LEIC/LETI – 2016/17 – 2º Exame de Sistemas Operativos

28 de Janeiro de 2017

Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas.

Duração: 2h30

Grupo I [3,3 Val]

1. Na cadeira, estudou 2 funções para criar novos fios de execução: `fork` e `pthread_create`. Compare-as relativamente aos aspectos indicados nas alíneas seguintes.

   a. [0,8v] Em caso de sucesso, qual a primeira instrução que é executada pelo novo fio de execução?

   fork:

   pthread_create:

   b. [0,8v] O novo fio de execução: i) recebe um novo espaço de endereçamento, isolado do fio de execução original; ou ii) o novo fio de execução partilha espaço de endereçamento com o fio de execução original?

   fork:

   pthread_create:

2. Considere o seguinte excerto de um programa. O seu objectivo é implementar uma *shell* que está continuamente a ler comandos e a executá-los.

```
while(1) {
   lê_comando (command, params);
   if (execv(command, params) < 0)
      perror("Problema ao executar comando indicado, tente de novo.");
   else
      printf("Comando executado.\n");
}
```

   a. [0,7v] Ao experimentar-se o programa acima, detectou-se um problema grave: após executar o primeiro comando, o processo terminava sem permitir executar mais comandos. Explique sucintamente qual a razão deste erro.

b. [1v] Apresente uma extensão do programa acima que corrija o problema grave.
Na sua solução pode omitir tratamento de erros.

Grupo II [3,4 Val]

1. [1v] As soluções algorítmicas o problema da exclusão mútua sofrem de espera ativa? Justifique a sua resposta apresentando o pseudo-código de uma possível solução algorítmica que assegure exclusão mútua.

2. [0,7v] Os trincos lógicos com suporte do núcleo conseguem minimizar os períodos de espera activa? Justifique a sua reposta relacionando com os estados de um processo.

3. Considere uma aplicação que gere um conjunto fixo de contas bancárias, e que oferece a função *transferir*, implementada da seguinte forma:

```
#define NUM_CONTAS = 100
typedef  struct {
   unsigned int saldo;
    ...
} conta_t;

conta_t contas[NUM_CONTAS];
```

```
int transferir(unsigned int contaA,
               unsigned int contaB,
               unsigned int quantia){

  if (quantia > contas[contaA].saldo)
    return -1;

  contas[contaA].saldo -= quantia;
  contas[contaB].saldo += quantia;
  return quantia;
}
```

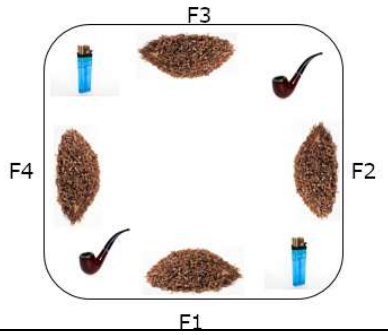
a. [0,7v] Assuma que a função *transferir* é chamada concorrentemente por diferentes fios de execução (e.g., tarefas). O programa acima poderá levar a estados incorrectos? Justifique a sua resposta apresentando um cenário detalhado.

b. [1v] Modifique a solução acima para que passe a ser correcta mesmo na presença de múltiplas tarefas que concorrentemente chamam a função *transferir*. Para tal, pode declarar e usar qualquer número de trincos lógicos (*mutexes*) e semáforos.
Na sua solução, assegure-se que:
- enquanto uma tarefa estiver a executar uma transferência entre as contas X e Y, nenhuma outra tarefa pode ler os saldos de X nem Y
- tarefas a trabalhar em contas distintas conseguem progredir em paralelo.
Nota: Não tem de solucionar eventuais problemas de interblocagem (deadlock).

Grupo III [3,4 Val]

Considere um problema de sincronização denominado "4-fumadores" que consiste no seguinte:

- 4 fumadores (F1..F4) fumam cachimbo à volta de uma mesa
- cada fumador tem o seu próprio tabaco à frente do lugar onde está sentado
- para fumar é preciso, além do tabaco, cachimbo e isqueiro; estes elementos estão dispostos como apresentado na figura ao lado
- cada fumador está num de 3 estados: DORMIR, QUER_FUMAR, FUMAR, em ciclo infinito



1. [1v] Considere a solução apresentada em seguida.

```
// um item pode ser um cachimbo ou um isqueiro
semaforo_t item[4] = {1, 1, 1, 1,};
```

```
fumador (int id) {
 while (TRUE) {
   dorme();
   esperar(item[id]);
   esperar(item[(id+1)%4]);
   fumar();
   assinalar(item[id]);
   assinalar(item[(id+1)%4]);
 }
}
```

Diga se a solução acima sofre de interblocagem (*deadlock*). Justifique detalhadamente a sua resposta.

