INSTITUTO SUPERIOR TÉCNICO

Departamento de Engenharia Informática

Enunciado da Parte II do Projecto

Sistemas Operativos

LEIC/LERC 2009/10

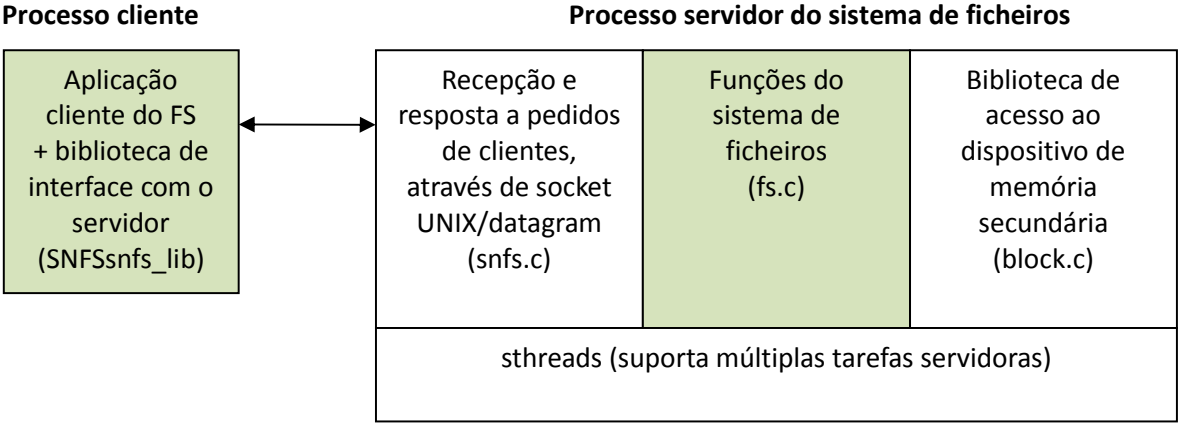
Este documento complementa o enunciado inicial do Projecto de Sistemas Operativos 2009/10

Parte II – Sistema de Ficheiros

Objectivos

Melhorar o desenho e implementação de um sistema de ficheiros já existente. A versão actual do sistema de ficheiros tem importantes limitações, que afectam o seu desempenho e eficiência no armazenamento, conforme se descreve de seguida. Pretende-se incorporar no sistema de ficheiros um número de melhoramentos que colmatem essas limitações.

Arquitectura Base



A arquitectura do sistema de ficheiros é a apresentada na figura acima.

Cada dispositivo de memória secundária é controlado por um servidor de sistema de ficheiros. O sistema de ficheiros suporta múltiplos níveis de directorias. No entanto, o sistema não utiliza qualquer tipo de cache. Os detalhes do desenho do sistema de ficheiros devem ser consultados no Anexo.

O acesso ao dispositivo é feito através de uma biblioteca que permite ler e escrever blocos de um dispositivo fictício. Os dispositivos também não incluem qualquer tipo de cache, pelo que a leitura

e escrita de blocos de um disco são operações consideravelmente demoradas (ordem dos segundos) e que são tratadas sequencialmente.

Os pedidos ao sistema de ficheiros chegam de aplicações cliente, cada uma a correr num processo independente na mesma máquina que o servidor. Os pedidos são enviados e respondidos através de um socket datagram UNIX do servidor, cujo nome é previamente conhecido.

Diversos clientes podem fazer pedidos ao mesmo servidor, que por isso pode receber múltiplos pedidos em paralelo. O sistema de ficheiros tira partido da biblioteca *sthreads* (a mesma implementada na primeira parte do projecto) para manter múltiplas tarefas, que são capazes de servir pedidos em paralelo.

Cada cliente consiste numa aplicação que, para aceder ao servidor do sistema de ficheiros, invoca funções da biblioteca snfs_lib. A biblioteca é, então, responsável por converter tais pedidos para uma variante do protocolo SNFS (ver Anexo), através do qual consegue interagir com o servidor.

Todos estes componentes são disponibilizados à partida. Os componentes que deverão ser modificados para a parte II estão indicados a sombreado na figura acima.

Requisitos

Claramente, o sistema de ficheiros é pouco interessante em vários aspectos. Para a segunda parte, pretende-se que se incorporem os seguintes melhoramentos:

1. Caching

Implementar no servidor os diferentes tipos de cache existentes em sistemas de ficheiros actuais: cache de blocos, cache de inodes e cache de directorias. Todas as caches devem ser mantidas em memória primária, usando as estruturas de dados que considerar mais adequadas. Os limites de cada cache devem ser facilmente parametrizáveis em tempo de compilação. Por omissão, devem ser usados os seguintes:

- Cache de blocos: 10 entradas
- Cache de inodes: 4 entradas
- Cache de directorias: 4 entradas

A política de substituição de todas as caches deve priveligiar as entradas mais recentemente acedidas, em especial aquelas que precisam de ser escritas em memória secundária antes de serem substituidas. Fica ao critério dos alunos a opção de como implementar tal política de forma eficiente.

