Número: Nome:

LEIC/LERC – 2009/10
Repescagem do 1º Teste de Sistemas Operativos

2 de Fevereiro de 2010
Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas.
Duração: 1h30m

Grupo I [1,1+1,1+1,6+1,1+1,1 valores]

Considere um Sistema Operativo com um escalonamento com as seguintes características:

- Máquina mono-processador
- Multilista com 4 níveis de prioridade: 1 (menos prioritário) a 4 (mais prioritário)
- Preemptivo
- Cada nível é gerido em FIFO
- Sempre que um processo termina o seu time-slice a sua prioridade é decrementada de uma unidade.
- Sempre que um processo se bloqueia num semáforo a sua prioridade é incrementada de 1 unidade (limitada a 4).
- Com semáforos com fila de espera gerida em FIFO.
- Sincronização directa com as funções Supender e Acordar.

Lista escalonador		TS1	TS2	TS3	TS4	TS5	TS6
	Prio 4	P1					
	Prio 3	P2,	P2,				
	Prio 2						
	Prio 1	P3	P3				
Lista suspensos		nil	nil				
Semáforo S1							
	Contador	0	0				
	Lista	nil	P1				
Semáforo S2							
	Contador	1	1				
	Lista	nil	nil				

Considere que no sistema existem três processos e dois semáforos

- P1 com prioridade inicial de 4
- P2 prioridade inicial 3
- P3 prioridade inicial 1
- S1 com o valor 0 na sua variável de controlo
- S2 com o valor 1 na sua variável de controlo

Legenda
Na tabela a ordem dos processos representa a sua posição na estrutura de lista FIFO.
PE é a abreviatura para Processo em Execução.
TSx – indica um timeslice sendo a representação na tabela a situação no início do timeslice. Num timeslice os processos podem executar o seu algoritmo mas apenas considerámos as operações de sincronização relevantes. O processo que fica em execução num dado timeslice executa-o até ao fim

(caso não se bloqueie entretanto); caso se bloqueie, o timeslice é considerado concluído nesse momento e começa o próximo timeslice (com novo processo em execução).
Se um processo estiver em mais que uma lista represente-o em ambas.

Considere a evolução SEQUENCIAL do sistema descrita nas alíneas seguintes.

1. O preenchimento inicial da tabela inicial corresponde à seguinte sequência:

   - TS1 – PE executa Esperar (S1)
   - TS2 – PE executa Suspender P1 e Esperar (S2)

Preencha o estado no início de TS3.

2. Continue a evolução do sistema considerando que:

   - TS3 - PE executa Esperar (S2)

3. Se, em TS4, PE fizesse Esperar (S1) o que sucederia? Justifique em sua opinião que problema pode surgir e como poderia ser evitado.

4. Considere que PE não executa a acção da alínea anterior mas:

   - TS4 – PE executa Assinalar (S1)

5. Continue a evolução do sistema considerando que:

   - TS5 - PE executa Acordar (P1)

Grupo II [6 + 2 v]

Considere um sistema que gere contas bancárias e que serve clientes que efectuam pedidos de transferência de dinheiro entre contas. Tais pedidos podem ocorrer concorrentemente, sendo cada pedido servido por uma thread diferente. Assuma os seguintes requisitos fundamentais:
Requisito A: Transferir uma dada quantia de uma conta origem, A, para uma conta destino, B, consiste em debitar essa quantia de A, e creditar B com a mesma quantia.
Requisito B: Uma dada transferência entre uma conta A e uma conta B deve parecer atómica do ponto de vista de outras transferências que observem o saldo de A ou de B.
Requisito C: Se, concorrentemente, um cliente pedir para transferir de A para B, e outro cliente pedir para transferir de C para D (em que A, B, C e D são contas diferentes), as duas transferências devem correr completamente em paralelo (i.e. sem que nenhuma seja obrigada a esperar pela outra).
Requisito D: O sistema não deve chegar a situações de interblocagem (deadlock).

