

Sistemas Operativos

(ano letivo 2023-24)

”

E-fólio B | Instruções para a realização do E-fólio



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 27 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 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 `mtxor.c`, que calcule o resultado da aplicação da operação XOR a uma sequência de itens (inteiros) aleatórios aplicando uma estratégia do tipo produtor/consumidor. Para efetuar este processamento, o programa deve cumprir as seguintes especificações,

- O programa é constituído por uma tarefa produtora (tarefa principal / main) e mais nt tarefas designadas tarefas consumidoras, num total de $nt+1$ tarefas.

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

```
>> ./mtxor dimbuf N nt
```

- nt é o nº de tarefas consumidoras, com $nt \geq 1$.

- N é o nº de itens da sequência aleatória, com $N \geq 1$.

- `dimbuf` é a dimensão do (único) buffer onde as tarefas produtoras colocam itens e as tarefas consumidoras retiram itens.

- O programa `mtxor` 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 deve imprimir a mensagem "Cálculo XOR de sequência com N itens, nt tarefas e buffer `dimbuf` itens" (substituir valores).

- A tarefa produtora (main) gera itens de valores aleatórios do tipo inteiro utilizando a função `rand()` previamente inicializada com a semente 737. Ciclicamente, cada vez que acede ao buffer enche o buffer até à sua capacidade máxima, repetidamente até ter gerado e inserido um total de N itens. Inicialmente, antes de criar as tarefas consumidoras, deve encher previamente o buffer.

- Ciclicamente, as tarefas consumidoras acedem ao buffer e de cada vez retiram um bloco de até `dimbuf/nt` itens. Cada tarefa deve dispor de uma variável inicializada a 0 com a qual calcula e armazena sucessivamente o XOR dessa variável com cada item retirado. Deve também dispor de um contador para armazenar o n.º de operações XOR que realizou.

- Quando criadas, as nt tarefas consumidoras devem receber como argumento o seu número de ordem, entre 0 e $nt-1$.

- Imediatamente antes de terminar, cada tarefa consumidora deve imprimir uma mensagem com o formato "T%d: %d operações realizadas" com o nº de ordem da tarefa e o total de operações que a tarefa realizou.

- Antes de terminar, a tarefa principal, que deve ser a última a terminar, deve imprimir uma mensagem com o número total de operações efetuadas (obtido pela soma do n.º operações de cada tarefa) e o resultado final constituído pelo XOR dos resultados parciais de cada tarefa.

- Para implementação do acesso da tarefa produtora ao buffer, utilize o seguinte algoritmo do tipo espera ativa mas com libertação do CPU caso o buffer esteja cheio:

```
// pseudo-código acesso do produtor ao buffer com espera ativa
// e libertação do CPU
buffer buf;
mutex mtx_buf;
// produtor
mutex_lock(&mtx_buf);
if buf.count == dim_buf
    // buffer cheio
    mutex_unlock(&mtx_buf);
    sched_yield(); // libertar CPU
else
    inserir(&buf,item);
    mutex_unlock(&mtx_buf);
end
```

onde a função sched_yield() liberta voluntariamente o CPU e tem o protótipo,

```
#include <sched.h>
int sched_yield(void);
```

- Projete e inclua no relatório o pseudo-código semelhante para o acesso de uma tarefa consumidora ao buffer.

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

- 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.

- Este e-fólio baseia-se nos conhecimentos adquiridos no capítulo 2 do livro recomendado MOS3e sobre tarefas e nos conteúdos do LAB3. Não é permitido utilizar elementos da biblioteca pthread significativamente mais avançados que os descritos no LAB3, como por exemplo variáveis de condição, R/W locks, etc, assim como semáforos e outros elementos de sincronização.

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

```
/*
** UC: 21111 - Sistemas Operativos
** e-fólio B 2023-24 (mtxor.c)
**
** Aluno: 327555 - Paulo Costa
**
*/
```

Critérios de correção:

- O programa desenvolvido difere significativamente das especificações e instruções do enunciado => 0 valores.
- O programa não compila ou produz avisos (warnings) com gcc -Wall => 0 valores.
- O código do programa não está correta e uniformemente indentado de modo a permitir a sua leitura fácil => 0 valores
- O programa não está comentado => 0 valores. Os comentários no programa devem elucidar questões relevantes do código locais ao comentário e não o funcionamento geral do programa.
- O formato do relatório é livre, com um máximo de 4 páginas, letra 12, incluindo capa. Este deve explicar a estrutura e funcionamento geral do programa de modo à sua fácil compreensão, referindo os pontos principais e opções tomadas do programa que desenvolveu. A utilidade do relatório para a compreensão do trabalho realizado também é avaliada.
- 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 em conjunto com o relatório 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, IA, etc.

FIM