Número: Nome:

LEIC/LERC – 2010/11
2º Exame de Sistemas Operativos

2 de Fevereiro de 2011
Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas.
Duração: 2h30m

Grupo I [2,9 valores]

1) Considere a pintura de um determinado equipamento numa fábrica.

Para tal, é necessário que que sejam executadas um número pré-determinado de tarefas (N), uma de cada vez, de acordo com uma ordem específica (e.g. aplicação de várias camadas de anti-ferrugem, isolantes, tintas, etc.). Cada tarefa é identificada por um inteiro, T.

Em cada momento, o equipamento está num dado estado, identificado por um inteiro E. Inicialmente, E tem o valor -1. Com base no estado E, a próxima tarefa a ser aplicada é a tarefa T=E+1. Cada tarefa faz com que o equipamento a ser pintado passe do estado E para o estado E+1.

Cada tarefa é executada por um operário distinto. Cada operário é especialista numa única tarefa, executando sempre e apenas essa. No entanto, existem vários operários que são especialistas na execução de uma mesma tarefa. Existe uma tabela, denominada tabela_especialistas, que para cada operário indica qual a tarefa respectiva em que ele é especialista. Cada operário alterna entre a execução da sua tarefa (respeitando a ordem correcta entre tarefas) e o descanso.

Considere as funções indicadas de seguida relativas ao enunciado acima descrito.

```
int estado = -1; // estado inicial do equipamento
int tabela_especialistas[MAX]; //  cada operário (índice da tabela) executa uma tarefa
                                  // indicada pelo conteúdo da entrada respectiva na tabela

// função que é executada por cada operário
operario (int o) {
  int time;
  for (;;) {
    time = random(); // random devolve um valor inteiro aleatório
    tenta_executar_tarefa (o, tabela_especialistas[o]);
    descansa(time); // time é o tempo de descanso do operário
  }
}

// quando um operário não está a descansar, executa esta função
tenta_executar_tarefa (int o, int t) {
  if (t = estado+1) {
    executa_tarefa_especifica(t); // operário executa tarefa
    estado = estado+1;
  }
}
```

a. [0,5 v] Descreva uma situação errónea (no que diz respeito à pintura do euipamento) que pode ocorrer caso as funções acima indicadas seja executadas tal como estão.

b. [0,5 v] Qual a causa da situação errónea que indicou na sua resposta anterior?

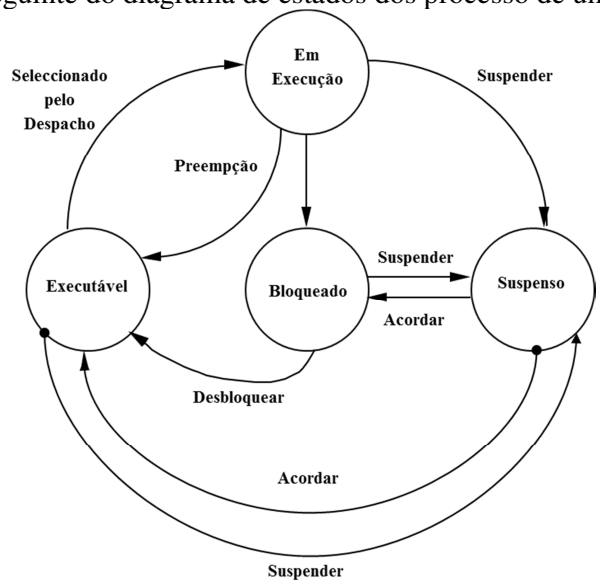
c. [0,7 v] Altere a função tenta_executar_tarefa, usando semáforos ou trincos lógicos, de modo a que a situação errónea que indicou não possa ocorrer.

d. [0,5 v] Tendo em conta a solução apresentada no enunciado, o que sucede ao operário caso este não tenha a especialização adequada ao estado do equipamento?

e. [0,7 v] Usando semáforos, altere o código da função tenta_executar_tarefa de modo a que os operários que não tenham a especialização adequada ao estado actual do equipamento se bloqueiem.

Grupo II [2,7 valores]

Considere a figura seguinte do diagrama de estados dos processo de um sistema operativo.



1) [0,5 v] Quais as transições de estado que correspondem à noção de sincronização directa ?

