Universidade do Minho
Departamento de Informática

Sistemas Operativos

Trabalho Prático

SDStore: Armazenamento eficiente e seguro de ficheiros

Feito por:
João Gonçalves - A95019
Rodrigo Pereira - A96561
João Martins - A96215

29 de maio de 2022

Conteúdo

1 Introdução

Neste trabalho prático, é nos proposta a implementação de um serviço que permita aos utilizadores armazenarem uma cópia dos seus ficheiros de forma segura e eficiente, poupando espaço de disco. Para tal, o serviço disponibiliza funcionalidades de compressão e cifragem dos ficheiros a serem armazenados.

2 Estrutura adotada para o projeto

Neste projeto, com os desafios que nos foram propostos desenvolvemos a seguinte estrutura:

- O servidor como pedido, tem a sua configuração base e o caminho para os ficheiros binários com as transformações.
- O servidor possui um pipe com nome para receber os pedidos.
- Cada cliente, também possui um pipe com nome para receber as mensagens de resposta passadas pelo servidor.
- A gestão das mensagens enviadas entre clientes-servidor, é feita por um intermiador, um 'router'.

3 Router

O router é uma forma de gestão de encaminhamento de mensagens. As mensagens podem ser dos seguintes tipos:

- "inited" - mensagem para inicialização do router.
- "kill" - mensagem para desligar o router.
- "proc-file" - mensagem para executar uma sequência de transformações no servidor.
- "destroy" - mensagem de informação, o router foi desligado.
- "status" - mensagem para executar um pedido de status no servidor.
- "waiting" - mensagem de espera pela execução de uma sequência de transformações.
- "refused" - mensagem a informar que um pedido de execução de transformações foi recusado.

O router acaba por ser o processo responsável por conhecer os fifos de todos os outros processos. Podemos pensar no router como um intermediário ao qual qualquer processo pode enviar as "mensagens"a outros processos só com o id que identifica o processo de destino no router. Desta maneira, só a comunicacao por mensagens fica centralizada. Para um processo se ligar ao router tem que enviar um pedido de tipo 1 a este e o router vai lhe responder com uma mensagem de tipo -1 com o seu id correspondente. O router tem sempre o id 0 e o server tem sempre o id 1. Quando enviámos uma mensagem, o destino da mensagem tem que ser para onde o router vai redirecionar a mensagem(se o id for 0 a mensagem é destinada ao router), a origem da mensagem tem que ser o próprio id que foi atribuido ao processo pelo router.

Nota: As mensagens "proc-file" e "status" possuem tipos de mensagens de resposta, necessárias para informar os clientes dos estados dos seus pedidos. As mensagens "inited"e "destroy"também possuem mensagens de resposta mas apenas para gestão das comunicações.

4 Programas cliente e servidor

4.1 Programa cliente - sdstore

O programa cliente(sdstore), vai ser o responsável por efetuar pedidos para o servidor executar, como por exemplo o status e executar transformações com prioridade ou sem prioridade. O programa cliente faz o "parsing" das transformações para serem enviadas no pedido. Este "parsing" é feito de forma a criar uma sequência de carateres que representem as transformações. No nosso programa, o nop é representado pelo 0, bcompress pelo 1, bdecompress pelo 2, gcompress pelo 3, gdecompress pelo 4, encrypt pelo 5 e decrypt pelo 6.

Para enviar um pedido ao servidor por parte de um cliente executa-se o seguinte comando:

- *./sdstore proc-file [-p priority] input-filename output-filename transformation-id-1 transformation-id-2 ...*

4.2 Programa servidor - sdstored

O programa servidor(sdstored), vai ser o responsável por executar certos pedidos por parte do cliente, quer seja um pedido de status, quer seja para executar certas transformações em um ficheiro de texto. Para executar o servidor executa-se o comando:

- *./sdstored config-filename transformations-folder*

4.3 Ligação entre o cliente e o servidor

A ligação entre o cliente e o servido é feita pelo router. Quando um cliente quer enviar um pedido de execução de transformações é enviada uma mensagem ao router (através do pipe com nome do router) indicando a fonte, o destino (servidor) e as transformações. O router como conhece o servidor reencaminha-lhe esta mensagem. Se for um pedido de status é envidado de forma semelhante mas não existem dados a enviar no pedido. Estas e todas as mensagens são diferenciadas pelo tipo que lhes foi atribuído.

Por outro lado, quando o servidor que enviar uma mensagem de resposta, ou seja, "Pending", "Processing", "Concluded" e "Refused", o router envia uma mensagem ao servidor indicando a origem, correspondente ao próprio servidor, e o destino, o cliente que é representado por um id que foi atribuído pelo router na sua primeira conexão com ele. Para enviar uma mensagem de resposta a um pedido de status, para além destas informações de origem e destino é enviado a informação dos pedidos de tipo proc-file em execução e o número de tranformações por tipo em execução em comparação com o máximo possível no servidor.

4.4 Ciclo de vida de um pedido

O cliente pode fazer dois tipos de pedidos ao servidor:

- "proc-file" - executar um conjunto de tranformações sobre um ficheiro.
- "status" - informações sobre o estado do servidor.

