

# Projeto de Sistemas Operativos

## 2021-22

### Enunciado do 1º exercício

#### LEIC-A/LEIC-T/LETI

## Projeto base (ponto de partida)

O projeto base é o TecnicoFS, um sistema de ficheiros simplificado em modo utilizador. É implementado como uma biblioteca, que pode ser usada por qualquer processo cliente que pretenda ter uma instância privada de um sistema de ficheiros no qual pode manter os seus dados.

### Interface de programação

O TecnicoFS oferece uma interface de programação (API) inspirada na API de sistema de ficheiros POSIX. No entanto, para simplificar o projeto, a API do TecnicoFS oferece apenas um subconjunto de funções com interface simplificada. São elas<sup>1</sup>:

- `int tfs_open(const char *name, int flags);`
- `int tfs_close(int fhandle);`
- `ssize_t tfs_write(int fhandle, const void *buffer, size_t len);`
- `ssize_t tfs_read(int fhandle, void *buffer, size_t len);`

Além destas funções, existem as funções de inicialização e destruição do sistema de ficheiros, `tfs_init` e `tfs_destroy`.

O código fonte do TecnicoFS encontra-se disponibilizado na página web da disciplina. A descrição detalhada de cada função pode ser encontrada na documentação no código fonte do TecnicoFS.

### Estado do sistema de ficheiros

Tal como em FS tradicionais modernos, o conteúdo do FS encontra-se referenciado numa estrutura de dados principal chamada tabela de *i-nodes*, global ao FS. Cada *i-node* representa uma diretoria ou um ficheiro no TecnicoFS, que tem um identificador único chamado *i-number*. O *i-number* de uma diretoria/ficheiro corresponde ao índice do *i-node*

---

<sup>1</sup> Nota: o tipo de dados `ssize_t` é definido no *standard* POSIX para representar tamanhos (em bytes), podendo também ter o valor -1 para representar erro. É, por exemplo, o tipo do retorno das funções *read* e *write* da API de sistema de ficheiros POSIX.

correspondente na tabela de *i-nodes*. O *i-node* consiste numa estrutura de dados que descreve os atributos da diretoria/ficheiro (aquilo que normalmente se chamam os seus *metadados*) e que referencia o conteúdo da diretoria/ficheiro (ou seja, os *dados*).

Além da tabela de *i-nodes*, existe uma região de dados, organizada em blocos de tamanho fixo. Esta região mantém os dados de todos os ficheiros do FS, sendo esses dados referenciados a partir do *i-node* de cada ficheiro (na tabela de *i-nodes*). No caso de ficheiros normais, é na região de dados que é mantido o conteúdo do ficheiro (por exemplo, a sequência de caracteres que compõem um ficheiro de texto). No caso de diretorias, a região de dados mantém a respetiva tabela, que representa o conjunto de ficheiros (ficheiros normais e sub-diretorias) que existem nessa diretoria.

Para facilitar a alocação e libertação de *i-nodes* e blocos, existe um vetor de alocação associado à tabela de *i-nodes* e à região de dados, respetivamente.

Além das estruturas de dados mencionadas acima, que mantêm o estado durável do sistema de ficheiros, o TecnicoFS mantém uma tabela de ficheiros abertos. Essencialmente, esta tabela conhece os ficheiros atualmente abertos pelo processo cliente do TecnicoFS e, para cada ficheiro aberto, indica onde está o cursor atual. Ao contrário das estruturas de dados anteriores, a tabela de ficheiros abertos é descartada quando o sistema é desligado ou termina abruptamente (ou seja, não é durável).

Nas aulas teóricas da 2ª semana, o código base será apresentado e discutido. Recomendamos a todos os estudantes que frequentem essas aulas antes de começarem a desenvolver a solução.

## Simplificações

Além de uma API simplificada, o desenho e implementação do TecnicoFS adotam algumas simplificações fundamentais, que sumamos de seguida:

- Em vez de uma árvore de diretorias, o TecnicoFS tem apenas uma diretoria (a raiz “/”), dentro da qual podem existir ficheiros (e.g., “/f1”, “/f2”, etc.) mas não outras sub-diretorias.
- O conteúdo dos ficheiros e da diretoria raiz é limitado a um bloco. Como consequência, o *i-node* respetivo tem um campo simples que indica qual o índice desse bloco.
- Assume-se que existe um único processo cliente, que é o único que pode aceder ao sistema de ficheiros. Consequentemente, existe apenas uma tabela de ficheiros abertos e não há permissões nem controlo de acesso.
- A implementação das funções assume que estas são chamadas por um cliente sequencial. Ou seja, a implementação pode resultar em erros caso uma ou mais funções sejam chamadas concorrentemente por duas ou mais tarefas (*threads*) do processo cliente. Por outras palavras, não é *thread-safe*.
- As estruturas de dados que, em teoria, deveriam ser duráveis, não são mantidas em memória secundária. Ou seja, quando o TecnicoFS é terminado, o conteúdo destas estruturas de dados é perdido.

