Sistemas Operativos

(ano letivo 2013-14)

e-fólio B

Este enunciado constitui o elemento de avaliação designado por "e-fólio B" no âmbito da avaliação contínua e tem a cotação total de 5 valores. A sua resolução deve ser entregue até às 23h55 do dia 19 de maio pelos alunos que escolheram a modalidade de avaliação contínua.

A resolução deve ser entregue através de um único ficheiro compactado .zip, que contém unicamente os dois ficheiros seguintes:

(i) O ficheiro `mtoes.c` com o programa pedido, pronto a ser compilado;

(ii) O ficheiro `relatorio.pdf` de formato livre, com um relatório simples e sucinto com informações complementares de modo a permitir uma fácil comprensão do trabalho realizado. É desnecessário incluir uma listagem integral do código.

O nome do ficheiro .zip a entregar deve seguir a seguinte convenção para o seu nome,

"NumeroAluno-PrimeiroNome-Apelido-21111-efB.zip"

Por exemplo, um aluno com número 327555 e nome Paulo ... Costa, deverá dar o seguinte nome ao ficheiro, "327555-Paulo-Costa-21111-efB.zip"

O ficheiro deve ser única e exclusivamente entregue através do recurso "E-fólio B" disponibilizado na plataforma (Nota: apenas é visível para os alunos inscritos em avaliação contínua) alguns dias antes da data de entrega, não sendo aceites trabalhos enviados por outras vias, como por exemplo por e-mail.

Esta é uma prova de avaliação **individual** e não "um trabalho de grupo". A sua resolução deve provir unicamente do conhecimento adquirido e trabalho original desenvolvido pelo próprio aluno. Os alunos deverão saber distinguir claramente entre discutir os conteúdos abordados na unidade curricular (permitido) e discutir a resolução específica do e-fólio (não permitido).

I

Nas questões que se seguem, além de apresentar o código, deverá também comentar/explicar a sua estrutura e funcionamento, factor igualmente importante para a classificação das respostas.

1. [5] Escreva um programa multitarefa em linguagem C padrão e segundo a norma POSIX, de nome `mtoes.c` (multitask odd-even sort), que ordene por ordem ascendente um vetor de n elementos aplicando o algoritmo de ordenação odd-even sort. Para efectuar este processamento, o programa deve cumprir as seguintes especificações,

- O programa `mtoes` recebe dois argumentos na linha de comandos,

```
>> ./mtoes numtarefas dimvetor
```

- `numtarefas` é o número de tarefas que o programa deve criar para ordenar o vetor (além da principal).

- `dimvetor` é a dimensão do vetor a ordenar. O programa deve criar um vetor de inteiros (tipo int) e inicializá-lo com valores aleatórios entre 0 e 999999 com recurso à função de biblioteca rand().

- O programa deve testar se o número de argumentos dado na linha de comandos é correto e validar os argumentos para 20 ≥ numtarefas ≥ 1, dimvetor ser um número par e dimvetor ≥ 2×numtarefas+2.

- O programa deve utilizar as seguintes variáveis globais,

```
/* variaveis globais */
int *v,                 /* vetor a ordenar */
    n,                  /* comprimento vetor */
    nt,                 /* numero de tarefas */
    fase,               /* flag fase 0-par, 1-impar */
    ntc;                /* contador de tarefas para cada fase */
```

- O algoritmo odd-even sort ordena um vetor v[] de dimensão n em n passos ou fases. As n fases dividem-se em n/2 fases denominadas ímpares e n/2 fases denominadas pares, executadas alternadamente e começando numa fase ímpar.

Numa fase ímpar, o algoritmo efetua as seguintes m=n/2 comparações de elementos do vetor,
v[0] ↔ v[1], v[2] ↔ v[3], …, v[n-2] ↔ v[n-1],
e por cada comparação os respetivos elementos são trocados se estiverem fora de ordem (esta operação designa-se por operação de comparação-troca).

Numa fase par, o algoritmo efetua as seguintes m=n/2-1 comparações de elementos do vetor,
v[1] ↔ v[2], v[3] ↔ v[4], …, v[n-3] ↔ v[n-2],
e por cada comparação os respetivos elementos são trocados se estiverem fora de ordem.

- A divisão de trabalho por tarefas consiste em cada fase dividir o número de operações comparação-troca por cada tarefa. Para efetuar a divisão, aplique a seguinte estratégia:

Para dividir m itens de trabalho numerados 0,1,2,…,m-1 por nt tarefas numeradas 0,1,2,…,nt-1, resultando em aproximadamente m/nt itens por tarefa, à tarefa i correspondem os itens com início em i*m/nt e fim em (i+1)*m/nt-1 (início dos itens da próxima tarefa menos 1). É válida aqui a aritmética de inteiros.

