Número: Nome:

LEIC/LETI – 2015/16 - 1º Teste de Sistemas Operativos

24 de Outubro de 2015

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

Duração: 1h

Grupo I [12 Val]

Considere os dois programas p.exe e t.exe resultantes da compilação dos respectivos p.c e t.c.

```
/* ficheiro p.c */

int i;

void *xpto(void *ptr) {
  printf ("xpto: inicio\n");
  for (; i<5; i++)
    sleep(1);
  printf ("xpto: fim i=%d\n", i);
  exit(i);
}

main() {
  int a, b, c;

  i = 0;
  printf ("antes do fork\n");
  a = fork();

  if (a == 0) {
    printf ("antes de chamar xpto\n");
    xpto(NULL);
    exit (EXIT_SUCCESS);
  }
  else if (a > 0) {
      b = wait(&c);
      printf ("depois do wait (%d, %d)\n",
              b, WEXITSTATUS(c));
      exit (EXIT_SUCCESS);
}}
```

```
/* ficheiro t.c */

int i;

void *xpto(void *ptr) {
  printf ("xpto: inicio\n");
  for (; i<5; i++)
    sleep(1);
  printf ("xpto: fim i=%d\n", i);
  pthread_exit((void*) &i);
}

main() {
  int a;
  pthread_t b;
  void *c;

  i=0;
  printf ("antes do pthread_create\n");

  pthread_create(&b, NULL, xpto, NULL);

  pthread_join(b,&c);
  printf ("depois do join (%d)\n", *(int*)c);
  exit(EXIT_SUCCESS);
}
```

Nas alíneas seguintes, considere que o processo original tem pid=1234 e que os pids de novos processos criados nos exemplos acima são gerados sequencialmente a partir desses identificadores. Assuma que não ocorrem erros aquando da execução dos programas acima apresentados.

1. [2 Val] Qual o *output* gerado pela execução do p.exe? Caso existam diferentes possibilidades, apresente apenas uma delas. Para cada linha de output, indique o *pid* do processo que o originou.

pid	output

2. [1 Val] Considere a varíavel global `int i` no programa p.exe na execução que considerou na alínea anterior. Se, antes de terminar (i.e. antes de executar a instrução `exit`), o processo principal (i.e. o processo pai) ler o valor de `i`, que valor obterá? Justifique a sua resposta.

3. [2 Val] Qual o *output* gerado pela execução do t.exe numa execução sem erros? Tal como na questão acima, caso existam diferentes possibilidades, apresente apenas uma delas.

   output

4. [1 Val] Considere a varíavel global `int i` no programa t.exe. Se, antes de terminar, a tarefa inicial ler o valor de `i`, que valor obterá? Justifique a sua resposta.

5. [1 Val] Assuma que as instruções `pthread_join` e seguinte não fazem parte do código do t.c. Qual o *output* gerado pela execução do t.exe ? Tal como na questão anterior, caso existam diferentes possibilidades, apresente apenas uma delas. Justifique a sua resposta.

   output

6. [1 Val] Assumindo que o programa t.exe usa tarefas núcleo, qual dos dois programas (p.exe e t.exe) apresenta melhor desempenho? Justifique a sua resposta.

7. [1 Val] Considere agora apenas o programa p.exe. Assuma que, enquanto o processo filho se executa, o processo pai morre (enquanto se encontra na instrução `wait`) devido a um *signal* que lhe é enviado (por exemplo através da instrução *kill* efectuada pelo administrador do computador). Qual o *output* gerado pela execução do p.exe ?

pid	output

8. [1 Val] Considere agora apenas o programa t.exe. Assuma que, quando a nova tarefa está na primeira iteração do seu ciclo, o processo recebe um *signal* que causa a respetiva terminação. Qual o *output* gerado pela execução do t.exe ?

output

9. Considere a linha: `for (i=0;i<MAX_ITER; i++) sleep(1);`
Durante cada iteração deste ciclo, há pelo menos dois momentos que implicam execução em modo núcleo.

   a) [1 Val] Descreva sucintamente cada um desses momentos.

b) [1 Val] A cada momento citado na alínea anterior, que interrupção lhe deu origem? Responda "nenhuma" caso não haja nenhuma interrupção envolvida.

Grupo II [8 Val]

Considere uma garagem na qual existem N lugares de estacionamento. A garagem tem várias entradas, sendo o acesso gerido por uma aplicação central com múltiplas tarefas. Cada tarefa é responsável por interagir com um terminal em cada entrada da garagem. Cada carro com acesso ao parque é identificado por um inteiro.

1. Cada condutor que se aproxima de uma entrada executa a função *ocupar_lugar*, implementada como de seguida, para reservar um lugar antes de entrar, e chama *liberta_lugar* quando sai da garagem. O vetor lugares[N] mantém o estado de cada lugar; a cada momento, cada entrada no vetor pode conter o valor LIVRE ou o identificador do carro que tem direito a ocupar o lugar.

```
int lugares[N] = {LIVRE, ..., LIVRE}; /* todos os elementos estão LIVRES */

1. int ocupa_lugar(int carId) {
2.   int i;
3.   for (i=0; i<N; i++) {
4.     if (lugares[i] == LIVRE) {
5.       lugares[i] = carId;
6.       return i;
7.     }
8.   }
9.   return -1;
10.}

11.void liberta_lugar(int carId) {
12.  int i;
13.  for (i=0; i<N; i++) {
14.    if (lugares[i] == carId) {
15.      lugares[i] = LIVRE;
16.      return;
17.    }
18.  }
19.}
```

a) [2 Val] Após a inauguração do sistema, verificaram-se situações de conflitos entre condutores que reclamaram que, tendo acedido à garagem concorrentemente por entradas diferentes, lhes tinha sido atribuído o mesmo lugar. Analisando a implementação acima, indique a razão deste erro. Ilustre-a com um cenário concreto de execução que leve a esta situação indesejada.

b) [2 Val] Proponha uma correção ao programa. Pode escrever a sua solução diretamente sobre o código apresentado acima ou então no espaço reservado em baixo. Na sua solução pode declarar e usar objetos de sincronização como trincos (*mutexes*), semáforos ou outros.

2. Sempre que a garagem está cheia, os carros devem esperar. O pseudo-código de uma solução é apresentado em seguida na qual: i) sempre um carro pretende estacionar na garagem, invoca a função `entra`, e ii) sempre que um carro sai da garagem, invoca a função `sai`.

semaforo_t sem;	

entra(int carId) {
  esperar (sem);
  ocupa_lugar (int carId);
}
	sai(int carId) {
	  liberta_lugar (carId)
	  assinalar(sem);
	}

a) [2 Val] Qual o valor com que deve ser inicializado o semáfor `sem`? Justifique a sua resposta.

b) [2 Val] Considere agora que, sempre que um carro pretende entrar e encontra a garagem cheia, só espera caso a fila tenha menos que C carros em espera. Caso contrário, o carro não espera e a função retorna -1. Escreva a nova versão da solução apresentada acima.