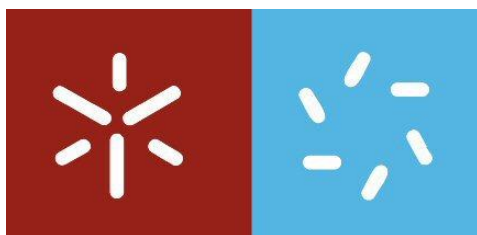


Relatório

29 de maio de 2022

Grupo 24



Licenciatura em Ciências da Computação

Sistemas Operativos

2021/2022

Projeto realizado por:

Carlos André Machado Costa a94543

Ruben Gonçalo Araújo da Silva a94633

Bruno Miguel Ferreira Fernandes a95972

1 Conteúdo

2	Introdução	3
3	Cliente	4
3.1	Proc-file	4
3.2	Status.....	4
3.3	Estado do Processo	4
3.4	Função Bytes_files.....	4
4	Servidor	5
4.1	Organização.....	5
4.2	Leitura de Processos.....	5
4.3	Comunicação	5
5	Testes	6
6	Conclusão	8

2 Introdução

O objetivo deste projeto consiste na criação de um sistema de transformação de ficheiros através de uma aplicação que implementa várias funções à escolha do seu utilizador.

O core deste trabalho é utilizar os diferentes recursos aprendidos nas aulas práticas e teóricas da disciplina para um ótimo desenvolvimento e eficiente execução da aplicação.

Esta tal é composta por duas grandes partes centrais, o servidor (*sdstored*) e os clientes (*sdstore*), que submetem pedidos para transformar o seu ficheiro ou recebem informação do estado deles.

Numa fase inicial o grande desafio foi encontrar uma forma de o cliente e o servidor se comunicarem, bem como a interpretação de todas as transformações desejadas pelo cliente a realizar. Mais tarde, tivemos também dificuldades em conseguir implementar a função *status* e interpretar o limite de transformações a realizar.

3 Cliente

Sobre o cliente (*sdstore*) este recebe como argumentos a opção que deseja que o servidor execute "*proc-file*" ou "*status*", o ficheiro que deseja alterar, o ficheiro onde quer que as alterações sejam guardadas e por fim recebe todas as transformações que deseja realizar ao ficheiro fornecido.

Também nele é criado todos os tipos de *pipes* que achamos necessário providenciar ao programa, entre eles são: "*pipe_status*", "*pipe_exec*", "*main_pipe*" e o "*pipe_process*".

3.1 Proc-file

Se o cliente solicitar a opção *proc-file*, esta é escrita no *pipe_exec* que mais tarde será aberto e lido pelo servidor para detetar a opção desejada pelo utilizador.

Também é colocado no *main_pipe* toda a informação disponibilizada, ou seja, é colocado o ficheiro a aplicar as transformações, o ficheiro para salvar essas alterações e por fim todas as respetivas transformações a exercer.

3.2 Status

Caso o cliente solicite a opção *status* esta é escrita no *pipe_exec* para também ser lida e aberta no servidor, e uma vez processada essa opção no servidor é nos retornado, toda a informação do mesmo, processos em andamento e os seus limites.

3.3 Estado do Processo

Depois de ser executado um pedido do utilizador, o servidor começa então a tratar do mesmo, no entanto este vai sinalizando ao cliente caso este processo tenha ficado no estado de "*pending*", "*processing*" ou "*concluded*", e quando este último estado é mostrado ao cliente, é feito também uma análise de bytes colocados no input e os bytes colocados no output ficando assim o utilizador a saber os respetivos tamanhos dos seus ficheiros.

3.4 Função Bytes_files

A função *bytes_files* foi criada para a contagem dos bytes dos ficheiros introduzidos pelo utilizador, ou seja, o ficheiro a realizar as transformações e o ficheiro onde estas vão ser salvas, e que retorna o seu resultado através da chamada de um sinal.

