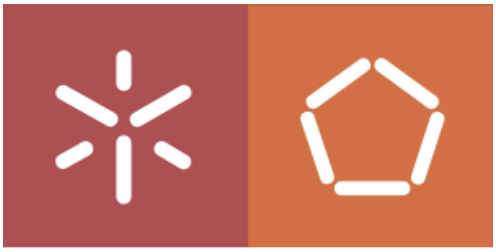

Universidade do Minho
Escola de Engenharia

Sistemas Operativos - Trabalho Prático

Grupo 14

Relatório

2021/2022

Autores:

A91660 – Pedro António Pires Correia Leite Sequeira

A91671 – João Manuel Novais da Silva

A91697 – Luís Filipe Fernandes Vilas

Braga,
27 de maio de 2022

Índice

Conteúdo

1- Introdução

Foi-nos proposto, no âmbito da unidade curricular de Sistemas Operativos, o desenvolvimento e implementação de um serviço que permita aos utilizadores armazenar uma cópia dos seus ficheiros segura e eficiente, poupando espaço no disco. Este tipo de sistema dispõe funcionalidades de compressão e cifragem dos ficheiros a serem armazenados. Além disso, o serviço suporta a submissão dos pedidos para processar e armazenar novos ficheiros assim como a recuperação do conteúdo original de ficheiros guardados previamente. Também é pretendido que o programa torne possível a consulta das tarefas em processamento num dado momento e estatística sobre os mesmos.

Neste relatório, vamos explicitar como abordamos o problema, justificando a estrutura e as decisões feitas no desenvolvimento do serviço demonstrando, assim, os conhecimentos adquiridos nas aulas, tais como a utilização e criação pipes e processos.

Inicialmente vamos descrever as funcionalidades, demonstrar estrutura com o exemplo de algumas funções importantes e, por fim, iremos demonstrar o raciocínio e o funcionamento em si do programa.

2- Funcionalidades

Informação de Utilização

O utilizador consegue saber que comandos o programa aceita introduzido na linha de comandos *./bin/sdstored*, para os comandos do lado do servidor, e *./bin/sdstore*, para o lado do cliente.

Inicialização do programa

A partir do comando ./bin/sdstored é possível abrir o servidor que irá tratar do processamento dos ficheiros submetidos pelo cliente. Este comando tem os seguintes argumentos: *config-filename* que é um ficheiro de texto onde está descrito o tipo de transformações que é suposto se fazer e quantas vezes se pode executar as transformações, e *transformations-folder* que é uma pasta onde estão armazenados o código fonte de todas as transformações possíveis. Sendo assim o comando deverá ser introduzido desta forma: *./sdstored <config-filename> <transformations-folde>r*.

O outro comando válido é *./bin/sdstore*, este comando está ser subdividido em outros dois comandos, o *./bin/sdstore* status que server para obter o estado de funcionamento do servidor, e o comando *./bin/sdstore <priority> <input-filename> <output-filename> transformation-id-1 transformation-id-2* …, em que o argumento *priority* representa a prioridade que cada transformação tem, o input-filename é o nome do ficheiro que irá sofrer transformações, output-filename é o nome do ficheiro resultante depois de todas as transformações, e por fim, o argumento *tranfomation-id-1*, … representa o "id" ou nome da transformação que é pretendida pelo cliente.

Transformações de Ficheiros

Este programa permite através de transformações comprimir, descomprimir, encriptar, desencriptar e clonar ficheiros. A compressão e descompressão (comandos *bcompress/bdecompress* e *gcompress/gdecompress*) pode ser feita em dois formatos, no formato *bzip* e *gzip*, os comandos *encrypt/decrypt* permitem cifrar e decifrar os conteúdos dos ficheiros, por último o comando *nop* permite copiar todos os dados do ficheiro sem qualquer transformação.

3 - Estrutura do Programa

Cliente (sdstore.c)

Para a implementação dos serviços do lado do cliente referidos anteriormente foi necessária a criação das seguintes funções:

- **void inicializaComunicacao()**

  Esta função abre o *pipe com nome* que faz ligação com o servidor e de seguida escreve nele o *pid* do processo (cliente) , depois cria o *pipe com nome* do cliente, para poder comunicar apenas o servidor com o cliente através desse *pipe*. Tendo em conta que o *pipe* com nome que faz ligação com o servidor, apenas serve para escalonar as operações para o cliente.