Por exemplo, para dividir m=10 itens 0,1,2,…,9 por nt=3 tarefas, resulta para a tarefa 0 os itens {0,1,2}, para a tarefa 1 os itens {3,4,5}, e para a tarefa 2 os itens {6,7,8,9}.

Para implementação desta estratégia, quando criadas as nt tarefas devem receber respetivamente como argumento o seu número de ordem, entre 0 e nt-1.

- Cada tarefa deve ter internamente um contador do número de fases que executou e terminar quando este atingir o valor n (o vetor está ordenado).

- A variável global fase indica a cada tarefa se deve executar uma fase ímpar ou par do algoritmo. Deve ser inicializada com fase=1 indicando uma fase ímpar inicial. Uma tarefa também deve ter internamente uma variável faselocal igualmente inicializada com faselocal=1. A tarefa deve testar se a sua variável faselocal é igual à variável fase global. Se ocorre a igualdade a tarefa executa a fase correspondente do algoritmo, caso contrário deve libertar o CPU invocando a função pthread_yield() e voltar a testar a igualdade, repetindo o ciclo.

- Só deve começar a ser executada uma nova fase pelas tarefas quando todas as tarefas tiverem concluído a fase anterior. A variável global ntc serve como contador para contabilizar quantas tarefas já terminaram de executar a fase atual, sendo inicializada a 0 no início de cada fase. Quando uma tarefa termina de executar a fase atual, altera (alterna) o valor da sua variável faselocal para indicar o tipo de fase seguinte (par ou ímpar). A seguir, incrementa ntc, e se for a última (ntc=nt), é responsável por repor ntc=0 e alterar (alternar) também o valor da variável global fase para indicar o tipo de fase seguinte (par ou ímpar).

- Cada tarefa i deve imprimir uma mensagem de progresso global do trabalho de ordenação do vetor, do tipo "Tarefa[i]: progresso geral a XX%", quando o número de fases que já executou corresponder à fração (i+1)/nt do trabalho total (n fases), ou seja, igual a n*(i+1)/nt onde i é o nº ordem da tarefa e XX é 100*(i+1)/nt. Note-se que uma tarefa apenas imprime uma mensagem ao longo de todo o programa.

- Após a conclusão da ordenação (todas as tarefas criadas já terminaram), o programa deve imprimir o vetor ordenado, um elemento por linha, com seis dígitos, com zeros à esquerda se necessário, na saída padrão.

- Sugere-se o desenvolvimento deste programa em 3 fases:
1) Valida argumentos da linha de comandos, gera vetor e imprime vetor.
2) Implementa o algoritmo de ordenação sem criação de tarefas (programa monotarefa vulgar).
3) Implementa o algoritmo de ordenação num esquema multitarefa como solicitado.

Pondere quais as funções da biblioteca pthread que vai utilizar no programa e consulte as respetivas man pages para se informar dos detalhes de funcionamento de cada uma. Pondere também cuidadosamente quais os recursos e as estruturas de dados manipuladas pelas tarefas e que requeiram exclusão mútua no seu acesso para o bom funcionamento do programa.

Notas:

- O programa deve estar identificado com um cabeçalho similar ao seguinte,

```
/*
** UC: 21111 - Sistemas Operativos
** e-fólio B 2013/14 (mtoes.c)
**
** Aluno: 327555 - Paulo Costa
*/
```

- A inclusão de "print screens" da linha de comandos no relatório só deve ser feita com fundo branco.

Critérios de correcção:

- Programa não compila => 0 valores.
- código do programa não está correta e uniformemente indentado de modo a permitir a sua leitura fácil => 0 valores
- Programa não está comentado/explicado => 0 valores. Os comentários no programa elucidam questões do código locais ao comentário. A estrutura e funcionamento do programa a nível global deve ser dada no relatório. Explique (sucintamente) o como e porquê relativamente às opções que tomou para desenvolver o programa.
- Programa correcto => 5 valores.
- Programa não funciona correctamente ou não cumpre todas as especificações => de 0 a 5 valores, sendo o programa avaliado como um todo e tendo em conta a implementação das características pedidas.

Nota ética: Nunca é de mais referir que o código a apresentar como solução para este e-fólio deve ser 100% original do aluno. A probabilidade de duas pessoas que efectivamente não comunicaram entre si, apresentarem programas "quase iguais" é considerada nula. Isto é válido para qualquer par de alunos (cópia), assim como entre um aluno e qualquer outra pessoa, em particular através da Internet (cópia/plágio), onde existem inúmeras soluções e código para os mais variados problemas, em sites, fóruns, blogs, etc.
Cumpra estritamente as normas de realização individual, como se estivesse num exame com consulta, onde pode consultar a documentação mas não pode falar com ninguém.

FIM