4 Servidor

O servidor, inicialmente está preparado para em primeiro lugar, tratar dos argumentos que lhe são fornecidos, neste caso, a pasta “*sdstore-transformations*” que contém todas as transformações possíveis a realizar pelo programa e o “*sdstore.conf*” que é o ficheiro que limita a quantidade que uma dada transformação possa realizar ao mesmo tempo.

4.1 Organização

O servidor é constituído por dois principais métodos de organização, o primeiro é uma *config struct*, onde nela é armazenado os nomes de todas as transformações possíveis a realizar, o número máximo de transformações que se podem executar ao mesmo tempo e também o número de transformações que estão a ser executadas.

Depois temos a queue onde é armazenada toda a informação dos pedidos do cliente, para que mais tarde seja feita, ordenadamente, a sua interpretação.

4.2 Leitura de Processos

Após a chegada de um processo ao servidor, este é então guardado na queue e é realizado o seu pedido. Caso haja mais do que uma transformação a fazer, são criados pipes para passar o resultado de cada um deles entre si, e quando é realizada a última transformação, o seu output é colocado então no ficheiro que o utilizador pretende salvar, se este ficheiro já tiver conteúdo nele este é substituído.

4.3 Comunicação

A comunicação do cliente para o servidor é feita como anteriormente mencionado pelos diversos pipes que nele são criados, no entanto do servidor para o cliente decidimos usar sinais. Quando um processo está á espera de inicializar a sua execução, é devolvido ao cliente um sinal, que depois de interpretado é colocado no seu terminal o estado “*pending*”, quando este está a ser executado é mandado outro tipo de sinal e desta vez é mostrado o estado “*processing*” e por fim quando este termina é mandado o último sinal que mostra então no terminal o estado “*concluded*” e também os respetivos números de bytes em cada ficheiro.

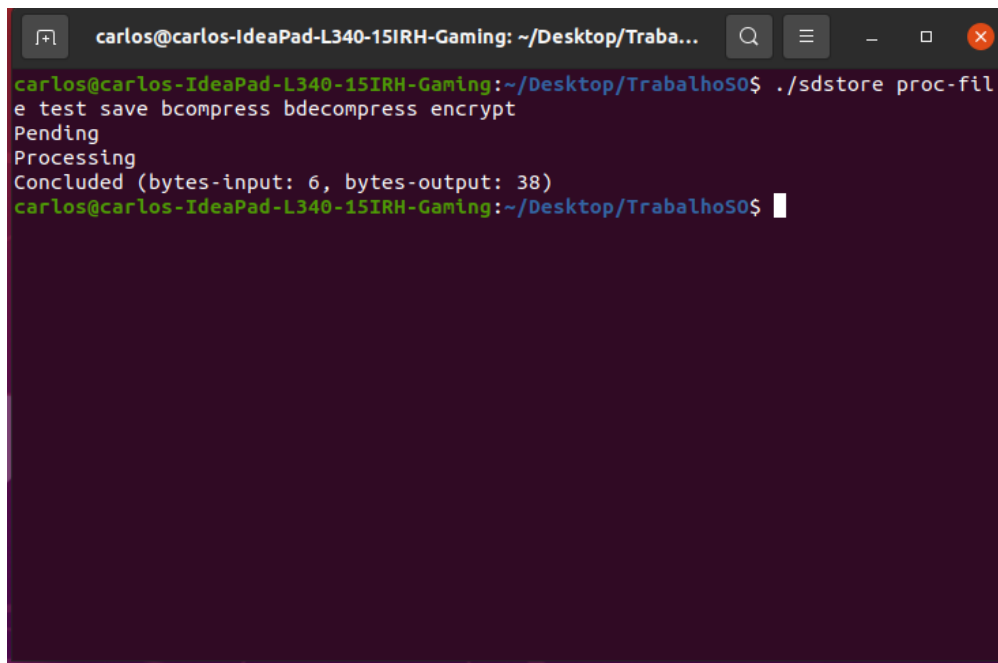
Quando é feito um pedido status ao servidor, neste caso, a comunicação é feita através do “*pipe_status*” que manda ao cliente a informação de cada uma das transformações, as que estão de momento a decorrer, bem como o máximo que o servidor suporta de cada uma.

