

Sistemas Operativos



Universidade do Minho
Licenciatura em Ciências da Computação

Grupo 11

Diogo Rodrigues (a94012) João Freitas
(a91667) Leonor Caldas (a91644)

Conteúdo

Introdução:.....	3
Funcionalidades:.....	4
Decisões Tomadas:.....	6
Conclusão.....	7

Introdução:

Este projeto da cadeira de Sistemas Operativos consiste num programa que aplica transformações a ficheiros de forma a que fiquem armazenados de forma mais segura e que ocupem menos espaço.

Este serviço é constituído por um cliente/servidor que comunicam através de um pipe com nome, sendo possível haver vários clientes em simultâneo.

Cada filtro tem um número máximo de utilizações em simultâneo, sendo controlado pelo servidor, colocando o cliente em espera até os filtros requeridos estarem disponíveis.

Funcionalidades:

Compilação:

\$ make all (criar objetos, criar executaveis)

\$ make clean (apagar objectos, executaveis e todos pipes com nome criados)

Servidor:

Através do comando seguinte irá ser criado o servidor, com ficheiro de configuração e a pasta das transformações usados (o servidor terá de ser executado na pasta principal) :

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

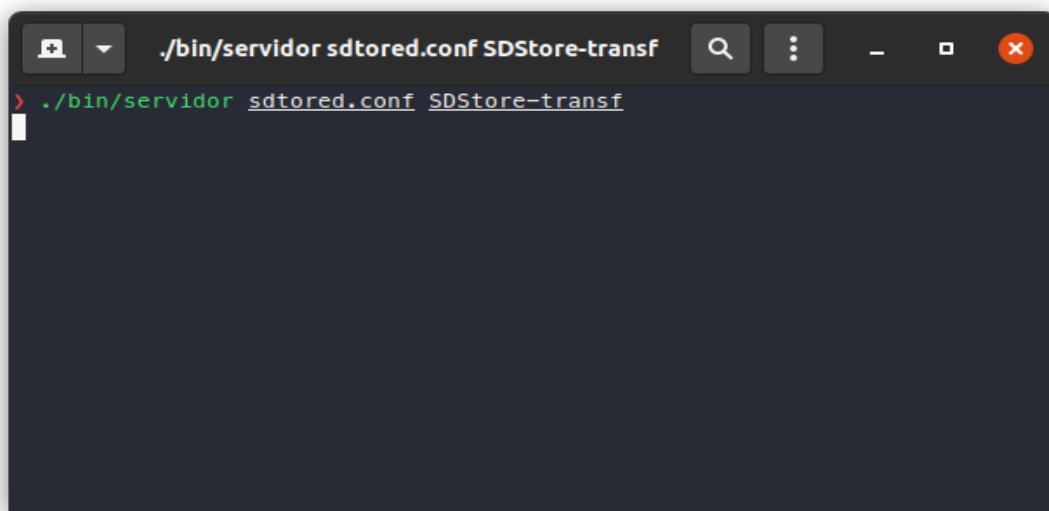


Fig 1: Inicializar Servidor

Ao criar o servidor irá ser criado um Pipe Publico onde este mesmo irá ler os pedidos de todos os clientes que efetuarem uma transformação.

Cliente:

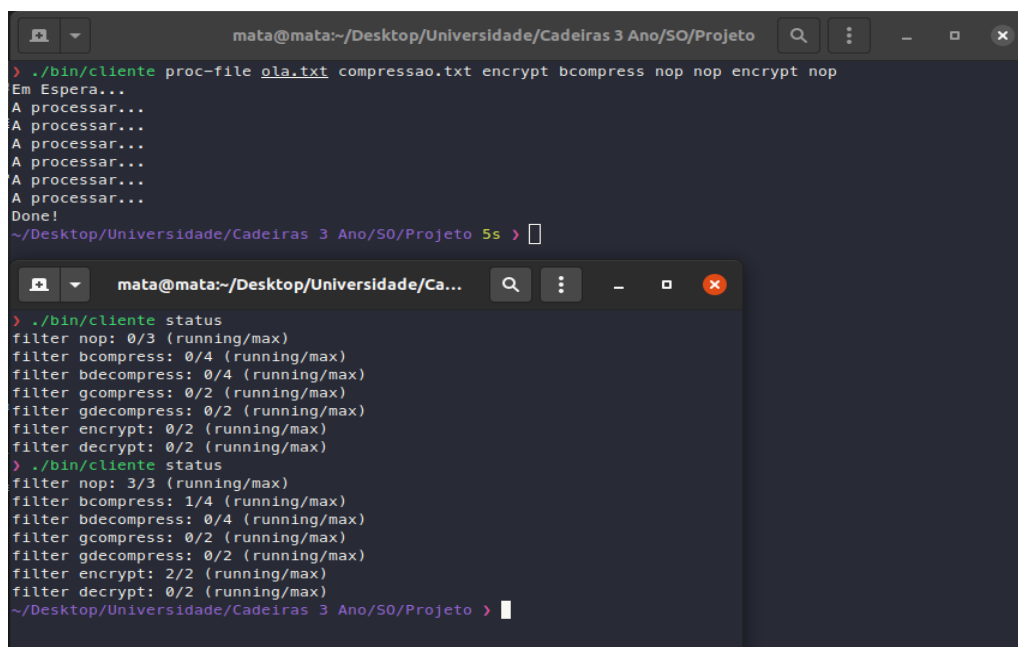
Num terminal diferente, para iniciarmos um pedido de um cliente executamos (terá de ser executado na pasta principal):

```
$ ./sdstore proc-file samples/file-a outputs/file-a-output  
trans-1 trans-2 trans-3 trans-4 trans-5 ...
```