2) [0,7 v] Assumindo que o processo P1 está em execução, diga se é possível que durante esse intervalo de tempo possa haver um outro processo P2 que suspende P1.

3) [0,5 v] Quais os componentes no sistema operativo que estão directamente implicados nas transições entre os estados “executável” e em “execução”? Qual a sua diferença fundamental no que concerne a sua funcionalidade?

4) [0,5 v] Em que consiste a noção de preempção?

5) [0,5 v] Quais os objectos do sistema operativo que conhece que suportam o estado "bloqueado"?

Grupo III [4,4 v]

1) Considere que um processo corre o seguinte programa:

```
main() {
char array[2048];
int i;
int total = 0;

for (i=0;i<2048;i++)
  array[i]=i^2;
for (i=2047;i>0;i--)
  total += array[i] * array[i-1];
return total;
}
```

Assuma que um processo P1 corre o programa acima numa máquina com memória virtual paginada, com páginas de dimensão 512 Bytes. Assuma também que, imediatamente antes do segundo ciclo *for* iniciar, o processo perde o processador para outro processo P2 cuja execução faz com que todas as páginas de P1 em memória primária no momento em que P1 perdeu o processador sejam substituídas (*paged-out*).

Indique se cada uma das seguintes afirmações é verdadeira ou falsa, justificando.

a. [0,5v] Seria possível correr P1 em máquinas com menos de 4 páginas de memória primária.

b. [0,7v] Como P1 não faz nenhuma chamada sistema, a sua execução nunca origina a execução de código em modo núcleo.

c. [0,7v] Como cada iteração de P1 em cada ciclo nunca acede a bytes repetidos no array, o desempenho deste programa seria o mesmo se não existisse uma TLB.

d. [0,5v] Assuma que, no processo P1, o endereço virtual do byte "array[0]" é 0x00000034. É possível que o endereço virtual do mesmo byte seja exactamente o mesmo em P2.

e. [0,5v] O valor retornado por P1 poderá ser diferente do que retornaria se corresse sozinho, pois P2 pode escrever valores diferentes sobre o array de P1.

f. [0,5v] Se P1 se executasse sozinho na máquina e esta tivesse uma arquitectura de memória real, a execução de P1 poderia ser mais rápida.

g. [0,5v] Assuma que, no primeiro ciclo *for*, quando P1 escreveu no byte “array[0]”, tal causou uma escrita no endereço 0x08700104 da memória física. Quando, no segundo ciclo *for,* P1 leu do mesmo byte (“array[0]”), o endereço real acedido foi necessariamente o mesmo (0x08700104).

h. [0,5v] Assuma que o programa tinha o seguinte bug: o cabeçalho do primeiro ciclo era "*for (int i=0;i<2048000;i++)*". Este bug seria detectado, pois o processo seria terminado antes do 2º ciclo.

Grupo IV [2,8 v]

Considere a seguinte implementação da função de leitura de um sistema de ficheiros:

```
1  int fs_read(fs_t* fs, inodeid_t file, unsigned offset, unsigned count,
2     char* buffer, int* nread)
3  {
4     if (fs==NULL || file >= ITAB_SIZE || buffer==NULL || nread==NULL) {
6        return -1;
7     }
9     if (!BMAP_ISSET(fs->inode_bmap,file)) {
10       dprintf(ERROR_MESSAGE_1);
11       return -1;
12    }
14    fs_inode_t* ifile = &fs->inode_tab[file];
15    if (ifile->type != FS_FILE) {
16       dprintf(ERROR_MESSAGE_2);
17       return -1;
18    }
19    if (offset >= ifile->size) {
20       dprintf(ERROR_MESSAGE_3);
21       *nread = 0;
22       return 0;
23    }
24    // read the specified range
25    int pos = 0;
26    int iblock = offset/BLOCK_SIZE;
27    int blks_used = OFFSET_TO_BLOCKS(ifile->size);
28    int max = MIN(count,ifile->size-offset);
29    unsigned int *blk;
30    char block[BLOCK_SIZE];
31    char index_block[BLOCK_SIZE];
32    int pos1, pos2;
33    while (pos < max && iblock < blks_used) {
34       if(iblock < INODE_NUM_BLKS_LEVEL_1) {
35          pos1 = iblock;
36          block_read(fs->blocks, ifile->direct_blocks[pos1], block);
          }
37       else if(iblock < INODE_NUM_BLKS_LEVEL_2 * BLOCK_SIZE / sizeof(int)) {
38          pos1 = (iblock - INODE_NUM_BLKS_LEVEL_1) / (BLOCK_SIZE / sizeof(int));
39          block_read(fs->blocks, ifile->index_blocks[pos1], index_block);
40          pos2 = iblock - INODE_NUM_BLKS_LEVEL_1 - (BLOCK_SIZE / sizeof(int))*pos1;
41          block_read(fs->blocks, index_block[pos2], block);
42       }
43       int start = ((pos == 0)?(offset % BLOCK_SIZE):0);
44       int num = MIN(BLOCK_SIZE - start, max - pos);
45       memcpy(&buffer[pos],&block[start],num);

46       pos += num;
47       iblock++;
48    }

49    *nread = pos;
50    return 0;
51 }
```

