UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
**SISTEMAS OPERATIVOS**

Frequência II, 28 de Maio de 2015, 18.15Horas Escala 0:20 Sem Consulta Duração: 1h30m

- Não é permitida a consulta de livros ou de apontamentos.
- 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. Especifique os cinco objetivos principais dum sistema de Gestão de Memória dum Sistema Operativo.

2. *Data Execution Prevention* (DEP) (Prevenção da Execução de Dados) é uma funcionalidade de segurança incluída nos sistemas operativos Microsoft Windows. O seu objetivo é o de impedir a execução de instruções vindas duma região da memória duma aplicação que costume ter apenas dados da aplicação (por exemplo a pilha do processo). Explique como é que este mecanismo poderia ser implementado usando um sistema de memória paginada.

3. 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 170K, 120K, 140K,80K, 50K (nesta ordem) usando os algoritmos de (i) First-Fit (Primeiro Ajuste) e (ii) Best-Fit (Melhor Ajuste).

4. Um sistema de memória virtual tem um tamanho de página de 32 palavras, 6 páginas virtuais e 4 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 | | *0* |
| 1 | 2 | *1* |
| 2 | 1 | *1* |
| 3 | 0 | *1* |
| 4 | | *0* |
| *5* | *3* | *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 vai **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. 001 00010
ii. 011 00001
iii. 010 00100
iv. 001 10000
v. 111 01111
vi. 000 01000
vii. 100 11100

5. Num sistema de memória virtual paginada, quantas faltas de página aconteceriam usando os algoritmos de substituição de página "FIFO" *(First in First Out)* com a seguinte *string* de referência:
1, 2, 3, 4, 5, 3, 2, 6, 1, 2, 3 com quatro e cinco molduras de memória física.

O algoritmo FIFO exibe ou não neste caso a Anomalia de Belady (1969 jornal do ACM) ? Explique a sua resposta.

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

7. DEADLOCK
   (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 R1 e R3 tem apenas uma instância mas o R2 tem duas instâncias do mesmo recurso.
   Ao processo P1 está atribuído uma das instâncias do recurso R2 e pediu e está à espera do R1 e R3. R1 está atribuído ao P2. O processo P2 está à espera da atribuição duma instância do R2 e o P3 está atribuído o recurso R3 e uma das instâncias do R2.

   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. THREADS & RACE CONDITIONS
   Considere o seguinte programa.

```
int x=0;
void *maisx(void *args)   { int id = *(int*)args ; x=x+id; }
int main() {
   pthread_t  th[3];
   int ids[3]={1,3,5};
   for (int i=0; i<3; i++)
       pthread_create( &th[i], NULL, maisx, &ids[i]);
   for (int 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-codigo, fluxograma etc) como é que o output poderá ser "x=4"
   c) Usando a sintaxe de Posix *Threads* explique como é que pode usar um trinco logico para garantir que o resultado deste programa seja "x=9"

9. SEMAFÓROS.
   a) Dê um exemplo (código em C/Java) que exemplifique um trinco lógico com espera activa.
   b) Explique como utilizando um semáforo a espera activa é evitada e a espera limitada garantida.
   c) Um semáforo é composto por um inteiro que necessita de ser incrementado (++)/decrementado(--) – como é que estas operações poderão ser implementadas duma maneira atómica?
   d) Considere o código para duas *threads* a executar concorrentemente

| | |
|---|---|
| void *a(){<br>puts("111\n");<br>puts("222\n");<br>puts("*cinco*\n")<br>puts("sete\n")<br>} | void *b(){<br>puts("*três*\n");<br>puts("444"\n");<br>puts("666\n")<br>puts("nove \n")<br>} |

Tomando em conta que é necessário imprimir "três" antes de "cinco" e "sete" antes do "nove" (e isto são as únicas restrições) reescreva as funções usando semáforos para sincronizar o output.
Deverá usar a sintaxe POSIX e explicar como os semáforos são inicializados.