2. [1v] Considere agora o pseudo-código da solução seguinte, que está incompleta.

```
#define DORMIR 0
#define QUER_FUMAR 1
#define FUMAR 2
#define  N 4
int estado[N] = {0, 0, 0, 0};
semaforo_t semfumador[N] = {0, 0, 0, 0};
```

```
(1) fumador(int id) {
(2)  while (TRUE) {
(3)    dormir();
(4)    estado[id] = QUER_FUMAR;
(5)    Testa(id);
(6)    esperar(semfumador [id]);
(7)    fumar();
(8)    estado[id] = DORMIR;
(9)    Testa((id-1+N)%N);
(10)   Testa((id+1)%N);
(11)   }
(12)   }
```

```
(13)   Testa(int k){
(14)     if (estado[k] == QUER_FUMAR &&
(15)     estado[(k+1)%N] != FUMAR &&
(16)     estado[(k-1)%N] != FUMAR){
(17)     estado[k] = FUMAR;
(18)     assinalar(semfumador [K]);
(19)     }
(20)   }
```

Complete a solução acima de forma a que esta fique correcta. Apresente a solução como quiser, mas de forma clara.

3. Considere de novo a solução que foi apresentada no início da questão anterior (nº 2). Justifique todas as suas respostas ilustrando com um exemplo claro em que o processo transite para o estado em causa.

   a. [0,7v] Ao executar a instrução na linha 6, um processo pode ficar no estado "bloqueado"?

   b. [0,7v] Depois de executar a instrução na linha 6, um processo pode-se encontrar no estado "executável"?

Grupo IV [3,5 Val]

1. Considere os seguintes programas:

```
int f = open(FILENAME, O_RDWR);
for (i=0..1000000) {
write(f, dados[i], sizeof(int));
}
```

```
FILE *f = fopen(FILENAME, "r+");
   for (i=0..1000000) {
   fwrite(dados[i], sizeof(int), 1, f);
}
```

   a. [0,7v] Considerando apenas a linha *open(FILENAME)*, constatou-se que esta é, em média, mais demorada quando FILENAME="/users/m/docs/f" do que quando FILENAME="/tmp/f". Apresente uma justificação para essa diferença. A sua resposta deve claramente relacionar com as estruturas persistentes que são consultadas.

b. [0,7v] Ao correr ambos os programas sobre um mesmo ficheiro, observou-se que o programa da direita é significativamente mais rápido.
Apresente uma justificação para essa diferença.

2. [0,7v] Em EXT, dado um nome de ficheiro, qual a estrutura persistente que permite determinar o número de i-node respetivo?

3. Considere os seguintes um sistema de ficheiros FAT com entradas de 16 bits (FAT-16). Assuma que que os blocos são de 2 Kbytes.
   a. [0,7v] Quantos blocos pode um ficheiro ter no máximo? Justifique.
   b. [0,7v] Aumentando o tamanho do bloco para 4 Kbytes, poder-se-ia suportar ficheiros com o dobro do tamanho. Indique uma desvantagem dessa alternativa?

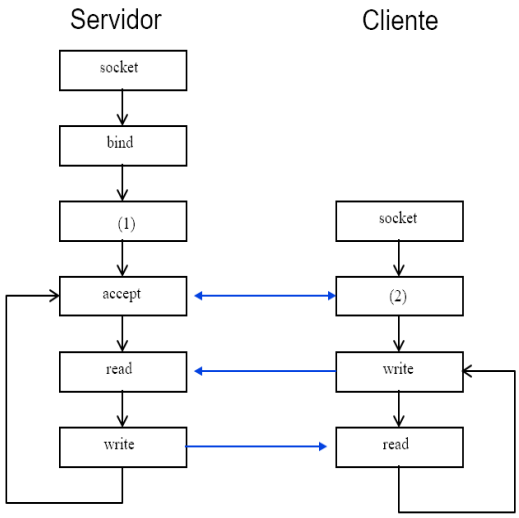
Grupo V [2,9 Val]

1. Considere os mecanismos de comunicação em Unix por: i) memória partilhada, e ii) cópia através do núcleo. Diga em qual deles se verifica o seguinte e justifique:

   a. [0,7v] Sincronização implícita.

   b. [0,7v] Programação complexa.

2. Considere agora a comunicação entre dois processos que usam *sockets* com ligação tal como descrito na figura seguinte.



   a. [0,8v] Indique: i) quais as chamadas sistema (1) e (2), e ii) se alguma delas ou ambas são bloqueantes.

b. [0,7v] Quantos sockets existem ? Justifique.

Grupo VI [3,5 Val]

Considere uma arquitetura de 64 bits, com memória virtual paginada com páginas de 4KBytes, com TLB.

1. Neste sistema, executa-se um programa constituído por um longo ciclo que analisa um grande vetor de dados. Sequencialmente, cada iteração deste programa lê uma entrada de 4 bytes do vetor e efetua alguns cálculos sobre essa entrada; a iteração seguinte faz o mesmo sobre a entrada seguinte no vetor.
   Durante a execução deste programa, o sistema foi monitorizado e detetaram-se as diferentes situações descritas nas alíneas seguintes. Para cada situação, apresente uma explicação possível para o que foi observado.

   a. [0,7v] Ao ler a primeira entrada de uma dada página, observa-se uma latência superior àquela observada todas as iterações seguintes dentro da mesma página; o programa mantém-se em modo utilizador.

   b. [0,7v] Semelhante à situação anterior, mas a latência foi muito superior e observou-se uma transição para modo núcleo.

2. Considere os campos R e M de uma entrada na tabela de páginas.
    a. [0,7v] Quando uma página é carregada em memória primária, qual o estado destes campos?

    b. [0,7v] Quando ocorre uma escrita à página em memória primária, quais os valores que são observados nos campos R e M?

    c. [0,7v] De que forma é que o bit M influencia o procedimento de substituição de uma página?