Nome:		Número:	

LEIC/LERC– 2008/09
Primeiro Teste de Sistemas Operativos

5 de Novembro de 2008
Duração: 1h30m

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, cada resposta errada desconta ¼ da cotação.
Se entregar este teste apenas poderá fazer o segundo teste na primeira época.

Grupo I [1,5 valores]

1. [0,5 valor] Assinale a afirmação que está **errada** na comparação entre um núcleo monolítico e um núcleo em camadas.

- O núcleo monolítico não tem protecção em relação a outros módulos que se executam em modo núcleo como os gestores de periféricos
- O núcleo em camadas estrutura em diversos níveis hierárquicos a protecção aos diferentes módulos do Sistema Operativo
- Um modelo em camadas pode implementar-se com base numa protecção de memória com o estado Núcleo e Utilizador
- O modelo de camadas necessita de um modelo de protecção de memória com tantos níveis quantas as camadas

2. [0,5 valor] Preencha a tabela seguinte com os **recursos lógicos** que efectuam a virtualização dos **recursos físicos**. Seja preciso nas designações.

Recursos Físicos	**Recursos Lógicos**
Processador	
Memória	
Disco	
Periféricos	
Redes de Dados	

3. [0,5 valor] Assinale a frase que é **verdadeira**. O Windows-NT tem uma subdivisão entre Kernel, Executive e Processos Sistema

- O núcleo é monolítico porque não existe nenhum mecanismo de protecção entre o Kernel, Executive e gestores de periféricos
- Trata-se de um exemplo de um micro-kernel
- A divisão entre kernel e executive obriga a que as funções do kernel sejam invocadas com chamadas sistema
- Os Processos Sistema têm acesso directo às funções do executive porque têm um nível de protecção diferente dos outros processos

Grupo II [4,5 valores]

Considere o algoritmo de Lamport (bakery) e a existência de 5 processos que o utilizam.

```
1  int senha[N];    // Inicializado a 0
2  int escolha[N]; // Inicializado a FALSE

3  Fechar (int i) {
4      int j;
5      escolha[i] = TRUE;
6      senha [i] = 1 + maxn(senha);
7      escolha[i] = FALSE;

8      for (j=0; j<N; j++) {
9         if (j==i) continue;
10        while (escolha[j]) ;
11        while (senha [j] && (senha [j] < senha [i])||
12               ((senha [i] == senha [j]) && (j < i)));
13     }
14 }

15 Abrir (int i) {senha [i] = 0;}
```

O estado do algoritmo no momento inicial está descrito nas tabelas senha e escolha.

Processos	senha	escolha
1	0	FALSE
2	1	FALSE
3	0	FALSE
4	2	FALSE
5	0	FALSE

Considere que em cada *timeslice* o processo consegue executar totalmente as funções Fechar e Abrir, a não ser que o processo entre em ciclo num while (nesse caso perderá o processador algures durante o ciclo). Com base nestas informações responda às questões seguintes preenchendo as tabelas. Note que cada alínea assume que a alínea anterior aconteceu.

1. [0,5 valor] O Processo 3 executa fechar e na instrução 6 calcula `maxn` e **perde o processador em seguida**

Processos	senha	escolha
1		
2		
3		
4		
5		

2. [0,5 valor] O Processo 4 retoma a execução na instrução 8 e executa-se até terminar o seu *timeslice*

Processos	senha	escolha
1		
2		
3		
4		
5		

Número:	

3. [0,5 valor] O Processo 5 inicia a execução de fechar e executa-se até terminar o seu *timeslice*.

Processos	senha	escolha
1		
2		
3		
4		
5		

4. [0,5 valor] O Processo 2 executa Abrir e termina o seu *timeslice*

Processos	senha	escolha
1		
2		
3		
4		
5		

5. [0,5 valor] O Processo 3 continua até terminar o seu *timeslice*

Processos	senha	escolha
1		
2		
3		
4		
5		

6. [0,5 valor] O Processo 4 executa Abrir e termina o seu *timeslice*

Processos	senha	Escolha
1		
2		
3		
4		
5		

7. [0,5 valor] Indique qual a ordem de execução dos processos 3 e 5. Justifique

8. [0,5 valor] Explique porque é que a solução anterior não introduz míngua.

