UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Frequência II, 6 de Junho de 2018, 14 Horas Escala 0:20 Sem Consulta Duração: 1h45m

- 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 : 5*2)

1. Descreva (sucintamente) os objetivos fundamentais dum sistema de Gestão de Memória dum Sistema Operativo.

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

3. Considere a tabela de segmentos em baixo. Quais são os endereços físicos dos seguintes endereços lógicos ?
   (a) 0,20 (b) 1,10 (c) 2,50 (d) 3,100 (e) 1,50

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

4. Memoria Virtual : Substituição de Páginas.

   Considera um sistema de memória virtual paginada com 3 molduras, preenche na sua folha de teste a tabela em baixo usando (i) o algoritmo de substituição de pagina "LRU" (Least Recently Used)" e (ii) "Optimal" com a seguinte string de referência R = 1, 2, 3, 2, 4, 1, 2, 3. Inicialmente todas as molduras estão vazias e nota que a tabela foi preenchida para as primeiras 3 valores da R. Indicar nos dois casos o número de faltas de página.

Moldura\Ref	1	2	3	2	4	1	2	3
1	1	1	1					
2		2	2					
3			3					

5. Memoria Virtual : Substituição de Páginas.

   Considera o string de referência $R=R_1,R_2,R_3,R_4$ = (1,2,3, ..,p ,p+1) (1,2,..,p-1) (p+2) (1,2,3..,p,p+1,p+2)

   Aqui R é uma sequencia de referências a paginas mas escrito como 4 secções R_1 , R_2, R_3, R_4

   O numero,#, de referencias é $\#R_1=p+1$ $\#R_2=p-1$ $\#R_3=1$ $\#R_4=p+2$ portanto o total $\#R=3p+3$

   Usando o algoritmo FIFO (First in First Out) : Calcule para cada secção Ri o número de falhas de página para os casos de (a) p+1 molduras e (b) p molduras em função de P. Calcule depois o total número de falhas em cada caso e concluir se este exemplo exibe a anomalia de Belady ou não. Inicialmente todas as molduras estão vazias.

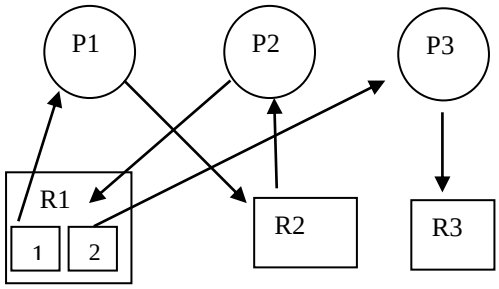
   Exemplo Ilustrativo com p=5 : R=1,2,3,4,5,6, 1,2,3,4, 7, 1,2,3,4,5,6,7 #R=18 (a) 6 molduras (b) 5 molduras

Grupo B: Threads, Concorrência e Sincronização (10 valores : 2+2+2+4)

6. Quais são as vantagens e desvantagens de usar *Multi-Threading* num servidor em vez de Multi-Processamento.

7. Explique as condições necessárias para existir uma situação de Deadlock entre um conjunto de processos e recursos.

8. Considera o grafo de atribuição/alocações de recursos em baixo onde há 3 processos e 3 tipos de recurso – um dos recursos, o R1, tem duas instancias.



   a. Explique porquê é que não existe DeadLock neste sistema.
   b. A execução de processos é um processo dinâmico ! Explique como é que esta situação poderá evoluir para uma situação de Deadlock.

9. Responde as perguntas seguintes considerando o seguinte programa em baixo. Nota que a instrução "x = x + k" onde "x" é uma variável inteira global e k um constante em *assembler* (x86 GNU) é a sequencia de três instruções: *movl x, %eax ; addl $k, %eax ; movl %eax, x*

```
int x=0;
pthread_mutex_lock  trinco;
sem_t  S;
void *maisx(void *args)   {
    int id = *(int*)args ;
    while (0==x) ;
    x=x+id;
}
main() {
   pthread_t  th[3];
   int i,ids[3]={1,3,5};
   pthread_mutex_init(&trinco,NULL);
   for (i=0; i<3; i++) pthread_create( &th[i],NULL, maisx, &ids[i]);
   getchar();
   x=1;
   for (i=0; i<3; i++) pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

   a) Quais são os outputs possíveis do programa ?
   b) Explique detalhadamente (faça uso dum *program trace*, pseudo-código, fluxograma etc) como é que o output poderá ser "x=7"
   c) Usando a sintaxe de POSIX Threads explique como é que poderia usar a variável de exclusão mutua para garantir que o resultado deste programa seja "x=10"
   d) Usando um semáforo pode-se evitar a utilização do *splinlock* na função maisx(). Usando a variável global, S, do tipo semáforo para este efeito indicar claramente quais as partes do programa que necessita de ser modificadas - Deverá indicar qual o valor de inicialização do semáforo e onde é feita, e onde as funções do sem_wait() e sem_post() são chamadas.

Variáveis de Exclusão Mutua.
int pthread_mutex_lock (pthread_mutex_t *mutex);
int pthread_mutex_unlock (pthread_mutex_t *mutex);
Semáforos
int sem_init (sem_t * sem, int pshared, unsigned int initialValue)
pshared =0 → the semaphore is shared between threads of the process;
int sem_wait (sem_t * sem); int sem_post (sem_t * sem);