

UNIVERSIDADE DO MINHO

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Sistemas Operativos - Trabalho Prático
Grupo 39

André Alves (A95033) João Bernardo Escudeiro (A96075)
Hugo Martins (A95125)

Ano Letivo 2021/2022

Conteúdo

1	Introdução	3
2	Transformações	4
2.1	bcompress e bdecompress	4
2.2	gcompress e gdecompress	4
2.3	encrypt e decrypt	4
2.4	nop	4
2.5	SIGTERM	5
3	Utilização dos programas	6
3.1	Inicialização do servidor	6
3.2	Uso do programa pelo cliente	6
4	Decisões tomadas	9
5	Conclusão	10

Capítulo 1

Introdução

Este projeto consiste na implementação de um serviço que permita aos utilizadores armazenarem uma cópia dos seus ficheiros de forma segura e eficiente, poupando espaço de disco. Para além de realizar este armazenamento, o serviço deverá permitir a submissão de pedidos para processar e armazenar novos ficheiros bem como para recuperar o conteúdo original de ficheiros guardados previamente. Ainda, deverá ser possível consultar as tarefas de processamento de ficheiros a serem efetuadas num dado momento e estatísticas sobre as mesmas.

Numa fase inicial, o maior desafio foi encontrar uma forma de poder encadear um número arbitrário de comandos sucessivos. Mais tarde, também tivemos alguma dificuldade em conseguir obter o estado de uma tarefa, mas acabámos por conseguir superar estes obstáculos.

O cliente e o servidor comunicam através de dois pipes com nome, um envia comandos do cliente para o servidor, e o outro envia o output dos comandos do servidor para o cliente.

Capítulo 2

Transformações

2.1 bcompress e bdecompress

Estes comandos realizam respetivamente a compressão e a descompressão dos dados com o formato bzip utilizando o comando *bzip2* com o intuito de comprimir os ficheiros para ocupar menos espaço de disco.

2.2 gcompress e gdecompress

Estes comandos realizam respetivamente a compressão e a descompressão dos dados com o formato gzip utilizando o comando *gzip* tendo o mesmo intuito que o bcompress e bdcompress, comprimir ficheiros para utilizar menos espaço de disco.

2.3 encrypt e decrypt

Estes comandos realizam respetivamente a encriptação e a decríptação dos dados onde para o efeito utiliza-se o comando *ccrypt* para, como é assumido pelo nome, encriptar o ficheiro recebido. Este ao contrário dos outros dois acima como realiza uma *encriptação* é apenas possível aceder ao ficheiro com a chave que foi realizada a encriptação, adicionando assim mais um *layer* de segurança.

2.4 nop

Este comando copia os dados de um ficheiro sem realizar qualquer transformação ao ficheiro original

2.5 SIGTERM

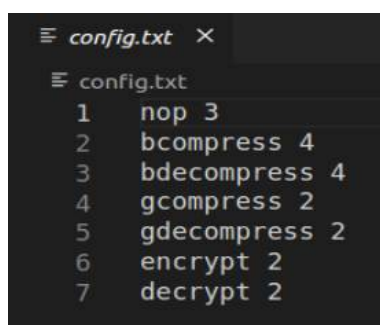
A partir do momento que o servidor recebe este sinal , começa a rejeitar novos pedidos e termina de executar os que ainda se encontrem na queue .Após a execução de todos os pedidos que estão na queue ,o servidor ”desliga”.

Capítulo 3

Utilização dos programas

3.1 Inicialização do servidor

Para o utilizador poder começar a realizar transformações sobre os ficheiros que pretenda, temos de inicializar o servidor. Para o efeito iniciamos com o comando `./bin/sdstored config.txt files`. Neste comando para além do `./bin/sdstored` temos um ficheiro `config.txt` que vai dizer ao servidor quais são as transformações que poderá realizar e juntamente o número de vezes que poderá realizar cada transformação concorrentemente. Estas transformações encontram-se na pasta `files`, logo é passado também ao servidor a diretoria como é representado acima. A baixo é ilustrado um exemplo de um ficheiro config usado por nós em testes.



```
≡ config.txt ×
≡ config.txt
1  nop 3
2  bcompress 4
3  bdecompress 4
4  gcompress 2
5  gdecompress 2
6  encrypt 2
7  decrypt 2
```

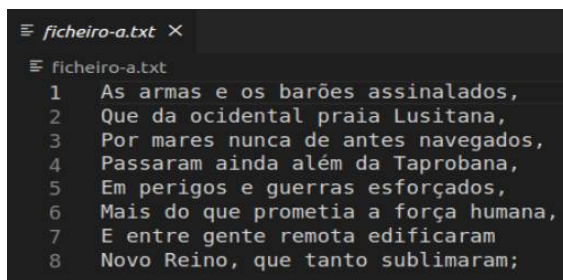
Figura 3.1: Exemplo de ficheiro `config`.

Como podemos ver, com esta configuração o cliente poderá usar, por exemplo, a transformação `nop` apenas 3 vezes concorrentemente.