5 Testes

Neste capítulo apresentamos alguns testes que realizamos para testar a funcionalidade do nosso programa.

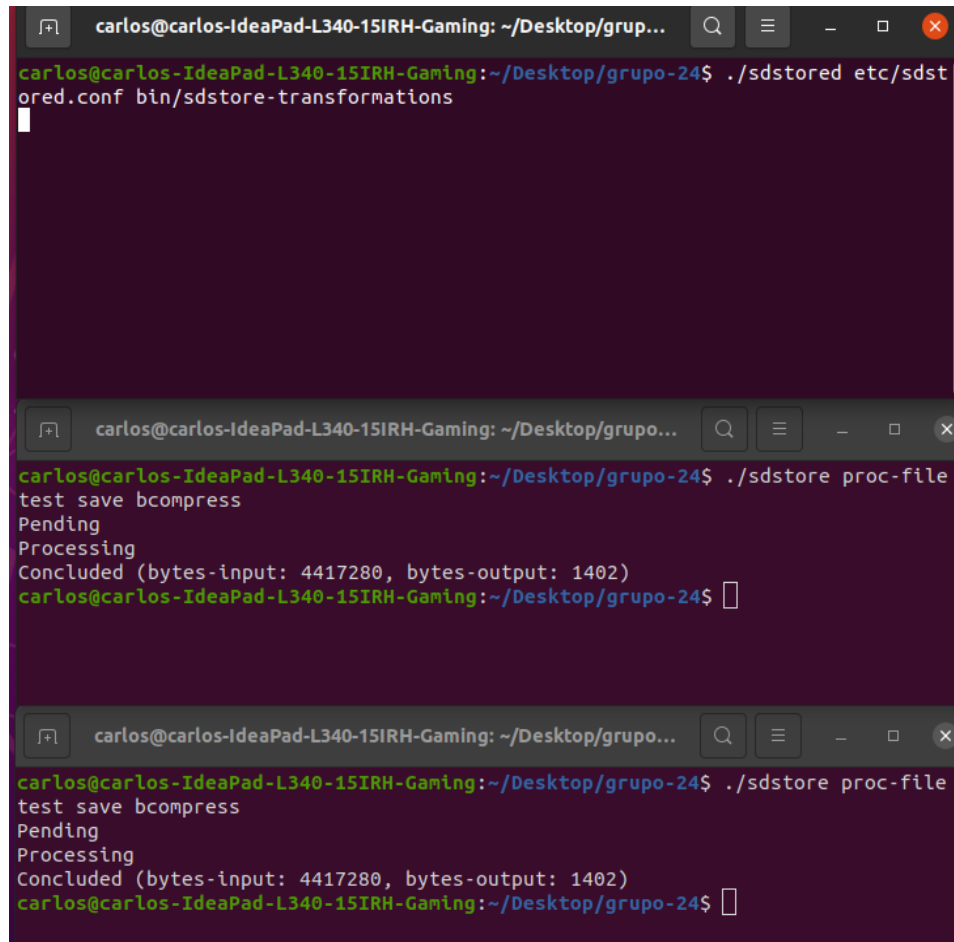
Em primeiro lugar testamos o nosso cliente para ver se para este era comunicado o estado do seu pedido, *"Pending"*, *"Processing"*, *"Concluded"*. Bem como se era feita a leitura avançada dos bytes de input do ficheiro *"test"* e dos bytes de output *"save"*.

Com este teste também conseguimos comprovar que o nosso servidor estava a funcionar como esperado, fazendo a interpretação de todos as transformações do nosso pedido no ficheiro pretendido, que neste caso foram: *"bcompress"*, seguido por *"bdecompress"* e por fim *"encrypt"*.

