UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Frequência II, 30 de Maio de 2016, 14:00Horas Escala 0:20 Sem Consulta Duração: 1h30m

- Não é permitida a consulta de livros ou de apontamentos. Por norma não se esclarecem dúvidas durante a prova. Se tiver dúvidas, indique na folha de teste a sua interpretação. Utilize uma caligrafia legível.

Grupo A: Gestão de Memória (10 Valores)

1. A esquema de alocação de memoria sujeita a fragmentação externa é "paginação", "segmentação", "partições de tamanho fixo e contíguos" ou "swapping" ? Explique a sua resposta.

2. Qual o efeito de diminuir o tamanho de página em termos de tamanho de tabela de páginas e na fragmentação interna/externa?

3. *Data Execution Prevention* (DEP) é uma funcionalidade de segurança incluída em alguns sistemas operativos cujo objetivo é o de impedir a execução de instruções vindas de certas regiões da memória duma aplicação. Explique como é que este mecanismo poderia ser implementado usando um sistema de memória paginada e dê exemplos das áreas de memoria dum processo a proteger.

4. Considere um sistema de gestão de memória com locação contígua de memória com partições múltiplas. Dadas partições de memória disponíveis de 50K, 300K, 200K e 100K (nesta ordem) como é que o SO colocará os processos com 270K, 20K, 80K,40K, 120K (nesta ordem) usando os algoritmos de (i) First-Fit e (ii) Best-Fit.

5. Um sistema de memória virtual tem um tamanho de página de 32 palavras, 8 páginas virtuais e 3 páginas físicas. Um endereço virtual neste sistema tem 8 bits, sendo que os primeiros 3 bits indicam o número de página. A tabela de páginas está inicialmente no estado apresentado em baixo:

Página virtual	Página física	*Valido/Invalido*
0	1	*1*
1	2	*1*
2		*0*
3		*0*
4		*0*
5		*0*
6		*0*
7	0	1

a) Indique se o resultado dos endereços lógicos indicados em baixo e referenciados por esta ordem decrescente é uma page hit (sucesso), um page miss (falha) ou trap (uma interrupção devido um erro).
Se for uma page-miss (falha) será **invocado** o algoritmo de substituição de paginas (LRU-Least Recently Used / Menor usada recentemente) – quer dizer que a tabela de paginas poderá **mudar**!

b) Calcule o endereço físico resultante (excepto no caso dum trap) em valor decimal.

c) Mostre a tabela de páginas no fim desta sequência de endereços.

i. 111 00010
ii. 000 00011
iii. 001 01100
iv. 010 10000
v. 000 01111
vi. 011 01000
vii. 100 11100

Grupo B: Threads, Concorrência e Sincronização (10 valores)

7. (a) Explique as condições necessárias (recursos e processos) para existir Deadlock.

   (b) Considere a seguinte situação com 3 processos (P1,P2 e P3) e 3 recursos (R1,R2 e R3) - os recursos R2 e R3 tem apenas uma instância mas o R1 tem duas instâncias do mesmo recurso.
   O processo P1 está atribuído uma das instâncias do recurso R1 e o recurso R3 e pediu e está à espera do R2. O processo P2 está atribuído o recurso R2. O processo P2 pediu e está à espera da atribuição duma instância do R1. O processo P3 está atribuído o recurso R3 e pediu e está a espera duma das instâncias do R1.

   i) Desenhe o grafo de atribuição/alocações de recursos para esta situação.

   ii) Existe uma situação de bloqueio mútua (*Deadlock*) entre os processos ou não? Justifique.

8. Responde as perguntas seguintes considerando o seguinte programa em baixo e tomando em conta que a instrução "x = x +k" onde "x" é uma variável inteira global e k um constante em assembler é a sequencia *movl x, %eax ; addl $k, %eax ; movl %eax, x*

```
int x=0;
void *maisx(void *args)   {
    int id = *(int*)args ;
    id++;
    printf("id = %d\n",id);
    x=x+id;
}
int main() {
   pthread_t  th[2];
   for (int i=0; i<2; i++)
       pthread_create( &th[i], NULL, maisx, &i);
   for (int i=0; i<2; i++)
         pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

   a) Diga o que se entende por instrução atómica.

   b) Identifique as duas condições de corrida neste programa.

   c) O programa é executado e o resultado é mostrada a direita. Explique detalhadamente (faça uso dum *program trace*, pseudo-codigo, fluxograma etc.) como é que o output neste caso poderá ter sido "x=1".

```
id = 1
id = 2
x = 1
```

   d) Sem alterando o fluxo lógico do programa – nomeadamente a criação e execução de função maisx() nas duas threads : Re-escreva na sua folha de teste o programa removendo todos as condições de corrida e, usando a sintaxe de Posix *Threads,* mostre como usando um trinco lógico de exclusão mútua podemos garantir que o resultado deste programa seja "x=3"

9. a) Uma variável do tipo semáforo poderá ser inicializada com os valores (zero, 1, ou k>1) – explique!

   b) Explique como utilizando um semáforo a espera activa é evitada e a espera limitada garantida.

   c) Considere o código para duas *threads* a executar concorrentemente :

void *a(){
puts("1a");
puts("1b")
puts("3")
puts("3mais")
}
	void *b(){
	puts("2")
	puts("2mais"
	}

   Tomando em conta que é necessário imprimir as strings contendo "1" antes das strings contendo "2" e as strings contendo "2" antes das strings contendo "3", reescreva as funções usando semáforos para sincronizar o output. Deverá usar a sintaxe POSIX e explicar como os semáforos são inicializados.