UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática

SISTEMAS OPERATIVOS

Frequência II, 1 de Junho de 2012, 14Horas Escala 0:20 Sem Consulta Duração: 1h30m

Grupo A: Gestão de Memória (10 Valores)

1. Um sistema de Gestão de Memória dum sistema Operativo tem cinco objectivos fundamentais. Descreva e explique três destes objectivos.

2. Considere a tabela de segmentos á direita. Quais são os endereços físicos dos seguintes endereços lógicos ?
   (a) 0,20 (b) 1,10 (c) 2,50 (d) 1,50

Segment	*Base*	*Length*
0	219	600
1	2300	14
2	90	100

3. Num sistema de memória virtual paginada com três molduras (frames), quantas faltas de página aconteceriam usando os algoritmos de substituição de página (i) "LRU" (*least recently used*) e (ii) "Optimal" com a seguinte string de referência: (Nota: todas as molduras estão inicialmente vazias)
   1, 3, 2, 3, 1, 1, 4, 2, 3, 5, 4, 2, 3, 4, 1.

4. Um sistema de memória virtual tem um tamanho de página de 16 palavras, 4 páginas virtuais e 3 páginas físicas. A tabela de páginas está no estado apresentado em baixo. Um endereço virtual neste sistema tem 7 bits, sendo que os primeiros 3 bits indicam o número de página. Para os seguintes endereços (a) diga se o resultado é uma page hit/miss (falha/sucesso) ou é uma interrupção/trap para o sistema operativo e (b) calcule o endereço físico quando apropriado - em valor decimal

   (a) 011 0101
   (b) 100 1001
   (c) 001 0100
   (d) 111 0011
   (e) 000 1001
   (f) 010 0001

Página virtual	Página física	*Válido/Inválido*
0	2	*1*
1		*0*
2	1	*1*
3	0	*1*

5. Explique a importância dum sistema de memória paginada dispor duma memória cache.

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

1. Considere a seguinte situação com 4 processos e 3 recursos. Ao processo P1 está atribuído os recursos R2 e R1. Ao processo P2 está atribuído o recurso R3. O processo P1 está à espera do R3. Os processos P2, P3 e P4 estão à espera do R2.

   a) Desenhe o grafo de atribuição/alocações de recursos para esta situação.
   b) Existe uma situação de bloqueio entre os processos ou não? Justifique.
   c) Explique porque uma das condições necessárias para ocorrer uma situação de deadlock é o facto que os recursos envolvidos não sejam "preemptíveis". Explique e dê dois exemplos de tais recursos.

2. A instrução "x = x +1" onde "x" é uma variável inteira global em assembler é :

```
movl    x, %eax
addl    $1, %eax
movl    %eax, x
```

   (a) Explique o que é uma instrução atómica e uma instrução divisível ?
   (b) Considere duas threads que acedem concorrentemente uma variável global x cujo valor inicial é 0 (zero) através da instrução "x = x +1" – explique, usando um "*program trace*" (fluxo de execução), como o valor final de x poderá ser apenas 1 (um).
   (c) As instruções assembler CLI e STD efectivamente ligam e desligam as (*mascarable*) interrupções. Explique como é que podem ser usados para os efeitos de exclusão mútua e as vantagens e desvantagens de tal utilização.

3. Considere o seguinte programa.

```
int x=0;
void *mais2x()   { x=x+2; }
void *mais3x3()  { x=x+3; }
int main() {
   pthread_t  th[3];
   pthread_create( &th[0], NULL, mais2x, NULL);
   pthread_create( &th[1], NULL, mais3x3, NULL);
   pthread_create( &th[2], NULL, mais3x3, NULL);
   for (int i=0; i<3; i++)
         pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

   a) Na execução deste programa com o OS Linux quantos processos e quantas threads são criados ?
   b) Quais são os outputs possíveis do programa?
   c) Usando a sintaxe de Posix explique como é que pode usar um trinco logico para garantir que o resultado deste programa seja "x=8"

4. (a) Porque é que no mecanismo de sincronização entre processos/threads através de semáforos não há Espera Activa ? Explique.
   (b) Como é que o mecanismo de semáforo pode garantir a *Progressão* e a *Espera Limitada* ?
   (c) Considere o seguinte programa que contenha o código de duas threads onde x é uma variável partilhada com valor zero.

//thread 1	//thread2
printf("estou a espera da thread 2");	fazerTrabalho();
while (0==x) ;	x=x+1;
printf("thread 2 Terminado.. Avançar"	

   (i) Explique onde está a espera activa neste programa ?
   (ii) Usando o sintaxe de Posix explique como se utiliza um semáforo para sincronizar as acções destas duas threads sem espera aciva.

Testes de Sistemas Operativos.

Rotinas (*Posix*) da biblioteca pthreads <pthreads.h> e de semáforos <semaphore.h>

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t * mutex, const pthread_mutexattr_t * attr);
sem_t    semaforo;
int  sem_init ( sem_t  * sem,  int pshared,  unsigned int initialValue);
int  sem_wait ( sem_t  *  sem);
int  sem_post ( sem_t  *  sem);
```