9. [0,5 valor] Explique qual a principal deficiência desta solução e de que forma é resolvida nos *mutex*.

Grupo III [3 valores]

1. Considere o enunciado do problema do jantar dos filósofos. Em que cada filósofo é um processo e existe um monitor garfos que exporta duas funções: requisitar e libertar garfos.

```
filosofo(int id)
{
    while (TRUE) {
        pensar();
        garfos.requisitar (int id);
        comer();
        garfos.libertar (int id)
    }
}
```

1.1. [2 valores] Programe as funções requisitar e libertar do monitor garfos usando monitor_enter(), monitor_exit(), monitor_wait(), monitor_signal(). A solução deve evitar a interblocagem.

1.2. [1 valor] Na programação com semáforos algumas soluções apresentavam interblocagem. Analise se ela existe no seu programa e **justifique** a razão pela qual é mais fácil evitá-la com um objecto de sincronização como os monitores.

Número:	

Grupo IV [6 valores]

1. Considere um sistema operativo com escalonamento com prioridades dinâmicas, de 4 níveis (4=mais prioritário; 1=menos prioritário), com quantum fixo de 100 ms, sem preempção. Um processo que se bloqueie numa operação de E/S ou num objecto de sincronização sobe 2 níveis de prioridade; um processo que consuma o seu quantum até ao final desce um nível de prioridade.

Suponha que neste sistema estão a correr 3 processos, cujos programas são os seguintes.

```
mainP1( ) {
 while (TRUE) {
  esperar(sem1);
  doCPUwork();
  doIO();
  assinalar(sem2);
  doCPUwork();
 }
}
```

```
mainP2( ) {
 while (TRUE) {
  esperar(sem2);
  doCPUwork();
  doIO();
  assinalar(sem1);
  doCPUwork();
 }
}
```

```
mainP3( ) {
 while (TRUE) {
  doCPUwork();
 }
}
```

Assuma também que:

- A função **`doCPUwork()`** usa o CPU até ao final do seu quantum.
- A função **`doIO()`** começa por usar o CPU durante 10 ms; depois bloqueia-se à espera de uma operação E/S, durante 50 ms; e finalmente passa o processo para a lista de executáveis, com incremento de 2 níveis de prioridade (assuma que o último passo demora 0 ms).
- **`sem1 e sem2`** são semáforos partilhados entre P1 e P2, sendo que no instante inicial **`sem1=1 e sem2=0`**.
- Caso o semáforo esteja a 0, a função **`esperar`** demora 10 ms a bloquear o processo chamador; ou 0 ms caso contrário.
- A função **`assinalar`** demora 10 ms.
- A verificação da condição do **`while`** demora 0 ms.

Considere que P3 já se encontra em execução, sendo que P1 e P2 são criados com a prioridade máxima durante um quantum em que P3 se executou. Do ponto de vista do escalonador, podemos caracterizar esse quantum por:

Quantum 0 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
	P3	100	2
Funções chamadas:	DoCPUwork		
Executáveis no fim do quantum (proc.+prio.)	P1 (4); P2 (4)		
Bloqueados no fim do quantum (proc.+razão do bloqueio)	-		
sem1 no fim do quantum	1		
sem2 no fim do quantum	0		

a. [3 valores] Caracterize os 6 quantums (quer tenham sido executados completamente, quer tenham sido executados parcialmente até o processo em execução ter perdido o processador) seguintes da execução deste sistema, usando o mesmo quadro. Considere que P1 antecede P2 na fila do despacho.

Quantum 1 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
Sem1 no fim do quantum			
Sem2 no fim do quantum			

Quantum 2 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
sem1 no fim do quantum			
sem2 no fim do quantum			

Quantum 3 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
sem1 no fim do quantum			
sem2 no fim do quantum			

Quantum 4 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
sem1 no fim do quantum			
sem2 no fim do quantum			

Número:	

Quantum 5 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
sem1 no fim do quantum			
sem2 no fim do quantum			

