Universidade do Minho

Licenciatura em Ciências da Computação

SO - Trabalho Prático

Grupo 21

Breno Fernando G. Marrão (A97768)
Tales Andre M. Rovaris (A96314) Tiago Passos Rodrigues (A96414)

2021/2022



Conteúdo

Capítulo 1

Introdução

Este projeto consistiu na criação de um serviço de execução de transformações em ficheiros, sendo essas transformações feitas com a finalidade de compactar e encriptar, e o inverso. O cliente é capaz de enviar sucessivas transformações para um servidor executar, o cliente pode também consultar quais processos estão a ser executados e quais os limites das transformações.

Capítulo 2

Funcionalidades

2.1 Proc-File

O comando proc-file recebe a prioridade da tarefa, o ficheiro de entrada ,o ficheiro de saída, e as transformações que executaremos do ficheiro de entrada para o ficheiro de saída . O servidor avisa o cliente sobre o estado do processo, pending quando o servidor recebe o pedido, processing quando começa a executar as transformações e concluded com o número de bytes do ficheiro de entrada e quantos bytes o ficheiro de saída acabou por ficar.

```
breno@breno-550XCJ-550XCR:~/Documentos/Universidade/SO/Trabs/SistemasOperativos$ ./sdstore proc-file 5 bcopia bcopiacompressencrypt bcompress encrypt
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 349523)
```

2.2 Status

O comando Status permite o cliente saber o estado do servidor. Indica as transformações a serem executadas pelo servidor no momento, e o respetivo limite, e os pedidos a serem processados.

Capítulo 3

Servidor

O servidor quando inicializado cria um pipe com nome para a comunicação cliente-servidor chamado "contacto" , depois disso espera receber alguma comunicação através desse pipe. Quando recebe alguma comunicação , fazemos o parsing da informação recebida e armazenamos numa variavel com o tipo processo, esta é uma struct que criamos para ser mais facil armazenar a informação dos pedidos. Após a criação desse variável adicionamos a uma lista ligada que nomeamos de fila.

3.1 Struct Processo

```
typedef struct Processos* processos;
typedef struct processo processo;
struct processo{
    int id ;
    int prioridade;
    char** transformacoes; //char* transformacoes[9]
    int n_transformacoes;
    int procfile ;
    char* pid;
    int tamanho_original ;
    int tamanho_final;
};

struct Processos {
    processo data;
    processos next;
};
```

3.2 Fila e Exec

Decidimos armazenar as informações em uma lista ligada pois acreditamos que é mais facil para adicionar, remover e não ocupar espaço desnecessário, como seria caso usássemos um array.

Teremos duas variáveis: a fila e o exec, a fila são os processos que estão à espera para serem processados e o exec os que estão a ser processados atualmente. Decidimos tornar estas em duas variáveis globais para posteriormente podermos continuar a percorrer a fila no caso de se receber um sinal SIGTERM para terminar de forma graciosa.

Criamos a função addFila e addExec que adiciona um determinado processo à fila e ao exec pela sua prioridade. O removeFila e removeExec remove um determinado processo da fila e do exec.

Por último, a função checkFila vê qual o processo com a maior prioridade que não ultrapassa os limites de cada transformação e corre a função executa dentro de um filho que irá comunicar com o cliente que o pedido está "processing" e corre a função que executa as transformações.

Por fim, o servidor continua à espera de mais pedidos até receber o sinal SIGTERM, que invoca o sigterm_handler que verifica a fila até esta estar vazia, ou seja todos os pedidos foram concluídos e quando todos os processos a serem executados terem finalizado termina o servidor.

Capítulo 4

Cliente

O cliente vai fazer um pedido ao servidor através do terminal , o cliente vai adicionar a informação recebida pelos argumentos e o PID do próprio para um pipe de maneira que o servidor depois consiga comunicar com o cliente.

Tomamos a decisão de passar o PID pois é um número único para cada processo, e vai transmitir essa informação para o servidor através de um pipe com nome chamado “contacto”.

De seguida o cliente irá criar o pipe com o PID para o contacto servidor-cliente e esperar a comunicação do servidor, quando receber informação do final do pedido o processo termina.

Capítulo 5

Testes

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstored sdstored.c
onf SDStore-transf/
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore proc-file 1
 teste1 final1 bcompress encrypt gcompress gdecompress decrypt bdecompress
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 17651266)
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore proc-file 1 teste2 final2
bcompress encrypt gcompress gdecompress decrypt bdecompress
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 17651266)
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore proc-file 5
 teste3 final3 bcompress encrypt gcompress gdecompress decrypt bdecompress
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 17651266)
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore proc-file 2 teste4 final4 bc
ompress encrypt gcompress gdecompress decrypt bdecompress
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 17651266)
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore proc-file 3 tes
te5 final5 bcompress encrypt gcompress gdecompress decrypt bdecompress
Pending
Processing
Concluded (bytes-input: 17651266, bytes-output: 17651266)
```

```
tales@tales-MS-7C37:~/Desktop/Sistemas/SistemasOperativos$ ./sdstore status
Processo 7 teste1 final1 bcompress encrypt gcompress gdecompress decrypt bdecompress
Processo 8 teste2 final2 bcompress encrypt gcompress gdecompress decrypt bdecompress
transf bcompress: 2 / 4 (running/max)
transf bdecompress: 2 / 4 (running/max)
transf gcompress: 2 / 2 (running/max)
transf gdecompress: 2 / 2 (running/max)
transf encrypt: 2 / 2 (running/max)
transf decrypt: 2 / 2 (running/max)
transf nop: 0 / 3 (running/max)
```

Capítulo 6

Conclusão

Este projeto foi do mais essencial para compreender e consolidar os conhecimentos aprendidos nas aulas práticas sobre criação de processos filho, manuseamento de ficheiros , comunicação entre processos com pipes com nome e comunicação entre pai e filho com pipes anônimos e de sinais.