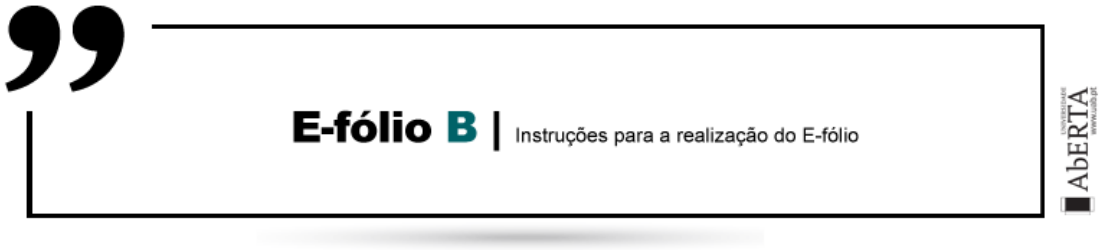
# Sistemas Operativos

(ano letivo 2020-21)




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 10 de maio pelos alunos que escolheram a modalidade de avaliação contínua.

A resolução deve ser entregue através de um único ficheiro compactado .zip, que:

(i) contém os ficheiros .c que constituem o código dos programas, prontos a serem compilados;

(ii) contém um ficheiro de nome relatorio.pdf (sem acento) com um relatório simples e sucinto com informações solicitadas e/ou complementares de modo a permitir uma fácil compreensão do trabalho realizado. É desnecessário incluir uma listagem integral do código.

(iii) 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", (sem acentos).

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), 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).

No caso de dúvidas de interpretação do enunciado, utilize o fórum de avaliação para pedidos de esclarecimento.

**I**

**1.** [5] Escreva um programa multitarefa em linguagem C padrão e segundo a norma POSIX, de nome `mthi.c`, constituído pela tarefa principal, uma tarefa designada tarefa geradora e nt tarefas designadas tarefas calculadoras, num total de nt+2 tarefas. O objetivo do programa é testar se a função de biblioteca rand() gera números aleatórios com distribuição uniforme, calculando uma tabela de n valores correspondente a um histograma com a frequência de ocorrência dos valores inteiros de 0 a n-1. A tarefa geradora enche buffers com números aleatórios e as tarefas calculadoras analisam os buffers e contam as ocorrências dos números.

- O programa mthi recebe obrigatoriamente 3 argumentos na linha de comandos,

```
>> ./mthi nt n dimbuf
```

- `nt` é o nº de tarefas calculadoras, com nt≥1.

- `n` é o nº de inteiros distintos, com $2 \leq n \leq 25$.

- `dimbuf` é a dimensão dos buffers usados para guardar os inteiros gerados, com dimbuf≥100.

- O programa mthi deve testar se o número de argumentos dado na linha de comandos é correto e se os seus valores são válidos. Em caso de erro o programa deve emitir uma mensagem e terminar.

- No início do programa, a tarefa principal (main) deve imprimir uma mensagem do tipo "Histograma de números inteiros entre 0 e n-1 com nt tarefas e buffers com dimensão dimbuf " (substituir valores).

- Para a passagem de números inteiros aleatórios entre a tarefa geradora e as tarefas calculadoras são usados 2 arrays de buffers (variáveis globais),

```
int *buf1[], /* endereços para vetores para analisar */
    *buf2[], /* endereços para vetores analisados */
    n1, n2;  /* número de endereços úteis em buf1[] e buf2[] */
```

- A dimensão dos arrays buf[] é 2*nt e contêm endereços para vetores de dimensão dimbuf, alocados dinamicamente com a função malloc(). São funcionalmente equivalentes a arrays bidimensionais buf[2*nt][dimbuf].

- Inicialmente, a tarefa geradora aloca e preenche com valores aleatórios 2*nt vetores de dimensão dimbuf. Os endereços dos vetores são guardados em buf1[] que inicialmente fica com n1=2*nt. O array buf2[] inicialmente fica vazio e n2=0.

- Ciclicamente, as tarefas calculadoras retiram de cada vez **um** endereço de buf1[], analisam o seu conteúdo e depois colocam-no em buf2[].

- Também ciclicamente, a tarefa geradora retira de cada vez **todos** os n2 endereços presentes em buf2[], preenche os vetores com novos valores aleatórios e recoloca os endereços em buf1[]. No total, a tarefa geradora deve gerar 1000*dimbuf números aleatórios, ou seja, preenche 1000 vetores.

- A tarefa geradora é a primeira a ser criada.

- Quando criadas, as nt tarefas calculadoras devem receber respetivamente como argumento o seu número de ordem, entre 0 e nt-1, e um vetor de dimensão n para devolver os seus resultados.

- Os valores aleatórios são gerados com a expressão rand()%n, resultando num valor entre 0 e n-1. A função rand() deve ser inicializada previamente com a semente 223.

- Se uma tarefa calculadora encontra o array buf1[] vazio (n1=0) e ainda não foram gerados todos os valores, deve invocar a função sched_yield() (ver man page) dando oportunidade à tarefa geradora de ser escalonada para execução, antes de tentar aceder ao array buf1[] novamente.

- Conceba uma estratégia para que as tarefas geradora e calculadoras determinem quando devem terminar.

- Imediatamente antes de terminar, cada tarefa calculadora deve imprimir uma mensagem do tipo "Tarefa (%d) analisou %d vetores" onde os %d representam respetivamente o nº de ordem da tarefa e o total de vetores que a tarefa analisou.

- Antes de terminar, a tarefa principal, que deve ser a última a terminar, deve reunir os resultados das tarefas calculadoras, imprimir a tabela final com o título "Histograma" e com a frequência total de ocorrência dos valores de 0 a n-1, um por linha, com o formato "%2d=%d\n" para o valor e para a frequência respetivamente.

- 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.

- Indique no relatório os troços de código correspondentes a regiões críticas do programa e justifique a sua existência/necessidade.

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

```
/*
** UC: 21111 - Sistemas Operativos
** e-fólio B 2020-21 (mthi.c)
**
** Aluno: 327555 - Paulo Costa
*/
```

**Critérios de correção:**

- Programa desenvolvido difere significativamente das especificações e instruções do enunciado => 0 valores.
- Programa não compila ou produz avisos (warnnings) com `gcc -Wall` => 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 => 0 valores. Os comentários no programa elucidam questões relevantes do código locais ao comentário.
- O programa em conjunto com o relatório não está estruturado, comentado ou explicado de modo à fácil compreensão da sua estrutura e funcionamento => 0 valores.
- O programa não funciona corretamente ou não cumpre todas as especificações ou é demasiado complexo => de 0 a 100% 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 efetivamente 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