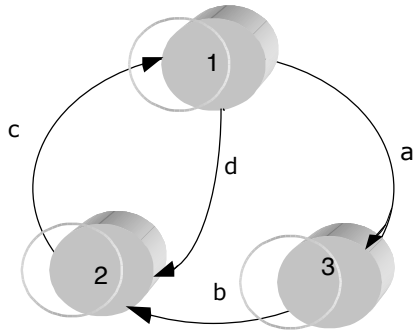
Número: Nome:

LEIC/LERC – 2010/11
1º Teste de Sistemas Operativos

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

Grupo I [6,8 v.]

1. Considere o seguinte diagrama de estados básico dos processos num sistema operativo (note que dele não consta o estado denominado *suspenso*).



a. [1 v] Escreva uma legenda para o diagrama (não se esqueça de indicar o nome dos estados bem como explicar as transições).

b. [0,8 v] No diagrama não existe uma transição entre o estado 3 e o estado 1. Explique por que motivo com base no modelo de filas de espera que suportam a execução do escalonamento.

c. [0,8 v] O diagrama não inclui o estado denominado *zombie*. Qual o objectivo deste estado?

d. [0,8 v] Explique de que estado ou estados do diagrama haverá transições para esse estado *zombie* e porquê.

2. [0,8 v] O algoritmo de escalonamento *round-robin* (sem prioridades) tem a desvantagem de conduzir a um aumento linear do tempo de execução em função dos processos executáveis. Explique a evolução para um sistema de prioridades dinâmicas (tomando como exemplo o do Unix) e esclareça por que razão um sistema com o mesmo número de processos executáveis se poderá comportar melhor.

3. A versão do Kernel do Linux 2.6 não utiliza prioridades dinâmicas no entanto o problema descrito na alínea anterior não se verifica.
    a. [0,5v] Sabendo que todas as prioridades são estáticas qual o mecanismo dinâmico que permite evitar o problema?

    b. [0,5v] De que modo é que o algoritmo dá prioridade às tarefas interactivas?

4. Considere que um dado processo, A, executa kill(pidProcessB, SIGUSER), enviando um signal a outro processo, B. Considere o tratamento desse *signal* em Unix.
    a. [0,8 v] Quais sãos as possíveis acções que o sistema pode executar aquando da recepção por parte de um processo?

    b. [0,8 v] O tratamento do signal ocorre em dois intervalos de tempo distintos, um quando o processo B não está em execução e outro quando B ganha o processador. Indique quais são e quais as operações que decorrem em cada um deles, para cada uma das acções identificadas na alínea anterior

Grupo II [6,5 v]

Considere o seguinte problema:
Numa corrida de estafetas 4x100m, existem 5 equipas em competição e um juiz de prova.
- Cada equipa é composta por 4 estafetas, numerados de 1 a 4.
- Cada estafeta corre 100m da prova da sua equipa.
- O estafeta número 1 começa no início da pista com o testemunho na mão, começando a correr assim que o juiz de prova disparar.
- Cada estafeta seguinte (2 a 4) espera que o estafeta anterior da sua equipa lhe entregue o testemunho para correr a sua parte da corrida.
- Ao chegar à meta, o estafeta nº 4 de cada equipa regista a sua chegada, acrescentando o número da sua equipa à classificação final. A classificação final é uma lista de inteiros, partilhada entre todos.
- Assim que todas as equipas cheguem à meta e registem a sua chegada na classificação, o juiz anuncia a classificação final ao público.

Nas alíneas seguintes, pretende-se programar este problema usando uma thread para cada atleta e juiz, partindo do seguinte esqueleto:

```
lista<int> classificação;

Juiz() {
       esperaPorTodosOK();
       inicializaClassificação();
       dispara();
       esperaPorTodosCompletarem();
       anunciaResultados();
}

Estafeta(int equipa, int num) {
       esperaPorMinhaVez(equipa, num);
       corre100m();
       if (num<4)
              passaTestemunho(equipa, num+1);
       else
              registaChegadaÀMeta(equipa);
}
```

1. [1,8 v] Programe as funções *esperaPorMinhaVez* e *passaTestemunho* (estafeta) e *dispara (*juiz). Caso precise de recorrer a primitivas de sincronização, use apenas semáforos e/ou mutexes.