- **void status()**

  Esta função usa a função *int escreverParaServer(char* nomePipe, char* mensagem)*, a qual, abre o *pipe* do cliente e escreve a mensagem que é passada nos seus argumentos. O objetivo da status é saber o estado de execução das diferentes tarefas que o servidor está a processar.

- **void transformação(int argc, char const *argv[])**

  Esta função envia para o servidor uma lista de transformações (separadas por espaços) para o servidor as poder processar. É também passado o input-file (ficheiro onde o servidor irá ler) e o output-file (ficheiro onde o servidor irá escrever).

Servidor (sdstored.c)

Para a implementação dos serviços do lado do servidor referidos no capítulo anterior foi necessária a criação das seguintes funções:

- **TRANSFORMACAO loadTransforms(char *config)**

Esta função lê o ficheiro config que descreve as transformações e o número máximo de vezes que podem ser utilizadas e cria uma lista ligada do tipo TRANSFORMACAO em que cada nodo tem o nome da transformação, o número de execuções desta transformação e o número máximo de transformações .

- **UTILIZACOES inicializaUti(char *comando)**

  Esta função lê o input do cliente, que esta armazenada na lista ligada TAREFA, e cria uma lista ligada do tipo UTILIZACOES em que é armazenada o nome da transformação e o número de vezes que a transformação é executada.

- **void trabalhador()**

  Esta função está encarregue de atualizar o número de execuções a ser feitas no momento e começa o processamento da tarefa chamando a função *void executaTarefa(int input, int output, TAREFA t).*

- **void executaTarefa(int input, int output, TAREFA t)**

  Esta função recebe o descritor do ficheiro de input e do output abertos pelo *void trabalhador()* e a tarefa a executar. Para cada uma das transformações é criado um processo filho e estes comunicam com o processo pai a partir de um único *pipe*. Cada processo filho está encarregue de executar uma transoformação.

3.3 – Variáveis Globais

Cliente

- **char pid_processos[MAX_BUFFER]**

  Representa o pid do cliente atual.

- **char ficheiroComunicacao[MAX_BUFFER]**

  Caminho para o pipe com nome do cliente.

Servidor

- **char pid_processo[MAXBUFFER]**

  Representa o pid do cliente atual.

- **TRANSFORMACAO tranformacoes**

  Lista ligada com as transformações lidas do ficheiro config.txt.

- **TAREFA tasks**

  Tarefas que o servidor tem em lista de espera para executar.

- **char *loc_tranformacoes**

  Localização do ficheiro das transformações.

- **int podeAceitarOperacoes**

  Indicação ao servidor se pode aceitar mais operações, é inicializado a 1, e só é mudado quando o servidor recebe um sinal de término (SIGTERM).

4- Funcionamento do Programa

O programa começa com a inicialização do servidor, a partir do comando ./bin/sdstored config.txt <localização das transformações>. A partir desse momento, é criado na pasta tmp/servidor um pipe com nome que permite os clientes abrirem a comunicação com o servidor. Quando um cliente se liga, escreve no tmp/servidor o seu pid e cria um pipe com nome, para comunicação direta com o servidor, sem usar a mesma pipe com nome partilhado. Caso o utilizador pretenda processar um ficheiro, deverá passar como argumentos ./bin/sdstore proc-file <priority> input-filename output-filename transformacao1 transformacao2 … .

O servidor possui a função trabalhador(), que é chamada para iterar sobre as tarefas que existem para processar e, caso tenha disponibilidade para executar, executa. Para isso, cria um processo filho, responsável por controlar outro processo filho que irá executar a tarefa e por fim o processo filho quando o seu filho termina, envia um sinal ao pai para este retirar a tarefa das suas tasks e informar o cliente, também através de um sinal. Para executar cada tarefa é chamada a função executaTarefa que recebe a lista das transformações e os dois escritores do ficheiro input e output, esta cria um processo filho para cada transformação e aplica as por ordem sendo a comunicação entres processos filho feita por um pipe anónimo. O trabalhador() volta a ser chamado para verificar se ainda existem tarefas para fazer.