

Projeto de Sistemas Operativos

Carolina Silva, Cristina Pinto, João Fraga

June 6, 2019

Contents

1	Introdução	3
2	Estrutura do projeto	4
3	Cliente	5
3.1	Abordagem de cliente	5
3.2	Client	5
3.2.1	connect	5
3.2.2	get menu	5
3.2.3	do while	5
3.2.4	send options	7
3.2.5	parse output	7
3.2.6	read file [4]	7
4	Servidor	8
4.1	Threads [5]	8
4.2	Primeira comunicação	8
4.3	Main	9
4.4	Server	9
4.5	Protocolo de comunicação	9
4.6	Listas	10
4.6.1	funções auxiliares	10
4.6.2	server queue	10
4.7	Ficheiros	11
4.7.1	server file	11
5	Testes	12
6	Conclusão	13

List of Figures

1	Estrutura de ficheiros do projeto	4
2	Menu principal	5
3	Menu queue sem listas criadas	6
4	Menu queue com listas criadas	6
5	Esquema de threads	8
6	Comunicação inicial entre clientes e servidores [2] [1]	8
7	Protocolo de comunicação um cliente e o servidor	9

List of Tables

1	Mensagens a receber do servidor e as suas respectivas descrições	7
---	--	---

1 Introdução

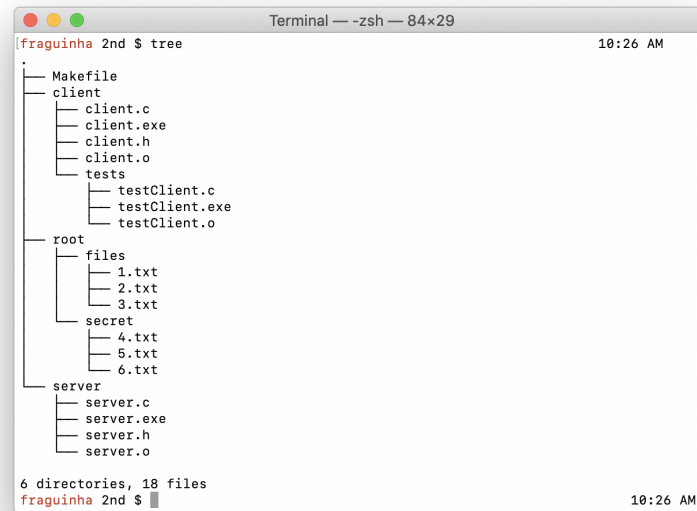
Este trabalho [3] tem como objetivo o desenvolvimento de uma estrutura de dados sincronizada para ser utilizada numa aplicação servidor e cliente. O projeto [3] tem também como objetivos o desenvolvimento de um servidor de ficheiros, que deverá usar multithreading para atender clientes em simultâneo, e uma aplicação para clientes, onde podem inserir, visualizar e apagar mensagens contidas em filas.

Assim, no servidor, foram desenvolvidos os ficheiros `server.h` (onde estão contidos todos os protótipos das funções associadas ao servidor) e `server.c` (onde está o programa principal do servidor e as funções referentes às threads, lists e ao funcionamento do próprio servidor e o menu principal).

No Cliente, foram desenvolvidos os ficheiros `client.h` (com os protótipos de funções) e `client.c` (onde se encontra o programa principal, bem como as funções `connect`, `getmenu`, `dowhile`, `sendoption` e `parseoutput`).

Como suporte para o desenvolvimento deste trabalho, recorreremos aos materiais fornecidos pelo docente da unidade curricular, bem como a bibliografia de referência sugerida para esta disciplina.

2 Estrutura do projeto



```
[fraguinha 2nd $ tree
.
├── Makefile
├── client
│   ├── client.c
│   ├── client.exe
│   ├── client.h
│   ├── client.o
│   └── tests
│       ├── testClient.c
│       ├── testClient.exe
│       └── testClient.o
├── root
│   ├── files
│   │   ├── 1.txt
│   │   ├── 2.txt
│   │   └── 3.txt
│   └── secret
│       ├── 4.txt
│       ├── 5.txt
│       └── 6.txt
└── server
    ├── server.c
    ├── server.exe
    ├── server.h
    └── server.o

6 directories, 18 files
fraguinha 2nd $
```

Figure 1: Estrutura de ficheiros do projeto

O projeto está dividido em 3 pastas principais: root, client e server.

