", os comandos "wc" e "sort" serão executados em processos separados e deverão estar ligados ao processo inicial por *pipes*. Complete o mais correctamente possível o seguinte esqueleto de código.

```
main(int argc, char * argv[]) {




  if (fork() == 0) {



    exec(argv[1], argv[1], NULL);
  }




  if (fork() == 0) {



    exec(argv[2], argv[2], NULL);
  }




  while (read(0,                                    ) {



    write(                                        );
    write(                                        );
  }



  close(        );
  close(        );
  exit(0);
}
```

Número:	

2. [0,6 val.] No domínio UNIX, a chamada de sistema `bind` falha se o endereço usado já existir. A função **`mkstemp`** cria um ficheiro temporário, contornando o problema da disponibilidade dos endereços. A quem pode ser útil: aos clientes, aos servidores, a nenhuns, a ambos? Justifique.

4. [0,5 val.] Relativamente à chamada de sistema `select`, assinale qual/quais das seguintes afirmações é/são verdadeiras.

☐ A. O `select` pode ser usado para saber se se pode escrever sem bloquear num descritor de ficheiro.

☐ B. O `select` nunca é bloqueante.

☐ C. O `select` nunca devolve 0.

☐ D. Uma chamada a `select` devolve um valor maior que zero. Em seguida, lê-se (`read`) dum descritor de ficheiro. Esse `read` nunca bloqueará o processo.

5. [0,6 val] Indique uma vantagem e duas desvantagens de um sistema operativo em que todos os gestores de dispositivo fossem bibliotecas de nível utilizador que acedessem directamente ao *hardware*?

6. [0,5 val.] Os subsistemas de E/S podem ser visto como estando organizados em pilha. Qual dos seguintes diagramas os representa mais correctamente?

A ☐

Aplicações
Sistema de Ficheiros
Chamadas de Sistema de E/S
Gestor de Bus
Gestor de Periférico
Hardware

B ☐

Aplicações
Chamadas de Sistema de E/S
Gestor de Bus
Sistema de Ficheiros
Gestor de Bus
Hardware

C ☐

Aplicações
Chamadas de Sistema de E/S
Sistema de Ficheiros
Gestor de Periférico
Gestor de Bus
Hardware

D ☐

Aplicações
Chamadas de Sistema de E/S
Gestor de Periférico
Sistema de Ficheiros
Gestor de Bus
Hardware