Para podermos explicar o ciclo de vida de um pedido "proc-file", primeiro temos de introduzir como é guardada a informação necessária para o resolver. Para isso, usamos uma estrutura de dados definida como "task", que contém um identificador único, o número de transformações que vão ser executadas, o tamanho da path de origem, e o tamanho da path de destino, um array de inteiros que guarda o numero de transformações que vão ser executadas de cada tipo, o id do router, uma variável booleana que indica se a task está a ser executada ou não, um apontador para caracteres que guarda as transformações que vão ser executadas por inteiros (0-nop, 1-bcompress, 2-bdecompress, 3-gcompress, 4-gdecompress, 5-encrypt, 6-decrypt), o path de origem e o path de saída.

Quando é efetuado um comando para executar um certo número de transformações num ficheiro de texto ("proc-file"), por parte do cliente, é criada uma mensagem de tipo 2 (nova tarefa para ser executada por parte do servidor). Essa mensagem é serializada e enviada para o fifo do router, para ser tratada. Depois de ser enviada, no lado do servidor a mensagem tem de ser deserializada, de forma a que a task possa ser inserida na fila de prioridades, e possa informar o cliente de que a task está pendente. Depois disso, a task vai ser executada por parte do servidor, e este vai verificar se o número de transformações de cada tipo disponiveis é maior que o numero de transformações presentes na task. Se for, então a task pode ser executada, e o servidor envia uma mensagem ao cliente a dizer que a task está a ser processada. Depois disso, o número de transformações disponíveis de cada tipo tem de ser decrementado consoante o número de tasks de cada tipo que são executadas, e essas transformações vão ser executadas. Depois da task ser completamente efetuada por um dos filhos do servidor, esse envia uma mensagem para o router, dizendo que a task foi completamente efetuada. O router recebendo essa mensagem decrementa o número de tasks que estão a executar e elimina essa task. Dado isso, o router envia uma mensagem de término para o cliente, e o cliente imprime no ecrã o número de bytes de output e de input presentes nos ficheiros.

Quando é feito um pedido sobre o estado de funcionamento do servidor ("status"), o cliente tem de ser informado das tasks que estão a ocorrer no momento, como também do numero de transformações de cada tipo que ocorrem naquele momento e também do número de transformações máximas que podem estar a ocorrer ao mesmo tempo. Para isso, o cliente vai ter de fazer um pedido de status ao servidor, e o servidor recebendo esse pedido vai enviar as informações de volta ao cliente, para assim conseguir mostrar na tela do utilizador todas as informações. O

4.5 Funcionalidades Avançadas

Número de bytes de input e número de bytes de output utilizando a operação proc-file

Utilizando a operação proc-file, quando a task é concluida por parte do servidor, é possível ver a comparação do número de bytes de input com o número de bytes de output, depois de executar todas as transfoemações no ficheiro de input.

Terminação graciosa do servidor

Quando recebe um sinal SIGTERM, o servidor em vez de terminar abruptamente, corre a funcao de handling desse sinal que passa o servidor para um estado de terminação graciosa. Para tal acontecer, usamos a system call sigaction() que nos permite definir flags do comportamento do handling de sinais. Neste, retiramos a flag SA_RESTART que seria usada normalmente pela system call signal(). Esta flag, se estiver ativada, faz com que se um read, write, open e outras operações que bloqueiem, nao tentem continuar o que estavam a fazer quando foram interrompidas pelo sinal. Em vez de continuarem, estas retornam -1 e o errno fica definida a EINTR. No periodo de terminação graciosa o servidor recusa todos os novos pedidos com as menssagens de tipo 6 e corre as tasks todas que estão na queue. Acabando todas as tasks, o servidor fecha o router e liberta a sua memória alocada.

Prioridade com a opção proc-file

Utilizando a opção proc-file, é possível definir uma certa prioridade para a task a executar, sendo que se houver uma task com 5 de prioridade e outra com 3, a task com 5 irá executar primeiro do que a outra task.

4.6 Gestão de pedidos - Pedidos Recusados

O servidor é capaz de recusar pedidos, quando recebe o sinal para terminar de forma graciosa, ou seja esses pedidos já não serão executados, como também quando as transformações que estão na fila ultrapassam a capacidade do servidor de executar transformações daquele tipo.

5 Execução das transformações

Como proposto as transformações são aplicadas ao ficheiro passado como argumento e executadas de forma concorrente, pela ordem dada e o resultado é gerado no destino passado como argumento também. Como não podemos usar ficheiros intermédios para guardar os resultados das diversas transformações intermédias aplicadas, utilizamos pipes sem nome para esse efeito.

A primeira transformação é executada sobre o ficheiro original, na qual redirecionamos o standard input para o ficheiro original e redirecionamos o standard output para o descritor de ficheiro de escrita do pipe que comunica com a segunda transformação. As tranformações intermédias têm o seu standard input redirecionado para o descritor de ficheiro de leitura do pipe anterior e o standard output redirecionado para o descritor de escrita do pipe seguinte. A última

transformação é executada com o seu standard input redirecionado para o descritor de ficheiro de leitura do pipe anterior e o standard output redirecionado para o ficheiro de destino que foi passado como argumento.

Por fim, todas as tranformações obrigatoriamente são executadas com funções exec em processos filhos diferentes, e os pipes anteriormente referidos garantem a comunicação entre estes processos.

6 Conclusão

Em suma, o trabalho proposto foi bem realizado, e ajudou a perceber melhor as interações com o Sistema Operativo, como a utilização de pipes com nome(fifos), pipes sem nome e sinais. Também foi importante na questão de fazer a comunicação entre o programa cliente e o programa servidor, através de um "router", como também na forma como se fez a sincronização entre processos, que foi bastante trabalhosa utilizando métodos lecionados nas aulas práticas.