# Projeto de Sistemas Operativos 2020-21
# 1º enunciado
# LEIC-A/LEIC-T/LETI

Este enunciado especifica em detalhe o 1º exercício do projeto. Pressupõem-se a leitura prévia do enunciado global do projeto.

## Exercício 1

Pretende-se desenvolver uma primeira versão paralela do servidor do TecnicoFS. Esta versão ainda não suportará a interação com os processos clientes (isso só será suportado no 3º exercício). Em alternativa, deve executar sequências de chamadas carregadas a partir de um ficheiro de entrada. Para efeitos de depuração, quando o servidor terminar, este deve também apresentar o tempo total de execução no *stdout* e escrever o conteúdo final da diretoria num ficheiro de saída.

Consequentemente, a arquitetura da solução a desenvolver no 1º exercício será mais simples que a arquitetura ilustrada na Figura 1. A Figura 2 apresenta a arquitetura da solução do 1º exercício.

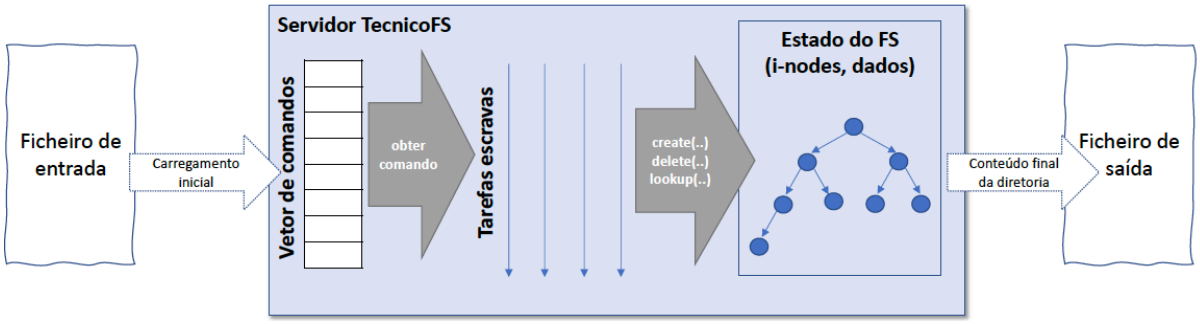


*Figura 2: Arquitetura da solução do 1º exercício*

O programa do servidor deve chamar-se *tecnicofs* e receber obrigatoriamente os seguintes quatro argumentos de linha de comando, cujo significado é descrito ao longo desta secção:

```
tecnicofs inputfile outputfile numthreads synchstrategy
```

De seguida descrevemos cada requisito do 1º exercício em maior detalhe.

### 1. Carregamento do ficheiro de entrada

O servidor recebe como 1º argumento de linha de comandos (*inputfile*) o caminho de acesso (*pathname)* de um ficheiro de entrada. Este ficheiro especifica uma sequência de comandos, um por cada linha. Cada comando deve dar origem a uma chamada à função correspondente do FS. Nesta fase do projeto, existem os 3 comandos seguintes, que apenas cobrem parte da funcionalidade do FS:

- **Comando 'c'**
  Argumentos: *nome tipo*
  Adiciona à diretoria uma nova entrada, cujo nome e tipo (diretoria ou ficheiro) são indicados em argumento. O tipo pode ser 'd' (diretoria) ou 'f' (ficheiro normal).

- **Comando 'l':**
  Argumento: *nome*
  Pesquisa o TecnicoFS por um ficheiro/diretoria com nome indicado em argumento.
- **Comando 'd':**
  Argumento: nome
  Apaga do TecnicoFS o ficheiro/diretoria com o nome indicado em argumento.

A especificação detalhada destas funções, incluindo os casos de erro, encontra-se definida no ficheiro *fs.h* do código fornecido (ver abaixo).

De notar que, nos exercícios posteriores do projeto, o FS será estendido com mais comandos que oferecerão a restante funcionalidade.

Além dos 3 comandos, podem existir ainda linhas começadas por **'#'**: estas linhas servem como comentários e são ignoradas pelo programa, não contando para o limite do número de comandos aceites pelo mesmo.

De seguida apresenta-se um exemplo do conteúdo do ficheiro de entrada:

```
# isto e' um exemplo
c /s d
c /s/f.txt f
l /s/f.txt
d /s/f.exe
```

Uma vez iniciado, o servidor deve criar a diretoria raíz, que está inicialmente vazia. De seguida, abre o ficheiro de entrada e carrega os comandos contidos no ficheiro para um vetor em memória, de tamanho fixo de 150 mil entradas (parâmetro definido numa constante). O carregamento termina assim que se alcance o final do ficheiro (*end of file*) ou o vetor seja totalmente preenchido.

A execução dos comandos só deve iniciar depois do ficheiro estar totalmente carregado no vetor, tal como se descreve de seguida.