Considere as seguintes implementações da função transfere (que cada thread executa para servir cada pedido de transferência):

```
I1:
transfere(int A, int B, int quantia) {
   saldo[A] -= quantia;
   saldo[B] += quantia;
}
```

```
I2:
monitor mon;

transfere(int A, int B, int quantia) {
   mon.enter();
   saldo[A] -= quantia;
   saldo[B] += quantia;
   mon.exit();
}
```

```
I3:
monitor mon[NUM_CONTAS];

transfere(int A, int B, int quantia) {
   mon[A].enter();
   saldo[A] -= quantia;
   mon[A].exit();
   mon[B].enter();
   saldo[B] += quantia;
   mon[B].exit();
}
```

```
I4:
monitor mon[NUM_CONTAS];

transfere(int A, int B, int quantia) {
   mon[A].enter();
   mon[B].enter();
   saldo[A] -= quantia;
   saldo[B] += quantia;
   mon[A].exit();
   mon[B].exit();
}
```

Assuma que a instrução alto-nível "`A.saldo -= quantia`" corresponde à seguinte sequência de instruções máquina:

```
mov AX, A.saldo
mov BX, quantia
sub AX, BX
mov A.saldo, AX
```

E que a instrução alto-nível "`B.saldo += quantia`" corresponde à seguinte sequência de instruções máquina:

```
mov AX, B.saldo
mov BX, quantia
add AX, BX
mov B.saldo, AX
```

1. Para cada implementação, qual ou quais dos requisitos não são satisfeitos. Ilustre com um exemplo cada requisito que afirmar nao ser satisfeito.

2. Estenda a **solução I3** para que, no caso da conta de origem não ter saldo suficiente, a transferência se **bloqueie** até que tal aconteça. Mais uma vez, não utilize nem semáforos nem mutexes.

Grupo III [1,5 v. cada]

Considere uma arquitectura paginada de 32 bits, com páginas de 4KB.
Não existe qualquer tipo de cache, nem TLB.
Existem **apenas 4 páginas** de memória física disponíveis para páginas de processos. A política de substituição é FIFO.
O tempo assume-se contínuo e crescente.
Num dado momento, um processo, P1, está em execução, tendo a sua tabela de páginas o seguinte estado:

	Presente	Permissões	Base	Instante de carregamento	Modificada (dirty)
0	1	X	0x00000	1002	0
1	0	R	0x00000	700	0
2	1	RW	0x00001	2030	1
3	1	RW	0x00002	2300	0
4	1	R	0x00003	4023	0
5	0	R	0x00001	650	0
6	0	R	0x00003	800	0

1. Partindo do estado inicial da tabela de páginas que é apresentado acima, P1 efectuou a seguinte sequência de acessos. Preencha a tabela seguinte.

Acesso pedido à mem. virtual	Tab. Páginas foi consultada?	Houve mudança para modo núcleo?	Houve substituição de pág.? Se sim: (i) qual a página e (ii) teve de ser escrita em disco?	End. real calculado
1. Execução de 0x00000003				
2. Leitura de 0x00005000				
3. Execução de 0x00000004				
4. Leitura de 0x00005500				
5. Execução de 0x00000005				

2. Indique o estado da tabela de páginas de P1 após os acessos na alínea anterior.

	Presente	Permissões	Base	Instante de carregamento	Modificada (dirty)
0					
1					
2					
3					
4					
5					
6					

3. Proponha um melhoramento ao sistema que permita um desempenho melhor que o observado na alínea 1. Preencha a tabela de novo, assumindo que a mesma sequência de acessos se repete no sistema melhorado.

Melhoramento:

Acesso pedido à mem. virtual	Tab. Páginas foi consultada?	Houve mudança para modo núcleo?	Houve substituição de pág.? Se sim: (i) qual a página e (ii) teve de ser escrita em disco?	End. real calculado
1. Execução de 0x00000003				
2. Leitura de 0x00005000				
3. Execução de 0x00000004				
4. Leitura de 0x00005500				
5. Execução de 0x00000005				

4. Proponha ainda um segundo melhoramento ao sistema. Preencha a tabela de novo, assumindo apenas o melhoramento que propõe nesta alínea (i.e. na sua resposta, **ignore o melhoramento que propôs na alínea anterior**).

Melhoramento:

Acesso pedido à mem. virtual	Tab. Páginas foi consultada?	Houve mudança para modo núcleo?	Houve substituição de pág.? Se sim: (i) qual a página e (ii) teve de ser escrita em disco?	End. real calculado
1. Execução de 0x00000003				
2. Leitura de 0x00005000				
3. Execução de 0x00000004				
4. Leitura de 0x00005500				
5. Execução de 0x00000005				