# 1º Exercício

No 1º exercício pretende-se estender a versão base do TecnicoFS com as funcionalidades que são descritas de seguida.

## 1. Ficheiros com múltiplos blocos

Pretende-se remover a restrição de um bloco por ficheiro apenas.

Em vez disso, os ficheiros devem passar a usar múltiplos blocos, referenciados da seguinte forma:

- 1º ao 10º bloco: referências diretas, em campos do próprio *i-node*
- Restantes blocos: *i-node* refere um bloco de índices (alocado na região de dados para ficheiros que tenham mais que 10 blocos), cujo conteúdo é usado como uma tabela de referências para até  $n$  blocos suplementares, em que  $n = \text{tamanho dos blocos} / \text{tamanho de um índice de bloco}$ .

O tamanho dos blocos deve ser 1 KByte (definido na constante/macro `BLOCK_SIZE`). Cada índice de bloco deve ser do tipo *int*.

Por simplificação, não se exige que esta extensão seja implementada para o conteúdo da diretoria raiz (ou seja, o seu conteúdo continua a ser limitado pelo tamanho de um bloco).

## 2. Cópia para o sistema de ficheiros externo

Pretende-se que seja oferecida e implementada uma nova função:

- `int tfs_copy_to_external_fs(char const *source_path, char const *dest_path);`

Esta função copia o conteúdo de um ficheiro existente no TecnicoFS, identificado por *srcPath*, para o conteúdo de um outro ficheiro, identificado por *destPath*, que é mantido no sistema de ficheiros do sistema operativo no qual o processo cliente/TecnicoFs correm. Por outras palavras, exporta o conteúdo de um ficheiro do TecnicoFS para fora deste.

Devolve 0 em caso de sucesso, -1 em caso de erro.

Caso o ficheiro identificado por *dest\_path* não exista, este deve ser criado. Caso contrário, o conteúdo prévio do ficheiro já existente deve ser totalmente substituído pelo novo conteúdo.

## 3. Suporte a chamadas concorrentes por cliente multi-tarefa

Embora a versão base do TecnicoFS seja sequencial, é desejável permitir que o programa do processo cliente seja constituído por múltiplas tarefas concorrentes (criadas por *pthread\_create*). Pretende-se então rever a implementação das funções do TecnicoFS para as tornar corretas quando executadas concorrentemente (i.e., *thread-safe*).

Para cumprir este requisito, devem ser usados trincos lógicos (*pthread\_mutex*) ou trincos de leitura-escrita (*pthread\_rwlock*), ficando a sua escolha ao critério dos alunos.

A solução deve maximizar o paralelismo, recorrendo a sincronização fina. Ou seja, uma solução baseada em trincos globais será fortemente penalizada na avaliação.

Caso sintam que a interface de funções que gerem o estado do TecnicoFS (definida em *fs/state.h*) não é a mais apropriada para resolver este requisito, podem adaptar essa interface para uma alternativa mais apropriada. Recomendamos que, antes de começarem a implementar, discutam o desenho da vossa solução com o docente de laboratório.

Além da adaptação do TecnicoFS para o tornar *thread-safe*, cada grupo deve **compor uma bateria de, pelo menos, três programas cliente paralelos**.

#### 4. Função de destruição bloqueante

Este último requisito consiste em implementar a seguinte função:

- `int tfs_destroy_after_all_closed();`

A função *tfs\_destroy\_after\_all\_closed* é uma variante mais complexa da função *tfs\_destroy*. Quando chamada, a função *tfs\_destroy\_after\_all\_closed* deve verificar se há algum ficheiro aberto. Caso haja pelo menos um ficheiro aberto, a função deve bloquear até que essa condição se deixe de verificar. Quando já não existirem ficheiros abertos, a função deve finalmente destruir o TecnicoFS, ou seja fazer aquilo que a alternativa *tfs\_destroy* faz.

Para implementar este requisito, devem ser evitadas soluções com espera ativa. Para tal, é aconselhado o uso de variáveis de condição (*pthread\_cond*).

Além disso, devem ser prevenidas situações de minguagem ou interbloqueio. Naturalmente, a implementação deve também ser *thread-safe*.

## Submissão e avaliação

A submissão é feita através do Fénix **até ao dia 7/janeiro/2022 às 23h59**.

Os alunos devem submeter um ficheiro no formato *zip* com o código fonte e o ficheiro *Makefile*. O arquivo submetido não deve incluir outros ficheiros (tais como binários). Além disso, o comando *make clean* deve limpar todos os ficheiros resultantes da compilação do projeto.

Recomendamos que os alunos se assegurem que o projeto compila/corre corretamente no cluster *sigma*. Ao avaliar os projetos submetidos, em caso de dúvida sobre o funcionamento do código submetido, os docentes usarão o cluster *sigma* para fazer a validação final.

O uso de outros ambientes para o desenvolvimento/teste do projeto (e.g., macOS, Windows/WSL) é permitido, mas o corpo docente não dará apoio técnico a dúvidas relacionadas especificamente com esses ambientes.

A avaliação será feita de acordo com o método de avaliação descrito no site da cadeira.