3.2 Uso do programa pelo cliente


Após a inicialização do servidor, o cliente poderá realizar as transformações que pretender sobre o ficheiro que introduza como comando. Um exemplo de comando que o utilizador possa realizar é `./bin/sdstore ficheiro-a.txt ficheiro-out.txt nop`. Aqui para além do executável `./bin/sdstored` logo de seguida o utilizador coloca o ficheiro que queira realizar as transformações, onde neste

exemplo foi o ficheiro *ficheiro-a.txt*, e após isso o nome do ficheiro de output, que aqui tem o nome *ficheiro-out.txt*, que irá ser o nome do ficheiro que será o resultado do ficheiro de input após todas as transformações que o cliente o submeta. A seguir a isso o cliente coloca as transformações que pretenda realizar sobre o ficheiro original, onde neste exemplo foi apenas aplicada a transformação *nop*. Em baixo podemos ver o resultado da execução deste comando.



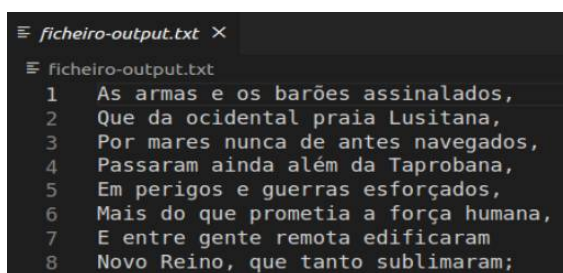
```
ficheiro-a.txt X
ficheiro-a.txt
1  As armas e os barões assinalados,
2  Que da ocidental praia Lusitana,
3  Por mares nunca de antes navegados,
4  Passaram ainda além da Taprobana,
5  Em perigos e guerras esforçados,
6  Mais do que prometia a força humana,
7  E entre gente remota edificaram
8  Novo Reino, que tanto sublimaram;
```

Figura 3.2: Ficheiro original.



```
andre@andre-VirtualBox:~/Desktop/TP50$ ./bin/sdstore ficheiro-a.txt ficheiro-output.txt nop
pending
executing
concluded (bytes-input: 276, bytes-output: 276)andre@andre-VirtualBox:~/Desktop/TP50$
```

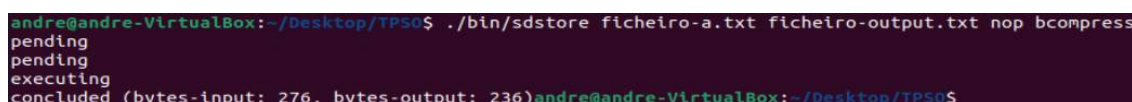
Figura 3.3: Comando introduzido pelo cliente.



```
ficheiro-output.txt X
ficheiro-output.txt
1  As armas e os barões assinalados,
2  Que da ocidental praia Lusitana,
3  Por mares nunca de antes navegados,
4  Passaram ainda além da Taprobana,
5  Em perigos e guerras esforçados,
6  Mais do que prometia a força humana,
7  E entre gente remota edificaram
8  Novo Reino, que tanto sublimaram;
```

Figura 3.4: Ficheiro de output.

Como podemos ver também na figura 3.3, é nos apresentado o estado do programa durante a sua execução, ou seja, apresenta que o programa se encontra *pending* enquanto aguarda para executar as transformações, *executing* enquanto está a executar as transformações passadas pelo cliente, e *concluded* quando terminar as transformações passadas pelo cliente. Para completar, após o término das transformações é nos apresentado pelo programa a quantidade de bytes que o programa recebeu pelo ficheiro de entrada e a quantidade de bytes que o ficheiro de output tem após as transformações. Como no exemplo acima só foi utilizada a transformação *nop*, sendo que esta só copia os dados e não altera o tamanho do ficheiro, vamos apresentar com a execução de mais uma transformação, nomeadamente a transformação *bcompress*, com o comando `./bin/sdstore ficheiro-a.txt ficheiro-output.txt nop bcompress`.



```
andre@andre-VirtualBox:~/Desktop/TP50$ ./bin/sdstore ficheiro-a.txt ficheiro-output.txt nop bcompress
pending
pending
executing
concluded (bytes-input: 276, bytes-output: 236)andre@andre-VirtualBox:~/Desktop/TP50$
```

Figura 3.5: Comando com a transformação *bcompress*

Como é representado acima, como foi realizada a transformação *bcompress* ao ficheiro de input, o ficheiro final vai ser uma versão comprimido do original, logo como era esperado, o seu tamanho é menor que o original, passando de 276 bytes para 236. Nesta imagem também podemos ver a implementação que referi acima de se poder atribuir as transformações de forma concorrente, sendo que pedi ao ficheiro para realizar a transformação nop e a seguir de forma concorrente a bcompress.

Capítulo 4

Decisões tomadas

Uma das principais decisões tomadas foi a implementação de uma queue para dar resposta , de forma ordenada , aos pedidos de transformações que sejam solicitados. Assim , os pedidos são inseridos na queue (sempre no fim da fila) à medida que são solicitados e são executados , sempre do início para o fim da fila .

O grupo optou pela não utilização de ficheiros temporários como sugere no enunciado .

Capítulo 5

Conclusão

Como é possível constar, este trabalho tem quase todas as funcionalidades pedidas implementadas, sendo que faltam apenas o comando *status* e a implementação do sistema de prioridades de ficheiros. Mesmo assim, acreditamos que o trabalho encontra-se bem conseguido e que nos permitiu consolidar tudo o que fomos aprendendo ao longo do semestre na cadeira de Sistemas Operativos.