2. [0,6 v.] A sua solução à alínea anterior tem espera activa? Justifique.

3. [1,8 v] Programe agora as funções *registaChegadaÀMeta* e *esperarPorTodosCompletarem.* Mais uma vez, use apenas semáforos e/ou mutexes.

4. [1,5 v] Programe as funções *registaChegadaÀMeta* e *esperarPorTodosCompletarem,* mas agora usando monitores (em vez de semáforos/mutexes).

5. [0,8 v] Considere e as seguintes implementações de *inicializaClassificação* e *anunciaResultados*:

```
inicializaClassificação() {
       classificação = emptyList();
}
anunciaResultados() {
       print("Classificação final:");
       for each int equipa in classificação
              print("Equipa: " + equipa);
}
```

Ambas as funções acedem à variável classificação, que é partilhada com outras threads. No entanto, o programador não usou qualquer primitiva de sincronização. É correcta esta implementação?
Se sim, justifique. Se não, indique o seria necessário modificar para corrigir as funções?

Grupo III [6,7 v]

Considere um sistema paginado com os seguintes parâmetros:
2^{32} bytes de memória física
Dimensão das páginas: 2^{12} bytes
Dimensão do espaço virtual de cada processo: 2^{16} páginas
Política de substituição NRU.

1. [0,7 v] Quantos bits tem um endereço virtual?

2. [0,7 v] Neste sistema a memória física disponível é superior ao espaço de endereçamento dos processos. Existe alguma vantagem de usar memória virtual paginada neste sistema?

3. [1,5 v] Considere a seguinte sequência de acessos feita por um dado processo assim que se começou a executar:
   0. <... Processo ganhou o processador ...>
   1. Leitura a endereço virtual 0;0, que resultou em leitura ao endereço físico 0x0224000
   2. Escrita a endereço virtual 0;4, que resultou em escrita ao endereço físico 0x0224004
   3. Leitura a endereço virtual 3;8, que resultou em leitura ao endereço físico 0x0240008
   4. <... Processo perdeu temporariamente o processador ...>
   5. Leitura a endereço virtual 0;4, que resultou em leitura ao endereço físico 0x09AC000

   Use a informação apresentada acima para construir o estado da tabela de páginas desse processo nos dois momentos indicados abaixo.
   Nota: Preencha apenas as entradas/campos que a informação apresentada abaixo lhe permita preencher. Tudo o resto deixe em branco.

   Imediatamente antes de perder o processador (passo 4)

P	Base	Prot	M	R

No final do passo 5

P	Base	Prot	M	R

4. Apresente uma explicação para os seguintes fenónemos observados neste sistema:
   a. [0,7 v] "Um dado processo alocou 2112 bytes de memória no total. No entanto, o sistema operativo indica que esse processo ocupa 4096 bytes em memória."

   b. [0,7 v] "O primeiro acesso a uma palavra de uma página marcada na tabela de páginas como "Presente" é muito mais lento que acessos a outras palavras da mesma página."

   c. [0,7 v] "Executou-se um dado programa P numa situação em que 98% da memória física estava ocupada por páginas de outros processos.
   Numa primeira experiência, os outros processos corriam programas que **practicamente só escreviam** em memória, enquanto que numa segunda experiência os outros processos **practicamente só liam** da memória. O desempenho do programa P foi muito pior na primeira experiência."

5. [1 v] Explique em que consiste o mecanismo de copy-on-write e como funciona dando um exemplo da sua utilização.

6. [0,7 v] Considere a utilização do algoritmo FIFO para a substituição de páginas. Qual a sua desvantagem principal ? Justifique.