As entradas que sejam alteradas enquanto estão em cache não devem ser escritas imediatamente em disco (*flushed*). A escrita deve ser adiada até (i) a entrada ser ser seleccionada pela política de substituição, ou (ii) depois de ter decorrido um intervalo de 10 segundos desde que a escrita em cache aconteceu.

2. Suporte a replicação de servidores para melhor desempenho

Para melhor desempenho, pretende-se que se suporte a instalação de múltiplos servidores, cada qual uma réplica do mesmo sistema de ficheiros. Se todos os servidores mantiverem o mesmo estado persistente, então operações que apenas envolvem leituras podem ser consideravelmente aceleradas: o cliente envia o pedido a múltiplos servidores; assim que o servidor menos carregado nesse momento responda, o cliente pode de imediato devolver a resposta à aplicação.

Onde antes se teria um servidor único, com um socket nomeado, por exemplo, *server_name,* agora ter-se-á N servidores-réplica instalados, cada um com um socket com nome *server_name_1, .., server_name_N*.

Na solução multi-servidor, a biblioteca snfs_lib (do lado do cliente) deverá tratar os pedidos de forma diferente consoante envolvam operações que alterem o estado persistente do sistema de ficheiros (e.g. snfs_write ou snfs_mkdir) ou não (e.g. snfs_lookup ou snfs_read):

- Funções que alteram o estado persistente do servidor:
  - o cliente envia o pedido para **todos** os N servidores-réplica instalados;
  - o cliente espera pela resposta de **todos** os N servidores-réplica instalados;
  - se nem todos os servidores retornarem o mesmo *STATUS* (*ok*, *error*, ou *unknown*) na sua resposta, então considera-se que ocorreu uma falha grave; nesse caso, o processo cliente deverá terminar abruptamente a sua execução (o sistema não trata faltas do grupo de servidores);
  - se o *STATUS* nas respostas de todos os servidores for idêntico, o cliente retorna uma dessas respostas à aplicação.
- Funções que não alteram o estado persistente do servidor:
  - O cliente envia o pedido a **M** servidores-réplica (M<=N é constante parametrizável); no caso de M<N, os servidores-réplica devem ser escolhidos em *round-robin*;
  - O cliente espera pela primeira resposta a chegar e entrega-a de imediato à aplicação.
  - À medida que as restantes respostas chegarem, elas devem ser descartadas.

3. Requisito transversal: sincronização eficiente

Transversalmente, todas as funções do sistema de ficheiros devem ser servidas da forma o mais paralela possível, para tirar partido de situações em que o servidor receba pedidos de múltiplos clientes em paralelo. A implementação da sincronização pode ser feita usando monitores e/ou trincos lógicos e/ou semáforos.

Apenas os componentes sombreados na figura devem ser modificados para assegurar os requisitos acima idenficidados.

Entrega e Avaliação

Data da entrega final: 9 de Dezembro às 12:00

A entrega final consiste na **entrega das Partes I (ver enunciado principal do projecto) e II**.
Consultar o enunciado principal do projecto para mais informação sobre formas de entrega e discussão do projecto.

Anexo – Pacote API SNFS, Servidor Multi-Tarefas e Sistema de Ficheiros SNFS

1. Material fornecido

O material dado consiste num pacote que implementa os seguintes componentes:

- Cliente SNFS;
- Servidor Multi-Tarefa;
- Sistema de Ficheiros SNFS.

A biblioteca SNFS desenvolvida permite a uma aplicação cliente aceder ao sistema de ficheiros de um servidor remoto SNFS. O código das aplicações cliente liga-se com a biblioteca SNFS que, por sua vez, está dividida em duas camadas:

- Uma camada de suporte à comunicação com o servidor (API SNFS).
- Uma camada de interface com a aplicação (Interface Sistema de Ficheiros).

A camada API SNFS consiste num conjunto de rotinas que:

- formata as mensagens SNFS;
- comunica com o servidor através de sockets domínio Unix sem ligação (SOCK_DGRAM). A utilização deste tipo de sockets possui a vantagem de delimitar automaticamente as mensagens e de enviar os dados nas mensagens sem necessidade de conversão.

Para esta parte é disponibilizado o material seguinte:

- Servidor SNFS funcional.
- Formato das mensagens do protocolo SNFS para todos os serviços.
- Esqueleto da biblioteca SNFS para os clientes.

A camada Interface Sistema de Ficheiros oferece aos programadores uma interface de programação semelhante àquela que é disponibilizada pela biblioteca standard da linguagem C. O cliente utiliza a API SNFS sempre que necessita de comunicar com o servidor. Internamente, esta camada gere os ficheiros abertos pela aplicação cliente.

O **servidor SNFS** que é fornecido aos alunos funciona no modo multi-tarefa. A arquitectura multi-tarefa do servidor SNFS obedece ao seguinte desenho:

- Existe um número limitado de tarefas consumidoras (TC) processam e responder aos pedidos dos clientes.
- Existe uma única tarefa (TP) que descobe quando existem novos pedidos e que os distribui pelas tarefas cliente.