## 2. Paralelização do servidor

### 2.1 Pool de tarefas

Para que o servidor seja capaz de executar operações em paralelo, a solução deverá manter uma *pool* de tarefas escravas. O número de tarefas escravas é definido como 3º argumento de linha de comando do servidor (*numthreads*). A *pool* de tarefas só é criada quando o ficheiro de entrada foi totalmente carregado para o vetor de chamadas.

### 2.2 Sincronização por trinco global

Cada tarefa escrava deve aceder ao vetor de chamadas, retirar o próximo elemento que esteja pendente e executá-lo sobre a diretoria partilhada. O vetor de chamadas e o conteúdo do TecnicoFS (nomeadamente, a tabela de *i-nodes* e os dados de cada diretoria/ficheiro) são partilhados, pelo que devem ser sincronizados adequadamente.

O acesso ao vetor de chamadas deve ser protegido por um mutex (*pthread_mutex*).

Em relação à sincronização do conteúdo do TecnicoFS , devem ser implementadas duas estratégias:

i) Usando um mutex (*pthread_mutex*) global

ii) Usando um trinco de leitura-escrita (*pthread_rw_lock*) global em vez do *mutex* acima

A escolha da estratégia a usar é determinada pelo 4º argumento de linha de comandos (*synchstrategy).* Este argumento pode ter 3 valores possíveis: mutex (corresponde a usar-se o mutex global), rwlock (corresponde a usar-se o trinco de leitura-escrita global) e nosync (nenhuma sincronização deve ser usada). Naturalmente, a última opção só é correta for se usada com execuções sequenciais (*numthreads=1*).

### 3. Terminação do servidor

Assim que todas as chamadas do ficheiro de saída tenham sido executadas, o servidor deve executar dois procedimentos antes de terminar. Primeiro, deve imprimir no *stdout* o tempo total de execução, contado a partir do momento em que a *pool* de tarefas foi iniciada, usando o seguinte formato:

```
TecnicoFS completed in [duration] seconds.
```

onde [duração] é o tempo medido em segundos com 4 casas decimais de precisão (p.e. “2.5897”).

Seguidamente, o conteúdo final do FS deve ser exportado para um ficheiro de saída. O nome deste ficheiro é indicado como 2º argumento de linha de comandos (*outputfile*), aquando do lançamento do servidor. Caso o ficheiro já exista, o seu conteúdo deve ser truncado. O formato do conteúdo do ficheiro de saída encontra-se já definido no código fornecido aos alunos (ver abaixo). Este ficheiro de saída poderá depois ser analisado para verificar se o estado final é o esperado ou não (neste caso, a implementação terá algum erro).

## Ponto de partida

Como ponto de partida para o 1º exercício, é fornecida uma implementação incompleta do servidor do TecnicoFS, que pode ser obtida no site da disciplina (secção “Laboratórios”).

Esta solução:

- Implementa o TecnicoFS, oferecendo já as 3 operações básicas (criação, pesquisa e remoção de ficheiro). No entanto, é uma implementação sequencial.
- Implementa também o carregamento da sequência de chamadas, mas a partir do *stdin* (em vez de ser a partir do ficheiro de entrada especificado como argumento de linha de comandos).
- Finalmente, implementa a impressão do conteúdo do TecnicoFS no *stdout* (em vez se ser para um ficheiro de saída especificado como argumento de linha de comandos).

## Experimente

Utilizando os exemplos de ficheiros de entrada que são fornecidos juntamente com o código inicial, execute-os sequencialmente no TecnicoFS (usando 1 tarefa apenas, sem sincronização). Guarde os respetivos ficheiros de saída (como o código base escreve no *stdout*, pode simplesmente redirecionar o *stdout* para o ficheiro pretendido) e anote os tempos de execução respetivos.

Numa máquina multi-core, experimente agora correr os mesmos exemplos usando 2, 4 ou mais tarefas (até ao número de cores da sua máquina[1]), experimentando ambas as estratégias de sincronização.

Para cada caso:

1. Compare o ficheiro de saída com o ficheiro obtido na execução sequencial do mesmo problema. Pode usar o comando *diff* para a comparação. O conteúdo é exatamente o mesmo? Caso tenha encontrado diferenças, como as explica?
2. Analise agora as diferenças de desempenho entre as diferentes variantes que testou. As execuções paralelas conseguem sempre vantagem no desempenho (em relação à versão sequencial)? Caso contrário, como explica que uma solução mais paralela seja por vezes mais lenta? O desempenho das variantes com *mutex* e com trinco de leitura-escrita depende do padrão de comandos de cada ficheiro de entrada?

*Nota: a resposta às perguntas acima não faz parte da avaliação do exercício.*

## Entrega e avaliação

Consultar o enunciado geral.

[1] Pode utilizar o comando *nproc* para obter o nº de *cores* lógicos disponíveis.