1. [0,7v] Proponha o conteúdo das três mensagens de erro que a função fs_read pode retornar, tentando que cada mensagem seja o mais completa possível sobre o erro e a sua causa, mas sem usar mais que o espaço para a resposta.

   a. `ERROR_MESSAGE_1`

   b. `ERROR_MESSAGE_2`

   c. `ERROR_MESSAGE_3`

2. [0,7v] Qual o impacto de alterar a constante `BLOCK_SIZE` para o dobro do valor actual, em cada um destes aspectos? Justifique.
   a. Tamanho máximo do ficheiro.

   b. Fragmentação interna.

   c. Volume de meta-dados ocupados em disco (para um mesmo conteúdo do sistema de ficheiros).

3. [0,7v] De que forma é que este sistema de ficheiros organiza o conteúdo dos seus ficheiros na partição?
   (Ajuda: analise apenas as linhas 34-42.)
   a. Conteúdo de cada ficheiro está armazenado em blocos consecutivos em disco, localizados imediatamente a seguir ao descritor do ficheiro. Descritor de cada ficheiro aponta para o início do próximo ficheiro.
   b. Conteúdo de cada ficheiro está armazenado em blocos consecutivos em disco. Directório global indica, para cada ficheiro, em que bloco o ficheiro começa.
   c. Conteúdo de cada ficheiro distribuído por blocos não necessariamente consecutivos em disco. Descritor do ficheiro inclui vector de tamanho fixo em que a entrada n indica o nº do n-ésimo bloco do ficheiro.
   d. Conteúdo de cada ficheiro distribuído por blocos não necessariamente consecutivos em disco. Descritor do ficheiro inclui um vector de tamanho fixo com números dos primeiros

blocos do ficheiro, e outro vector com números de blocos dos índices (que por sua vez incluem os números dos restantes blocos do ficheiro).

e. Outra organização não listada acima.

Justifique a resposta dada na escolha múltipla com base em excertos do código apresentado. (Justificação vazia ou errada anula eventual escolha múltipla correcta.)

4. [0,7v] Pretende-se alterar o sistema de ficheiros para passar a ter uma organização FAT-16.
   a. Preencha as partes em falta na seguinte definição (em C) da variável global que conterá a tabela FAT em memória primária:

      ____ fat_table[_____];

   b. Usando a variável que declarou acima, escreva as linhas de código que substituirão as linhas 34-42 do programa acima para o sistema de ficheiros passe a usar FAT.
      Assuma que o inteiro tem 16 bits nesta arquitectura. Se necessário alterar algum(ns) campos da estrutura de dados fs_inode_t, indique claramente essa(s) alterações.

Grupo V [4,4 valores]

```
#DEFINE DIM_REGIAO 1028
/* PROGRAMA  A*/
1 main () {
2  int *Apint, IdRegPart, i;
3  IdRegPart = shmget (1234, DIM_REGIAO, 0777| IPC_CREAT);
4  if (IdRegPart<0) perror(" shmget:");
5  Apint = (int *)shmat (IdRegPart, (char *) 0, 0);
6  if (Apint == (int *) -1) perror("shmat:");
7  for (i = 0; i<257; i++) *Apint++ = i;
8  while(Apint[256]!=0) sleep(1);
9  shmctl (IdRegPart, 0, IPC_RMID,0);
10 }
```

```
#DEFINE DIM_REGIAO 1028

/* PROGRAMA B */
1 main() {
2     int *Apint, IdRegPart, i;
3     IdRegPart = shmget (1234, DIM_REGIAO, 0777);
4     if (IdRegPart <0) perror("shmget:");
5     Apint=(int*)shmat(IdRegPart, (char *)0, 0);
6     if(Apint == (int *) -1) perror("shmat:");
7     while(Apint[256]!=256) sleep(1);
8     for (i = 0; i<257; i++) printf ("%d ", *Apint++);
9     Apint[256]=0;
10}
```

Considere os programas A e B, que são executado concorrentemente no mesmo processador.

