Projeto de Sistemas Operativos
2019-20
1º enunciado
LEIC-A/LEIC-T/LETI

Este enunciado começa por apresentar uma visão global do projeto de SO. De seguida, especifica em detalhe o 1º exercício do projeto.

Visão global do projeto

O objetivo final do projeto é desenvolver um sistema de ficheiros (*File System*, FS) em modo utilizador, chamado TecnicoFS.

Os FS em modo utilizador são uma alternativa que tem vindo a ganhar relevância recentemente, já que permitem o desenvolvimento rápido de FS facilmente portáveis e com forte isolamento de falhas, tal como será discutido nas aulas teóricas durante o semestre. Num FS em modo utilizador, as funcionalidades do FS são oferecidas num processo servidor (que, naturalmente, corre em modo utilizador). Outros processos podem chamar funções do FS através de pedidos ao núcleo do Sistema Operativo, que, por sua vez, encaminha esses pedidos ao processo servidor do FS através de um canal de comunicação estabelecido com este. Posteriormente, o retorno da função é devolvido ao cliente invocado pela via inversa.

Ao contrário de FS tradicionais, que guardam a informação em blocos (e.g., num disco magnético ou SSD), o TecnicoFS é desenhado para armazenar a informação nas novas tecnologias de memória persistente, tipicamente chamadas Non-Volatile RAM (ou NVRAM). As memórias NVRAM começaram a ser comercializadas recentemente na forma de DIMMs (tal como os DIMMs de memória DRAM). Como são persistentes, podem ser usadas para, por exemplo, manter o conteúdo de um FS. Além de persistentes, estas novas memórias podem ser acedidas através dos mesmos mecanismos de memória virtual que os programas usam para ler e escrever de DRAM; ou seja, são acessíveis ao nível do byte através de instruções *assembly* como a instrução MOV. Este aspeto constitui a grande motivação para o desenvolvimento destas novas memórias. Esta tecnologia não estará disponível no projeto concreto da cadeira, logo o mesmo será testado sobre memórias tradicionais.

Arquitetura

A arquitetura final do TecnicoFS encontra-se ilustrada na Figura 1. Para suportar acessos concorrentes e otimizar o desempenho, as funcionalidades do TecnicoFS são oferecidas, em paralelo, por um conjunto de tarefas escravas no processo servidor do FS.

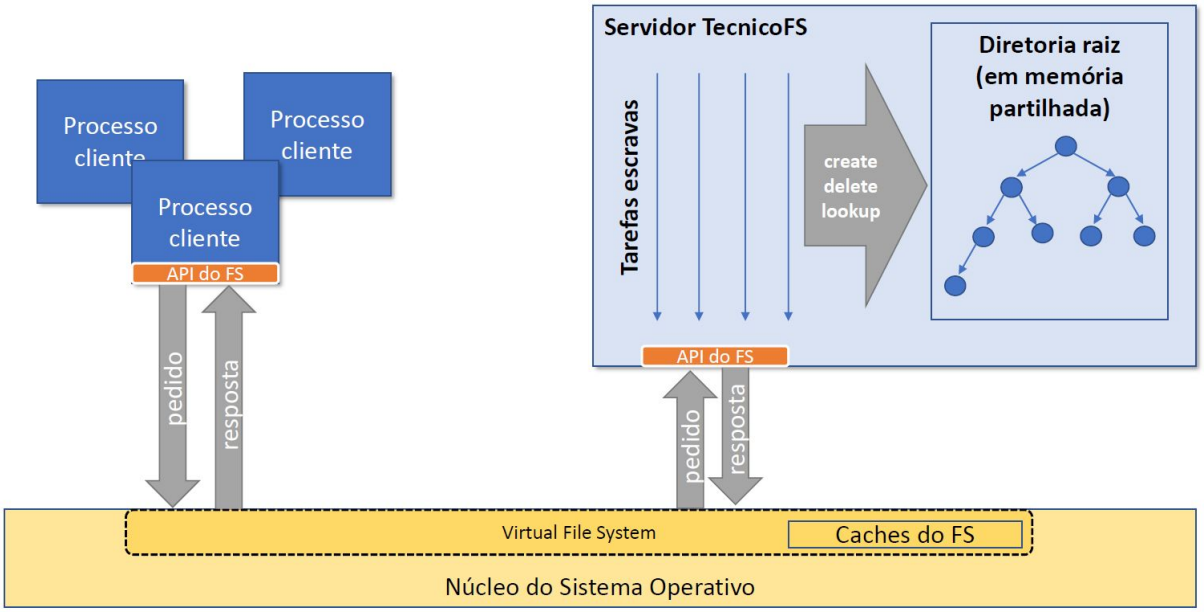

Figura 1: Arquitetura final do TecnicoFS

Por simplificação, o projeto foca-se apenas na implementação da diretoria raiz do TecnicoFS e a sua API apenas oferece três funções básicas: **criar**, **pesquisar** e **remover ficheiro**. A especificação destas funções encontra-se definida no ficheiro *fs.h* do código fornecido (ver abaixo). Note-se que as 3 funções suportadas são apenas ilustrativas, não suportando todas as opções das funções equivalentes na API do FS do Unix.

Toda a restante funcionalidade do FS fica fora do âmbito do projeto (incluindo o suporte à restante hierarquia de sub-diretorias e o acesso propriamente dito aos ficheiros).

Para suportar eficientemente pesquisas sobre um grande número de ficheiros, a diretoria é mantida numa árvore binária de pesquisa (*Binary Search Tree*, BST). Cada nó na árvore representa um ficheiro. A chave de um nó corresponde ao nome do ficheiro (tipo *string*). O valor mantido no nó corresponde ao nº de inode (também chamado i-number, do tipo *int*), que identifica univocamente o ficheiro no FS.

Os i-numbers a atribuir aos novos ficheiros são gerados sequencialmente, havendo um contador de i-numbers (variável partilhada), que mantém o i-number a atribuir ao próximo novo ficheiro. Os i-numbers dos ficheiros removidos não são reutilizados.

Etapas de desenvolvimento

O TecnicoFS será desenvolvido gradualmente ao longo de 3 exercícios, realizados pelos alunos, e terá uma fase final (não avaliada) que será realizada nas aulas teóricas após a entrega do último exercício. De seguida resumem-se estas etapas.

Exercício 1: Desenvolve o servidor do TecnicoFS, composto por uma pool de tarefas escravas que executam as operações do FS sobre a diretoria partilhada em memória. Nesta fase, há duas simplificações importantes: i) a sincronização dos acessos à diretoria partilhada recorre a um trinco global (ou seja, restringindo o paralelismo efetivo); ii) as chamadas a funções do TecnicoFS por parte de processos cliente ainda não são suportadas; em vez disso, as chamadas são simuladas pelo carregamento de um ficheiro contendo sequências de comandos.

Exercício 2: Estende a solução anterior com uma sincronização mais fina, o que permitirá maior paralelismo efetivo. Também melhora o carregamento do ficheiro de entrada para permitir que as operações se executem em paralelo com o carregamento inicial do ficheiro.

Exercício 3: Suporta a invocação de operações do FS por processos cliente. Nesta fase, por simplificação, essas chamadas **não** são tratadas pelo gestor de sistemas de ficheiros do SO (VFS, em Linux). Em vez disto, os processos cliente fazem chamadas ao TecnicoFS enviando mensagens através de um *socket*, que são consumidas pelas tarefas no servidor e que respondem aos clientes com o respetivo resultado. Adicionalmente, este exercício estende o TecnicoFS com uma nova funcionalidade que, quando ativada, salvaguarda o conteúdo da diretoria do novo FS num FS externo.

Etapa final (não avaliada, realizada nas aulas teóricas): Finalmente, o projeto desenvolvido até esta fase será adaptado para que as chamadas sejam feitas através da API do FS e passem pelo Gestor de FS do Sistema Operativo (VFS em Linux).

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 não será exatamente aquela descrita na Figura 1. A Figura 2 apresenta a arquitetura da solução do 1º exercício.

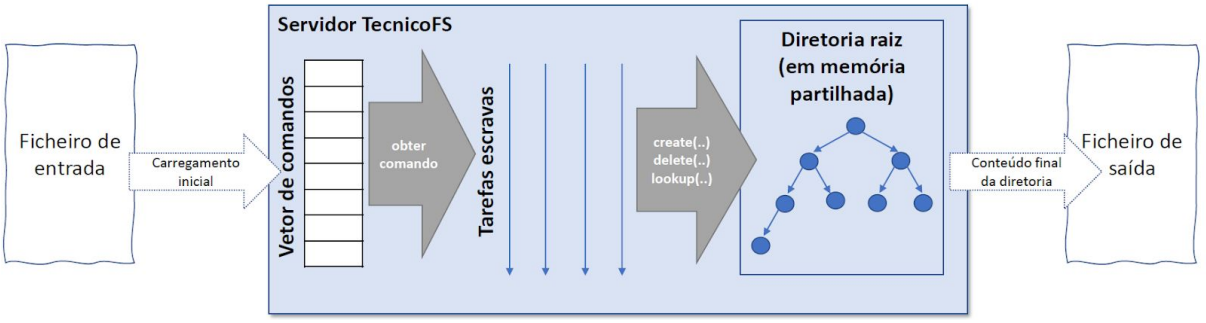

Figura 2: Arquitetura da solução do 1º exercício

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

```
tecnicofs inputfile outputfile numthreads
```

De seguida descrevemos cada requisito em maior detalhe.

Carregamento do ficheiro de entrada

O servidor recebe como 1º argumento de linha de comandos 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, e assim ser

executado sobre a diretoria raiz mantida pelo servidor. Existem 3 comandos possíveis, sendo que todos recebem como argumento único o nome de um ficheiro:

- **Comando 'c':** adiciona à diretoria uma nova entrada, cuja chave é o nome indicado em argumento;
- **Comando 'l':** pesquisa a diretoria por uma entrada com nome indicado em argumento;
- **Comando 'd':** apaga da diretoria a entrada com o nome indicado em argumento.

Além destes 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 f.txt
c s.exe
l f.txt
d s.exe
```

Uma vez iniciado, o servidor deve criar a diretoria, 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.

Paralelização do servidor

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. A *pool* de tarefas só é criada quando o ficheiro de entrada foi totalmente carregado para o vetor de chamadas.

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, o contador de *i-numbers* e a diretoria são partilhados, pelo que devem ser sincronizados adequadamente.

O acesso ao vetor de chamadas e ao contador de i-numbers deve ser protegido por um mutex (*pthread_mutex*). A sincronização deve assegurar o seguinte invariante: se duas chamadas de tipo 'c' surgem no vetor numa dada ordem, os *i-numbers* gerados para cada entrada devem também refletir a ordem no vetor.

Em relação à sincronização dos acessos à diretoria, devem ser implementadas duas estratégias, que são escolhidas em tempo de compilação por uma *macro* do pré-processador de C:

i) Usando um mutex (*pthread_mutex*) associado à diretoria (escolhida passando `-DMUTEX` ao compilador)

ii) Usando um trinco de leitura-escrita (*pthread_rw_lock*) em vez do *mutex* acima (escolhida passando `-DRWLOCK` ao compilador)

Caso não seja passada nenhuma das flags indicadas acima, deve ser compilada uma variante em que toda a sincronização é desativada. Esta variante serve apenas para execuções sequenciais (*numthreads=1*).

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 da diretoria deve ser exportado para um ficheiro de saída. O nome deste ficheiro é indicado como 2º argumento de linha de comandos, 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 a diretoria raiz, oferecendo já as 3 operações básicas (criação, pesquisa e remoção de ficheiro) sobre esta diretoria. 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 da árvore no *stdout* (em vez se ser para um ficheiro de saída especificado como argumento de linha de comandos).

Notamos que a solução fornecida assume algumas simplificações importantes, que se manterão durante o projeto (ou seja, não será valorizada qualquer tentativa de suprir estas simplificações):

- A BST não usa mecanismos de balanceamento (como aqueles usados nas árvores AVL, estudadas em IAED).
- A solução não está desenhada para tolerar situações em que o sistema falha a meio de uma operação de criação/remoção na BST, deixando o conteúdo da BST inconsistente.
- Como a NVRAM não está ainda disponível em máquinas comuns, a BST é mantida em DRAM. Para maior realismo, as instruções que lêem/escrevem na BST são precedidas de uma instrução que impõe um atraso que simula a latência esperada de uma NVRAM. Esta instrução não deve ser removida nas soluções submetidas pelos alunos.

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

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. A *Makefile* deve assegurar que, quando executado o comando *make* sem argumentos, sejam gerados 3 executáveis: `tecnicofs-nosync`, `tecnicofs-mutex` e `tecnicofs-rwlock` (cada um correspondendo a cada estratégia de sincronização descrita anteriormente). Além disso, o comando *make clean* deve limpar todos os ficheiros resultantes da compilação do projeto.

O exercício deve **obrigatoriamente** compilar e executar nos computadores dos laboratórios. 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 submissão é feita através do Fénix até ao dia 11/outubro às 23h59.

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

Os alunos devem consultar regularmente a plataforma piazza. É lá que serão prestados esclarecimentos sobre este enunciado.

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