Apartir do cliente, estamos a conectar ao pipe público, previamente criado pelo servidor, e a enviar toda a informação que queremos transformar

Também, utilizando o comando status, podemos verificar o estado do funcionamento do servidor, ou seja, podemos consultar o numero de transformações que estão a ser utilizadas no momento para cada tipo de transformação.

```
$ ./sdstore status
```



```
mata@mata:~/Desktop/Universidade/Cadeiras 3 Ano/SO/Projeto
> ./bin/cliente proc-file ola.txt compressao.txt encrypt bcompress nop nop encrypt nop
Em Espera...
A processar...
A processar...
A processar...
A processar...
A processar...
A processar...
Done!
~/Desktop/Universidade/Cadeiras 3 Ano/SO/Projeto 5s >

mata@mata:~/Desktop/Universidade/Ca...
> ./bin/cliente status
filter nop: 0/3 (running/max)
filter bcompress: 0/4 (running/max)
filter bdecompress: 0/4 (running/max)
filter gcompress: 0/2 (running/max)
filter gdecompress: 0/2 (running/max)
filter encrypt: 0/2 (running/max)
filter decrypt: 0/2 (running/max)
> ./bin/cliente status
filter nop: 3/3 (running/max)
filter bcompress: 1/4 (running/max)
filter bdecompress: 0/4 (running/max)
filter gcompress: 0/2 (running/max)
filter gdecompress: 0/2 (running/max)
filter encrypt: 2/2 (running/max)
filter decrypt: 0/2 (running/max)
~/Desktop/Universidade/Cadeiras 3 Ano/SO/Projeto >
```

Fig 2: Novo Pedido e Estado

Decisões Tomadas:

Inicialmente, ao inicializar o servidor, criamos um Pipe Público que irá fazer a conexão entre Cliente → Servidor, logo é necessário que seja executar primeiramente o servidor e nunca terminá-lo.

De Seguida, cada cliente irá efetuar o pedido desejado e irá enviar pelo Pipe Público referido anteriormente. Neste processo cliente, irá ser criado o respetivo pipe privado, com o nome pipe*pid*, pois cada pipe privado será único e a única maneira deste cliente obter uma resposta do servidor.

O servidor, ao obter o pedido do cliente, cria um fork e conecta-se ao pipe privado, pois recebeu do cliente o PID do próprio. De seguida, efetua as transformações requeridas. Nesta situação, utilizamos pipe anónimos para efetuar as transformações encadeadas. Inicialmente lemos do ficheiro de input, e escrevemos no pipe anonimo, consequentemente, iremos ler do mesmo pipe que escrevemos, evitando assim a criação de ficheiros temporários e tornando tudo mais simplificado. Quando aplicamos apenas um efeito nada disto será necessário, apenas iremos ler do ficheiro e de seguida gravar no novo.

De forma a que tudo funcione da maneira correta, por exemplo, estarem todas as transformações à disposição, colocamos todos os pedidos em espera até que possam correr todas as transformações necessárias.

Sempre que reiniciarmos o Servidor, temos que correr o comando *\$make clean* e *\$make all*.

Conclusão

Concluindo, podemos constatar que foi um bom projeto para desenvolver as nossas capacidades na cadeira Sistemas Operativos. De maneira que no futuro possamos implementar comunicação entre diferentes processos de forma a criar um ambiente funcional.