A pasta root é a pasta raiz da funcionalidade de tratamento de ficheiros do nosso servidor. Esta pasta contém duas subpastas (files e secret) que por sua vez contém ficheiros sem, ou, com password respetivamente.

A pasta client contém todo o código relativo ao programa cliente e uma subpasta tests com código relativo aos testes automaticos.

A pasta server contém todo o código relativo ao programa servidor.

3 Cliente

3.1 Abordagem de cliente

Durante a realização do nosso projeto decidimos fazer do cliente um programa "dumb" que apenas interpreta mensagens do servidor e pede input ao utilizador, não efectuando nenhum processamento local.

3.2 Client

Este programa vai permitir a um cliente interagir com o servidor.



```
Terminal — client.exe — 80x24
[fraguinha 2nd $ ./client/client.exe 2:00 AM ]
Menu Client

1: files
2: queues
0: Exit

> █
```

Figure 2: Menu principal

3.2.1 connect

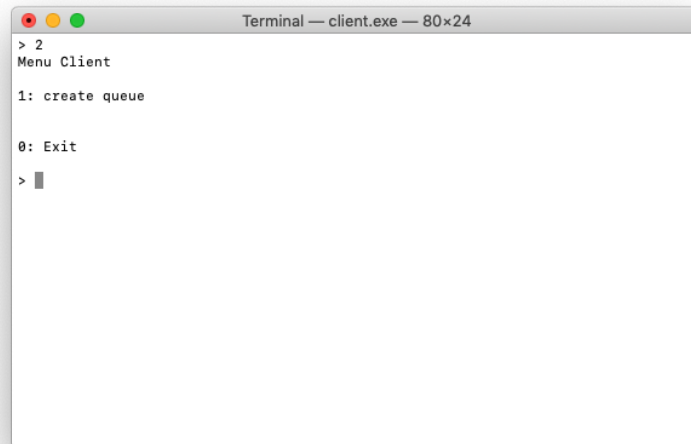
Vai permitir ao cliente aceder a informação do pipe individual que o servidor estabelece para ele.

3.2.2 get menu

Esta função vai permitir que o cliente tenha acesso ao menu inicial.

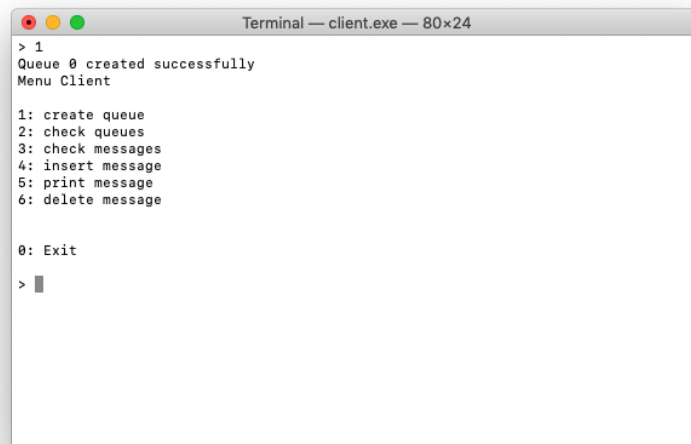
3.2.3 do while

Esta função vai imprimir continuamente menus e faz chamadas ao send options.



```
Terminal — client.exe — 80x24
> 2
Menu Client
1: create queue
0: Exit
> █
```

Figure 3: Menu queue sem listas criadas



```
Terminal — client.exe — 80x24
> 1
Queue 0 created successfully
Menu Client
1: create queue
2: check queues
3: check messages
4: insert message
5: print message
6: delete message
0: Exit
> █
```

Figure 4: Menu queue com listas criadas

3.2.4 send options

Vai estar a tratar continuamente o que vai ser necessário enviar ao cliente, fazendo uso de uma função auxiliar parse output.

3.2.5 parse output

Esta função vai fazer a interpretação de comandos enviados pelo servidor.

Table 1: Mensagens a receber do servidor e as suas respectivas descrições

Mensagens	Descrição
id	Esta mensagem é utilizada para pedir ao utilizador o id de uma fila
idp	Esta mensagem é utilizada para pedir o id de uma mensagem
filename	Esta mensagem é utilizada para pedir o nome de um ficheiro
message	Esta mensagem é utilizada para pedir uma mensagem para uma lista
password	Esta mensagem é utilizada para pedir a password dos ficheiros secretos
invalid	Esta mensagem é utilizada para indicar uma opção invalida
print	Esta mensagem é utilizada para indicar o envio de informação a imprimir
error	Esta mensagem é utilizada para indicar a ocorrência de um erro
file	Esta mensagem é utilizada para indicar o envio do conteúdo de um ficheiro a imprimir

3.2.6 read file [4]

Esta função vai ler o ficheiro pedido até ao fim.

4 Servidor

4.1 Threads [5]

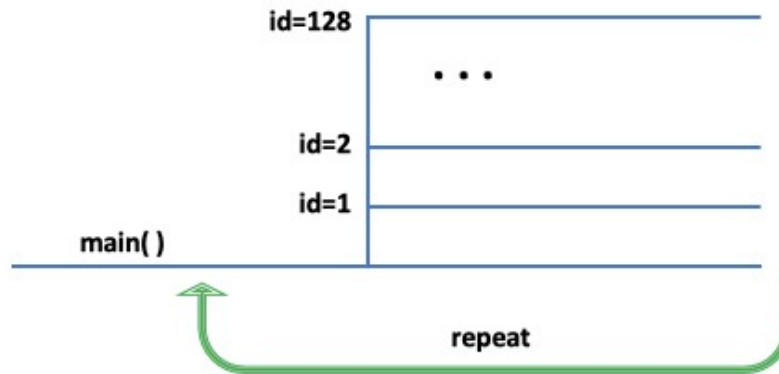


Figure 5: Esquema de threads

Neste trabalho efetuou-se uma abordagem de um servidor multithreaded, em que para cada pedido de conexão de um cliente é criada uma thread para servir o mesmo.

De modo a garantir o constante funcionamento do servidor foi necessário ignorar os broken pipes obtidos quando um cliente termina. Para isto utilizou-se a função `signal(SIGPIPE, SIG_IGN)`. [7] Assim foi possível realizar o unlink dos pipes estabelecidos com essa thread não causando problemas no servidor.

4.2 Primeira comunicação

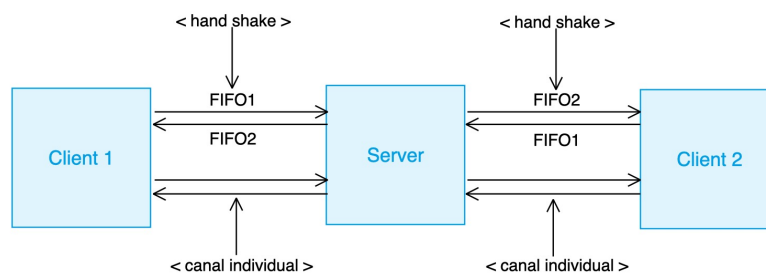


Figure 6: Comunicação inicial entre clientes e servidores [2] [1]

Inicialmente os clientes utilizam um pipe comum (`FIFO1` & `FIFO2`) para se

conectarem ao servidor e realizarem um handshake. O servidor utiliza este pipe para informar os clientes sobre quais os novos fifos a utilizar. Deste modo as comunicações realizadas entre um cliente e o servidor são efectuadas através de um pipe privado.

4.3 Main

Esta é a função principal do servidor a partir da qual o mesmo começa a execução. Aqui corre um loop infinito onde são descobertas conexões e criadas threads para lidar com as mesmas. [5]

4.4 Server

Esta função vai executar a thread individual de cada cliente, e permite que o cliente possa usufruir de todas as funcionalidades do servidor. O servidor terá a função de informar o cliente sobre as opções que este pode tomar e age de acordo.

4.5 Protocolo de comunicação

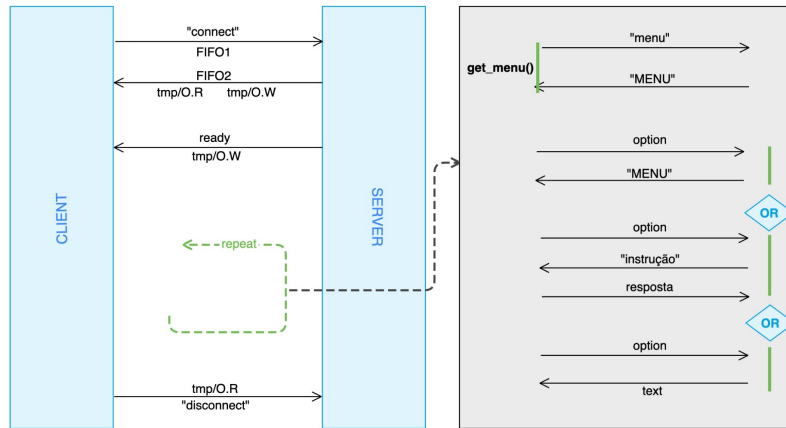


Figure 7: Protocolo de comunicação um cliente e o servidor

Após o handshake inicial e a criação de um canal de comunicação privado, a comunicação entre o cliente e o servidor inicia-se com um pedido do menu por parte do cliente. O servidor responde com o menu inicial.

Conforme as opções tomadas pelo utilizador o servidor poderá fornecer novos menus, enviar mensagens de resposta e solicitar o envio de informação.

4.6 Listas

4.6.1 funções auxiliares

Implementamos algumas funções auxiliares que serão utilizadas na execução de outras funções principais, nomeadamente:

- make node
- push
- pop
- is empty list
- print list
- count list
- print message list

4.6.2 server queue

Esta é a função a partir da qual são seleccionadas todas as funcionalidades referentes às filas. O servidor apenas apresenta as opções que se encontram disponíveis para utilização pelo cliente.

Todas estas funções fazem uso de um lock para se manterem sincronizadas.

[6]

1. create queue

Esta função vai permitir criar uma fila.

2. check queue

Esta função vai mostrar quais são os IDs das fila que existem.

3. check messages

Esta função vai permitir mostrar os IDs das mensagens de uma determinada fila.

4. insert message

Esta função vai verificar se o ID é válido e insere a mensagem na fila correspondente.

5. print message

Esta função imprime uma mensagem com um determinado ID de uma determinada lista.

6. delete message

Esta função vai apagar uma mensagem de uma determinada lista.

4.7 Ficheiros

4.7.1 server file

Esta é a função a partir da qual são executadas todas as funcionalidades referentes aos ficheiros. [4]

É solicitado ao cliente um nome de um ficheiro e caso este exista, e lhe enviada a informação contida no ficheiro. Se o ficheiro estiver na pasta secret, é requerido ao cliente a password.

1. print file

Esta função vai enviar para o cliente toda a informação referente a um determinado ficheiro.

5 Testes

Este programa vai permitir fazer testes para verificar a correta implementação do servidor de filas. Para isto, serão criadas várias threads para simular o acesso ao servidor por varios clientes. [5] Cada um destes clientes realizará várias das diferentes operações de filas em simultâneo de modo a validar a correcta implementação do servidor de filas multithreaded. [5]

6 Conclusão

O desenvolvimento deste projeto ajudou no aprofundamento de competências de Sistemas Operativos, nomeadamente no desenvolvimento, implementação e especificação de estruturas de dados clássica, utilização de listas e filas e implementações thread e multithread, bem como no desenvolvimento de um servidor aplicacional.

Assim, o desenvolvimento do projeto proposto na unidade curricular de Sistemas Operativos foi bastante enriquecedor, pois permitiu mobilizar os conteúdos apreendidos no decorrer das aulas teóricas e práticas.

References

- [1] Paul Crocker. Comunicação entre processos - 2ª parte (ipc - interprocess communication) [chapter 12]. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [2] Paul Crocker. Comunicação entre processos (ipc - interprocess communication) [chapter 11]. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [3] Paul Crocker. Enunciado do projecto. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [4] Paul Crocker. Ficheiros e directorias [chapter 10]. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [5] Paul Crocker. Programação com posix threads [chapter 9]. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [6] Paul Crocker. Sincronização, deadlock, starvation, mutex locks e semaphores (posix) [chapter 13]. *Universidade da Beira Interior : Sistemas Operativos*, 2018/2019.
- [7] Linux. <http://man7.org/linux/man-pages/man2/signal.2.html>. *Linux Programmer's Manual*, 2017.