A screenshot of a terminal window with a dark background. The window title is 'carlos@carlos-IdeaPad-L340-15IRH-Gaming: ~/Desktop/Traba...'. The prompt is 'carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/TrabalhoS0\$'. The command entered is './sdstore proc-fil e test save bcompress bdecompress encrypt'. The output shows the states 'Pending', 'Processing', and 'Concluded (bytes-input: 6, bytes-output: 38)'. The prompt returns to 'carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/TrabalhoS0\$' with a cursor.

```
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/Traba...
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/TrabalhoS0$ ./sdstore proc-fil
e test save bcompress bdecompress encrypt
Pending
Processing
Concluded (bytes-input: 6, bytes-output: 38)
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/TrabalhoS0$
```

Para testar se o nosso servidor era capaz de ser concorrente, ou seja, saber se este era capaz de aguentar diversos pedidos de clientes ao mesmo tempo, abrimos 3 terminais, um para iniciar o servidor e os outros dois para clientes diferentes. E comprovamos assim que estava tudo a funcionar como esperado.



The image shows three terminal windows stacked vertically. Each window has a title bar that reads 'carlos@carlos-IdeaPad-L340-15IRH-Gaming: ~/Desktop/grupo...'. The first window shows the command `./sdstore etc/sdstored.conf bin/sdstore-transformations` being executed. The second window shows the command `./sdstore proc-file test save bcompress` being executed, followed by the output: 'Pending', 'Processing', and 'Concluded (bytes-input: 4417280, bytes-output: 1402)'. The third window shows the same command `./sdstore proc-file test save bcompress` being executed again, with the same output: 'Pending', 'Processing', and 'Concluded (bytes-input: 4417280, bytes-output: 1402)'.

```
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$ ./sdstore etc/sdstored.conf bin/sdstore-transformations

carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$ ./sdstore proc-file test save bcompress
Pending
Processing
Concluded (bytes-input: 4417280, bytes-output: 1402)
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$

carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$ ./sdstore proc-file test save bcompress
Pending
Processing
Concluded (bytes-input: 4417280, bytes-output: 1402)
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$
```

Para testar a função *status*, realizamos o mesmo teste, mas desta vez abrimos um cliente para requisitar o comando *status* ao servidor enquanto os clientes estavam no estado de “Processing”



The image shows a terminal window with the title 'carlos@carlos-IdeaPad-L340-15IRH-Gaming: ~/Desktop/grupo...'. The command `./sdstore status` has been executed, resulting in the following output: 'transf nop 0/3 (running/max)', 'transf bcompress 2/4 (running/max)', 'transf bdecompress 0/4 (running/max)', 'transf gcompress 0/2 (running/max)', 'transf gdecompress 0/2 (running/max)', 'transf encrypt 0/2 (running/max)', and 'transf decrypt 0/2 (running/max)'.

```
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$ ./sdstore status
transf nop 0/3 (running/max)
transf bcompress 2/4 (running/max)
transf bdecompress 0/4 (running/max)
transf gcompress 0/2 (running/max)
transf gdecompress 0/2 (running/max)
transf encrypt 0/2 (running/max)
transf decrypt 0/2 (running/max)
carlos@carlos-IdeaPad-L340-15IRH-Gaming:~/Desktop/grupo-24$
```

6 Conclusão

Como é possível constatar, conseguimos realizar uma grande parte do trabalho prático, implementando a grande maioria das funcionalidades pedidas no enunciado, contudo sentimos dificuldades na parte de implementação da status que se encontra incompleta e da limitação de transformações que infelizmente também não conseguimos implementar. No entanto introduzimos no nosso trabalho matéria relacionada com sinais que era facultativo e também conseguimos realizar uma das funcionalidades avançadas que neste caso foi a de contar os bytes dos ficheiros de input e de output.

Este trabalho permitiu-nos consolidar tudo aquilo que fomos aprendendo ao longo das aulas de Sistemas Operativos de uma forma criativa, eficiente e prática.