1) [0,7 v] Para que servem as linhas 5 de ambos os programas?

2) [0,9 v] Qual a diferença entre as linhas 3 e 5 de ambos os programas?

3) [0,7 v] O que significa o número "1234" na linha 3.

4) [0,7 v] Para que servem as linhas 8 e 7 dos programas A e B, respectivamente? O que sucederia se não existissem?

5) [0,7 v] As linhas 8 e 7 são pouco eficientes, porquê? Escreva o código de uma solução mais eficiente.

6) [0,7 v] O que poderia a linha 7 do programa B fosse a seguinte? Porquê?

```
7           while(Apint[100]!=100) sleep(1);
```

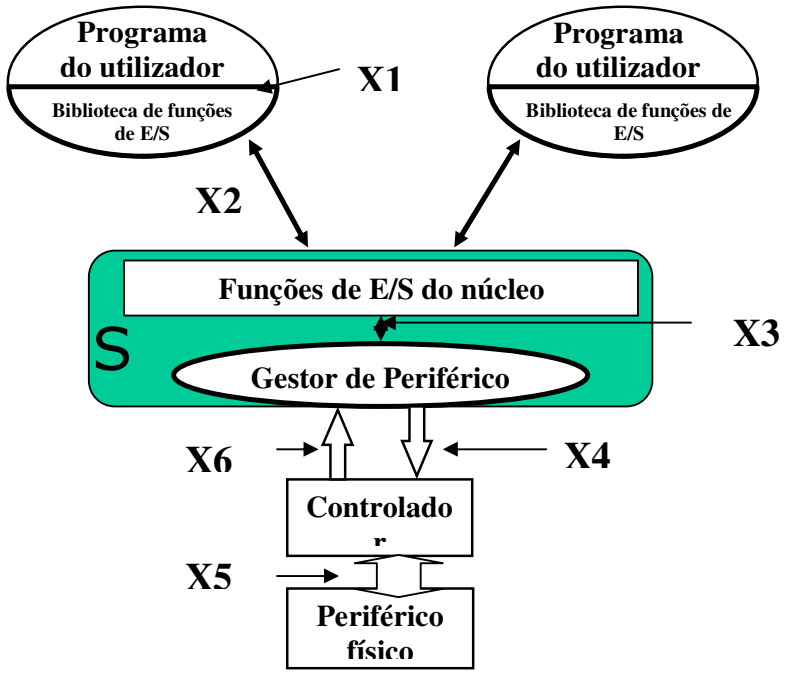
Grupo VI [2,8 v]

1) No âmbito do sistema de E/S de um sistema operativo Unix:
   a. [0,7 v] Quais as vantagens da integração dos nomes dos periféricos no espaço de nomes do sistema de ficheiros?

   b. [0,7 v] Descreva o mecanismo de abertura do periférico "`/dev/xpto`" desencadeado pela função `open("/dev/xpto",O_RDWR)`, identificando todas as estruturas do núcleo envolvidas e as relações entre elas.

   c. [0,7 v] Na execução do device driver os periféricos são identificados com base em dois números: *minor* e *major number.* Como são obtidos estes valores na sequência da execução da função referida na alínea anterior?

2) [0,7 v] Considere a figura seguinte que descreve a arquitectura de gestão de entradas/saídas do Unix. Para cada elemento das seguintes alíneas, faça corresponder a respectiva localização na figura indicando a legenda de X1 a X6 (se achar que há mais de uma hipótese deve indicá-las). Justifique a sua opção.



a. Interrupção hardware:
b. Read (fd, tampão, N):
c. Trap, Interrupção de software:
d. HAL (Hardware Abstraction Layer):
e. DMA:
f. VGA:
g. IORB:
h. USB:
