INSTITUTO SUPERIOR TÉCNICO
Departamento de Engenharia Informática

Enunciado do Projecto de
Sistemas Operativos – Parte II
LEIC/LERC 2011-2012

Introdução

O ponto de partida da parte II do projecto de Sistemas Operativos é o sistema de ficheiros *sofs,* cujo código fonte está disponível no site do projecto. O sofs pode ser instalado em qualquer directoria (*mount point*) num sistema Linux. Os utilizadores e aplicações podem gerir os ficheiros mantidos pelo sofs usando a API tradicional de ficheiros do Linux.

O sofs é um sistema de ficheiros muito simples, baseado na plataforma FUSE. Possui as seguintes características:

- O tamanho máximo dos ficheiros é de 10 blocos;
- Tamanho dos blocos tem dimensão fixa de 512 bytes. O número de blocos que perfazem o disco é definido em tempo de compilação.
- A arquitectura do sistema de ficheiros é baseada numa simplificação dos i-nodes Unix , com 10 entradas directas para blocos de dados. O tamanho de um i-node é de 64 bytes. O número máximo de i-nodes é, por omissão, de 64 (a tabela de i-nodes ocupa, por isso 8 blocos), mas este valor pode ser definido em tempo de compilação.
- Existe apenas um directório (raiz), onde todos os ficheiros residem. Ou seja, não são suportadas sub-directorias. O directório raiz é um ficheiro estruturado sob a forma de uma tabela de entradas. Cada entrada tem 16 bytes e é constituída por: nome do ficheiro e número do i-node respectivo. Os nomes de ficheiros são limitados a 14 caracteres e o número do i-node a 2 bytes. Considera-se que a tabela do directório raiz pode ocupar, no máximo 4 blocos, isto é suporta até 128 entradas.
- Os blocos do disco estão organizados da seguinte forma:
  - Bloco 0: reservado para o bitmap de blocos livres.
  - Bloco 1: reservado para o bitmap de i-nodes livres.
  - Blocos 2-9: reservados para a tabela de i-nodes.
  - Blocos >= 10: para os dados dos ficheiros e directórios.

O *sofs* está pronto a funcionar sobre dois modos de armazenamento: modo simples (*simple-sofs*) e modo baseado em log (*log-sofs*). A opção de configuração -DUSE_LOG_SOFS serve para activar o modo *log-sofs* (caso não seja usada, será usado o modo *simple-sofs*.

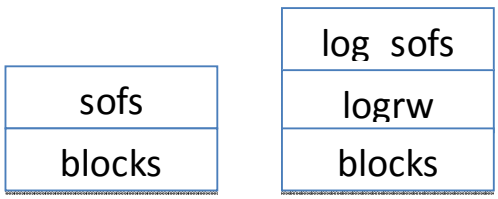

Figura 1. Divisão em módulos do *simple-sofs* e *log-sofs*

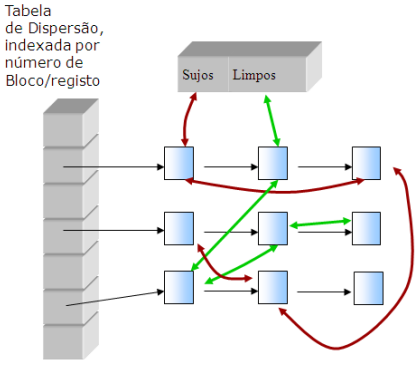

Figura 2. Organização da cache

O modo *simple-sofs* trabalha directamente sobre blocos num disco virtual, usando a API definida em *include/block.h*.
O modo *log-sofs* usa a solução construída no problema I.A do projecto de Sistemas Operativos para armazenamento dos dados dos ficheiros. Mais precisamente, o disco virtual *log-sofs* divide o disco virtual em dois vectores de blocos, sendo que em cada momento um desses vectores é o *array activo* (no qual novas escritas serão feitas) e o outro array é o *array inactivo*. Como consequência, sempre que o *log-sofs* precisa de modificar os dados de algum ficheiro, os novos conteúdos são escritos num novo bloco do disco virtual (mais precisamente, o próximo bloco que esteja livre no actual array activo no disco virtual).
Conceptualmente, a diferença entre os modos *sofs-simple* e *log-sofs* está ilustrada na Figura 1. Enquanto que o *log-simple* manipula *blocos* usando a API de block.h (usando as funções *block_read*, *block_write*, etc), o *log-sofs* manipula *registos*, usando a API definida em *logrw.h* (usando as funções *logrw_read*, *logrw_write*, etc). Onde no *sofs-simple* o inode de um ficheiro ou directoria referenciava um conjunto de blocos, no *log-sofs* referencia um conjunto de registos (cujo conteúdo está armazenado num dado bloco, bloco esse que mudará de cada vez que haja escrita ao registo).

O objectivo da parte II do projecto de Sistemas Operativos é optimizar o desempenho do *sofs* através da introdução de uma cache de blocos/registos (consoante o modo considerado, *simple-sofs* ou *log-sofs*). ~~As diferenças de desempenho da solução final comparativamente ao *sofs* original deverão ser analisadas e explicadas num artigo curto a entregar com o projecto~~.

O guião da aula laboratório da semana de 14-18 Nov. descreve o FUSE em detalhe. Recomenda-se vivamente a consulta e execução desse guião antes de começar a desenvolver a parte II do projecto.

Entrega Final: 2 de Dezembro às **23:59**
A entrega final consiste na submissão das Partes I e II (problemas I.A, I.B e II.A). Mais informação sobre a entrega e avaliação será afixada na página da cadeira.

Problema II.A: Cache de blocos/registos
O *sofs* não utiliza cache de blocos/registos. Logo, qualquer acesso a um bloco (por exemplo, para ler um inode ou conteúdo de um ficheiro) implica um ou mais acessos ao disco virtual, que é extremamente lento.
Pretende-se explorar a localidade de acessos implementando uma cache de blocos que intercepte os acessos de leitura e escrita de blocos do disco virtual.

A cache deverá funcionar tanto sobre o *simple-sofs* como sobre o *log-sofs*. No caso do *simple-sofs*, será uma cache de *blocos*, que interceptará chamadas *block_read* e *block_write*. No caso do log-sofs, será uma cache de *registos,* que interceptará chamadas *logrw_read* e *logrw_write*.

O conteúdo dos blocos/registos em cache deverá ser mantido num vector (*array*) com capacidade fixa, definida pelo parâmetro *dim_cache* (que indica quantos blocos/registos poderão estar simultaneamente em *cache*).

O funcionamento da cache será equivalente sobre ambos os sistemas de ficheiros (*sofs* e *log_sofs*).
Quando há um pedido de leitura de um bloco/registo, a cache verifica se já tem esse bloco/registo. Se sim, serve imediatamente o pedido. Se não, obtém o bloco/registo do sistema de ficheiros (chamando

block_read ou *logrw_read*, consoante o sistema de ficheiros em causa seja o *sofs* ou o *log-sofs*), guarda-o numa posição livre na cache e retorna à aplicação.
Quando há um pedido de escrita de um bloco/registo, a cache verifica se já tem esse bloco/registo. Se sim, altera a cópia em cache e marca-a como *sujo* (*dirty*). Se não, cria nova entrada em cache com o conteúdo a escrever e marca-a como sujo.

A cache deverá estar organizada como ilustrado na Figura 2. Ou seja, deverá haver uma tabela de dispersão (*hash table*) que mantém ponteiros para cada bloco/registo actualmente em cache, indexados pelo respectivo número de bloco/registo. Esta tabela de dispersão suportará a procura de cada bloco/registo na cache.
Para além da tabela de dispersão, deverão também ser mantidas 2 listas ligadas: uma cujos elementos referenciam os blocos/registos em cache que estão limpos; outra lista cujos elementos refenciam os blocos/registos em cache que estão sujos.
As listas de blocos/registos *clean* e *dirty* deverão ordenar os seus blocos/registos pelo tempo do último acesso a cada bloco. Ou seja, na lista de blocos/registos clean, aqueles cujo último acesso ocorreu há mais tempo devem estar no início da lista. O mesmo acontece com a lista de blocos/registos sujos.

Quando um acesso a um bloco/registo que não estava em cache encontra a cache cheia, é necessário retirar um bloco/registo da cache para receber o novo bloco/registo. A política de escolha do bloco/registo a retirar da cache é: se houver pelo menos um bloco/registo limpo em cache, então aquele cujo último acesso aconteceu há mais tempo deve ser escolhido; caso contrário, deve ser escolhido o bloco/registo sujo cujo último acesso aconteceu há mais tempo. No segundo caso, o conteúdo do bloco/registo sujo deve ser primeiro escrito em disco.

Quando o *sofs* é *unmounted,* todos os blocos/registos *dirty* em cache devem ser escritos para o sistema de ficheiros, sequencialmente (ou seja, um a seguir ao outro).

Como podem ocorrer acessos concorrentes à cache, a mesma deve ser protegida por um único trinco de leitura/escrita. Qualquer operação que pretenda ler da cache deverá antes trancar o trinco para leitura. Qualquer operação que modifique a cache deverá primeiro obter o trinco para escrita.

Para efeitos de debugging e de teste, a cache deve oferecer a função *cache_dump*, que deverá imprimir a seguinte informação no ecrã.

```
Capacidade: <capacidade da cache, em número de blocos/registos>
Clean blocks/registers: <lista do número de cada bloco/registo limpo em cache>
Dirty blocks/registers: <lista do número de cada bloco/registo sujo em cache>
Hit ratio: <nº de hits/nº total de acessos>
Misses: <nº total de faltas de cache (misses)>
Flushed: <nº total de blocos/registos sujos que foram flushed para FS>
```

Notas úteis:

1) Recomenda-se um desenho único da cache para ambos os sistemas de ficheiros (*sofs* e *log-sofs*), que apenas difira na API que usa para interagir com o sistema de ficheiros (API definida em *block.h* no caso do *sofs* e API definida em *logrw.h* no caso do *log-sofs*). Sugestão: usar a opção de compilação –DUSE_LOG_SOFS para controlar qual a API que deve ser usada pela cache.
2) Os alunos deverão suportar a opção de compilação –DNO_CACHE, que desactiva a cache.
3) Para tabela de dispersão, recomenda-se o da implementação *uthash* (disponível em http://uthash.sourceforge.net/).

4) Os grupos que não tenham concluído com sucesso uma solução concorrente para o problema I.A podem usar uma versão não-sincronizada do módulo *logrw.c*. Como o trinco que sincroniza os acessos à cache é global, tal garante, em cada momento, se houver uma tarefa a chamar a função logrw_write, mais nenhuma outra tarefa irá chamar nem *logrw_write* nem *logrw_read*. Isto permite que os acessos ao logrw feitos através da cache sejam correctamnte servidos mesmo se o *logrw* não sincronizar acessos concorrentes.
5) Deverá ser usada a nova versão do módulo *io_delay*, que atrasa os acessos ao disco virtual consoante a distância relativamente ao acesso anterior. Ou seja, caso dois acessos seguidos acedam ao mesmo bloco ou a blocos consecutivos, o atraso imposto ao acesso é muito menor que na situação em que ambos os blocos estejam em posições distantes no disco. Esta distinção nos atrasos pretende modelar mais fielmente o comportamento de um disco magnético, em que acessos a sectores que estejam fisicamente distantes têm um tempo de posicionamento e de latência muito maior que acessos a sectores consecutivos na mesma pista.

~~Artigo: Avaliação do desempenho da solução~~

~~É expectável que a cache tenha impacto no desempenho do sistema de ficheiros desenvolvido.~~
~~Pretende-se que os alunos completem o seu trabalho medindo e explicando as diferenças de desempenho obtidas.~~
~~Para tal, serão publicadas no site do projecto um conjunto de *benchmarks* que os alunos deverão executar com as seguintes 3 opções:~~

~~1) *simple-sofs* com cache inactiva;~~
~~2) *simple-sofs* com cache activa;~~
~~3) *log-sofs* com cache activa;~~

~~Os tempos medidos com cada combinação benchmark/solução deverão ser descritos num artigo a entregar juntamente com o projecto. Mais importante que descrever os resultados obtidos, o artigo deverá apresentar explicações que fundamentem as principais diferenças observadas entre as 3 diferentes soluções.~~
~~O artigo terá um máximo de 2 páginas A4, com letra de tamanho 12 pontos e margens de página não inferiores a 2,5cm. O formato deverá ser o *pdf*.~~

Apêndice:

Instruções de instalação do projecto da parte I no módulo *sofs* da parte II

No pacote da parte II (*sthreads+sofs+logrw-student-pkg-v1,* disponível no site do projecto), efectuar as seguintes operações:

- Substituir o directório *sthread_lib* pelo directório *sthread_lib* da parte I;
- No directório *logrw*, substituir o ficheiro *logrw.c* pelo ficheiro *logrw.c* da parte I.