Quantum 6 (completo ou executado parcialmente)			
Em execução	Processo	tempo de CPU (ms) consumido no quantum	Prioridade no início do quantum
Funções chamadas:			
Executáveis no fim do quantum (proc.+prio.)			
Bloqueados no fim do quantum (proc.+razão do bloqueio)			
sem1 no fim do quantum			
sem2 no fim do quantum			

b. [1 valor] Assuma que repetia a situação anterior usando uma versão do escalonador anterior que também suportava preempção. A execução anterior sofreria alterações devidas a preempção? Se não, justifique. Se sim, indique o que mudaria no primeiro quantum cuja execução fosse diferente devido a preempção.

c. [1 valor] Com o gestor de processos da alínea anterior (prioridades dinâmicas com quantum fixo + preempção), diferentes situações podem levar o despacho a comutar o processo em execução. Descreva sucintamente cada situação possível (máximo de 3 situações).

1:
2:
3:

2. [0,5 valor] O escalonamento em Unix também usa prioridades dinâmicas; no entanto, ao contrário do escalonador da alínea anterior, um processo pode ter diferentes prioridades durante a execução de um quantum. Indique o que pode gerar tal mudança de prioridade e como a nova prioridade é definida.

3. [0,5 valor] O escalonamento em Unix utiliza preempção. No entanto, é possível que um processo menos prioritário continue em execução apesar de outro processo mais prioritário estar executável. Explique porquê com um exemplo.

Grupo V [5 valores]

1. [0,5 valores] Num sistema com memória virtual segmentada, assuma que o processo P1 tem uma variável x1 do tipo (byte *) com o valor 0x00505555, que aponta para uma posição cujo valor actual é 5. Assuma também que o processo P2 tem uma variável x2 do tipo (byte *) com o mesmo valor. Se P2 escrever o valor 10 na posição apontada por x2, e P1 posteriormente ler a posição apontada por x1, que valor irá P1 ler? Justifique.

2. Considere uma arquitectura elementar de gestão de memória segmentada em que cada processo em execução ocupa um único segmento (0). Considere que só possui 16K de memória, a qual se encontra ocupada do seguinte modo (valores apresentados em base decimal):

Processo	Segmento	Base	Tamanho	Presença	Tipo de acesso
Sistema	0	0	4096	Sim	R
P1	0	6144	1024	Sim	RW
P2	0	8192	2048	Sim	R
P3	0	14336	2048	Sim	RW
P4	0	6144	2048	Não	RW

2.1. [0,5 valores] Se o processo P2 tentar ler um valor do endereço virtual 0;2256, qual o endereço físico gerado?

- ☐ 2048
- ☐ 2256
- ☐ 4304
- ☐ 10448
- ☐ É gerada uma excepção.

Número:	

2.2 [0,5 valores] Se o processo P1 tentar aceder ao endereço virtual 0;126, qual o endereço físico gerado?

- ☐ 126
- ☐ 6144
- ☐ 6270
- ☐ 1150
- ☐ É gerada uma excepção.

2.2 [0,5 valores] Se o processo P3 tentar aceder ao endereço virtual 1;122, qual o endereço físico gerado?

- ☐ 122
- ☐ 1122
- ☐ 0
- ☐ 14458
- ☐ É gerada uma excepção.

2.3 Considere que o processo P1 está em execução.

a. [0,5 valores] Indique em que componente da memória física (RAM ou mem. secundária) a Unidade de Gestão de Memória do Processador (MMU) obtém a informação das colunas "Segmento", "Base", "Tamanho", "Presença" e "Tipo de acesso" da tabela anterior.

b. [0,5 valores] Indique como é que a MMU determina o endereço físico onde essa informação existe na(s) componente(s) que referiu.

2.4 [0,5 valores] Suponha que o processo P1 pretende permitir que o processo P2 possa ter acesso de leitura ao segmento 0 de P1. Indique a alteração necessária no seguinte quadro:

Processo	Segmento	Base	Tamanho	Presença	Tipo de acesso

2.5 [0,5 valor] Aquando do primeiro acesso ao segmento 0 de P4, todo o segmento será carregado em memória primária. Indique uma possível vantagem a nível de desempenho que daí provém, relativamente à solução de carregar cada byte acedido individualmente.

3. [0,5 valor] Poderá ocorrer fragmentação externa com gestão de memória segmentada? Justifique.

4. [0,5 valor] Poderá ocorrer fragmentação externa com gestão de paginada? Justifique.