A tarefa TP sincroniza-se com as tarefas TC, de modo a que:
Quando chega um novo pedido, TP assegura que uma tarefa TC, e apenas uma, fica responsável por este pedido:

- Se todas as tarefas TC estiverem ocupadas, o mecanismo de sincronização assegurar que o pedido não é esquecido.
- O número de tarefas TC é definido em tempo de compilação.

O **Sistema de Ficheiros** suportado pelo servidor de SNFS é muito simples, possuindo as seguintes características:

- O tamanho máximo dos ficheiros é de 10 blocos;
- Suporta a criação de subdirectórios no directório raíz;
- Tamanho dos blocos tem dimensão fixa de 512 bytes. O número de blocos 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 de i-nodes por omissão é de 64 (a tabela de i-nodes ocupa, por isso 8 blocos), mas este valor pode ser definido em tempo de compilação.
- Os directórios são ficheiros estruturados sob a forma de uma tabela de entradas. Cada entra da tem 16 bytes e é constituída por: nome do ficheiro/directório e número do i-node. Os nomes são limitados a 14 caracteres e o número do i-node a 2 bytes. Para simplificar a listagem dos directórios remotamente, considera-se que os directórios podem ocupar, no máximo 4 blocos, isto é suportam até 128 entradas.
    - A disposição dos dados nos blocos é a seguinte:
    - 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.

2. Protocolo SNFS

O protocolo SNFS define a comunicação entre clientes e servidores SNFS. O SNFS é baseado no (mas não igual ao) protocolo NFS Versão 2 (RFC 1094, http://tools.ietf.org/html/rfc1094).:

Serviço	Argumentos	Resultado	Descrição
SNFS_LOOKUP	Filename Pathname	stat status: se STAT_OK fhandle file unsigned fsize	Obtém o identificador "fhandle" e respectivo tamanho "fsize" do ficheiro ou directório "Pathname" localizado no directório "dir", se a resposta obtida for STAT_OK.
SNFS_READ	fhandle file unsigned offset unsigned count	stat status: se STAT_OK bytes data	Devolve até "count" bytes de "data" do ficheiro "file", a partir do byte "offset" a contar do início do ficheiro. O primeiro byte do ficheiro corresponde ao offset 0.
SNFS_WRITE	fhandle file unsigned offset unsigned count byte data	stat status: se STAT_OK unsigned fsize	Escreve "data" a partir do byte "offset" a partir do início do ficheiro "file". O primeiro byte do ficheiro está no offset 0. A operação de escrita é atómica. Os dados de uma escrita não serão misturados com os dados da escrita de outro cliente. Se for bem sucedida, a escrita devolve a dimensão actual do ficheiro em "fsize"
SNFS_CREATE	fhandle file filename name	stat status: se STAT_OK fhandle file	Os atributos iniciais do ficheiro são dados por "attributes". Se o resultado é STAT_OK, o ficheiro foi criado com sucesso e "file" e "attributes" contém o "fhandle" e os atributos do ficheiro. Caso contrário a operação falhou e nenhum ficheiro foi criado.

SNFS_MKDIR	fhandle dir filename name	stat status: se STAT_OK fhandle file	Cria o novo directório "name" no directório "dir". A resposta STAT_OK indica que o directório foi criado com sucesso e "file" contém o fhandle do directório. Caso contrário, a operação falhou e o directório não foi criado.
SNFS_READDIR	fhandle dir unsigned count	stat status: se STAT_OK entry* list unsigned count	Retorna um número variável de entradas do directório até "count" bytes do directório dado por "dir". Se o valor retornado é STAT_OK, então segue-se de um número variável de "entry"s. Cada entry é uma estrutura com o nome da entrada (ficheiro ou directório), tamanho do nome e indicação se o nome se trata de ficheiro/directório.

Tabela 1. Serviços do Protocolo SNFS

3. API do Cliente

A interface de programação do sistema de ficheiros SNFS é semelhante à oferecida pela biblioteca standard da linguagem C, estando disponíveis:

- *int my_init_lib();*

Inicializa as estruturas internas da biblioteca de ficheiros.

- *int my_open(char * nome, int flags);*

Devolve um inteiro que identifica o ficheiro em operações posteriores (*handle*). O Argumento *flags* é usado para modificar o comportamento da função, sendo que a *flag* com valor 1 indica que o ficheiro deve ser criado caso já não exista.

- i*nt my_read(int fileID, char * buffer, unsigned numBytes);*

Lê a partir de um ficheiro, para um *buffer* em memória, um número especificado de *bytes*. Devolve o número de *bytes* lidos, ou zero caso esteja no final do ficheiro.

- *int my_write(int fileID, char * buffer, unsigned numBytes);*

Escreve para um ficheiro, o conteúdo de um *buffer* em memória, com o tamanho especificado.

- *void my_close(int fileID);*

Fecha o ficheiro identificado pelo argumento *fileID*.

- *int my_listdir(char* path, char **filenames, int* numFiles);*

Escreve para o ponteiro **filenames* o endereço de memória onde estão contidos os nomes dos ficheiros que se encontram no directório *path* do sistema de ficheiros, separados por '\0', e no inteiro *numberOfFiles* a quantidade de dos mesmos ficheiros.