



Instituto Superior de Engenharia de Coimbra

Engenharia Informática

Leandro Adão Fidalgo | a2017017144

Pedro dos Santos Alves | a2019112789

Sistemas Operativos

Trabalho Prático

Coimbra, 05 de janeiro de 2020

Índice

1. Introdução	3
2. Definições.....	4
2.1 Definições dos Comandos	4
3. Ficheiros utilizados.....	5
4. Makefile	5
5. Estruturas de dados	6
5.1 Estruturas de dados utilizadas	7
6. Funcionalidades.....	10
6.1 Gestor.....	10
6.1.1 Comandos administrador.....	10
6.1.2 O que o gestor permite?	14
6.2 Cliente	15
6.2.1 O que o cliente permite?.....	16
6.3 Interação automática Cliente-Servidor.....	18
7. Testes efetuados.....	19
7.1 Makefile	19
7.2 Leitura de variáveis de ambiente	19
7.3 Variáveis de linha de comando.....	19
7.4 Comando filter e verificador	19
7.5 Comando shutdown.....	19
7.6 Restantes comandos e funcionalidades.....	19
8. Conclusão	20

1. Introdução

O presente relatório descreve o projeto desenvolvido pelos alunos: Leandro Fidalgo e Pedro Alves, no âmbito da disciplina de Sistemas Operativos, na Licenciatura de Engenharia Informática do Instituto Superior de Engenharia de Coimbra. O propósito deste trabalho é consolidar todos os conhecimentos adquiridos no decorrer do semestre.

O objetivo do trabalho consiste na implementação de um sistema de gestão e redistribuição de mensagens denominado MSGDIST. As mensagens são simples mensagens de texto com as quais o sistema terá de se encarregar de as aceitar, armazená-las e redistribuí-las a quem estiver interessado nelas.

2. Definições

2.1 Definições dos Comandos

- CMD_DC – Comando enviado pelo cliente para avisar o servidor que o cliente se desconectou.
- CMD_SDC – Comando enviado pelo servidor para avisar os clientes que o servidor terminou.
- CMD_CON – Comando enviado pelo cliente para avisar o servidor que o cliente se quer desconectar.
- CMD_OK – Comando enviado após um comando CMD_CON. (Por enquanto)
- CMD_ERR – Comando enviado para transmitir mensagens de erro.
- CMD_FDC – Comando enviado pelo servidor para desconectar o cliente com o comando de administrador “kick”.
- CMD_NEWMSG – Comando enviado pelo cliente para enviar uma “mensagem”.
- CMD_INVMSG – Comando enviado pelo servidor para avisar o cliente que a “mensagem” é inválida.
- CMD_GETTOPICS – Comando enviado pelo cliente para avisar o servidor que quer uma lista de tópicos.
- CMD_GETTITLES – Comando enviado pelo cliente para avisar o servidor que quer uma lista de títulos.
- CMD_GETMSG – Comando enviado pelo cliente para avisar o servidor que quer a mensagem que contem um determinado “ID”.
- CMD_SUB – Comando enviado pelo cliente para avisar o servidor que se quer subscrever a um tópico.
- CMD_ALERTSUB – Comando enviado pelo servidor para avisar o cliente sobre uma nova mensagem que foi inserida num tópico, ao qual este está subscrito. (comando removido, pois foi utilizado o CMD_SUB para enviar o alerta de subscrição)
- CMD_UNSUB – Comando enviado pelo cliente para avisar o servidor que quer deixar de seguir um tópico.
- CMD_HEARTBEAT – Comando enviado para saber se o outro está “vivo”.
- CMD_ALIVE – Comando enviado para avisar que ainda está “vivo”.
- CMD_IGN – Comando que serve para ignorar.

3. Ficheiros utilizados

Foram criados 3 ficheiros header, tendo estes a definição dos tipos de dados e os protótipos das funções, em que o msgdist.h é o ficheiro que é partilhado entre o cliente e o gestor (ficheiro comum), o msgdist_c.h é o ficheiro associado ao cliente e o ficheiro msgdist_s.h é o ficheiro associado ao gestor.

Ainda foram criados 3 ficheiros de código, em que o msgdist_common.c é o ficheiro que contém o código comum entre o cliente e o gestor, o msgdist_client.c é o ficheiro que contém o código sobre o cliente e o msgdist_server.c que contém o código sobre o gestor. Ainda existe verificador.c que nos foi disponibilizado no moodle e o qual não pode sofrer alterações.

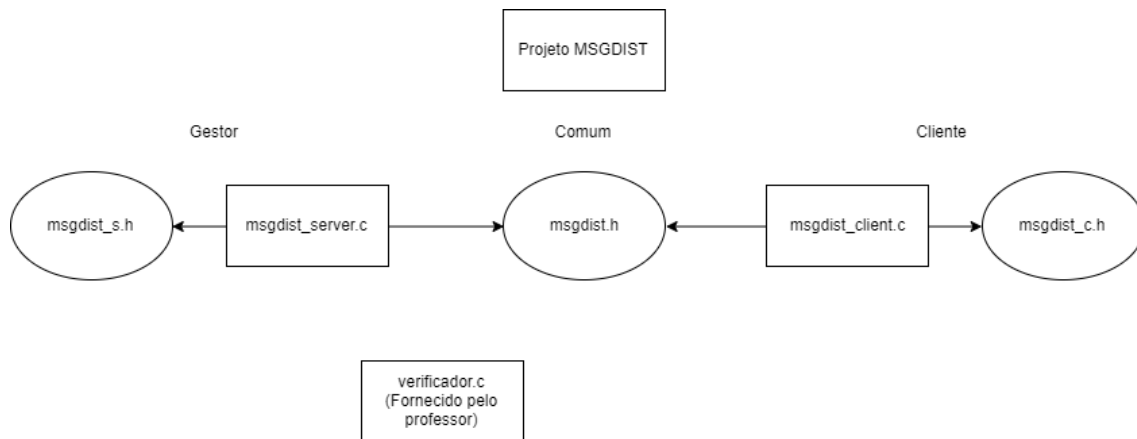


Figura 1 – Esquema dos ficheiros utilizados

4. Makefile

No makefile consta os comandos para a compilação de todos os ficheiros, ou um de cada vez (cliente, gestor, comuns e ainda do verificador) e remove os ficheiros objeto. Como funcionalidade adicional foi feito com que o comando make elimine os ficheiros binários caso seja necessário.

5. Estruturas de dados

O esquema abaixo representado demonstra a forma como o cliente e o gestor irão comunicar. O gestor cria o seu próprio FIFO (named pipe) com o nome msgsv, o qual serve para o cliente comunicar com o gestor. Cada cliente cria o seu próprio FIFO (named pipe) com o nome do seu PID (Process ID) e serve para o gestor comunicar com os clientes.

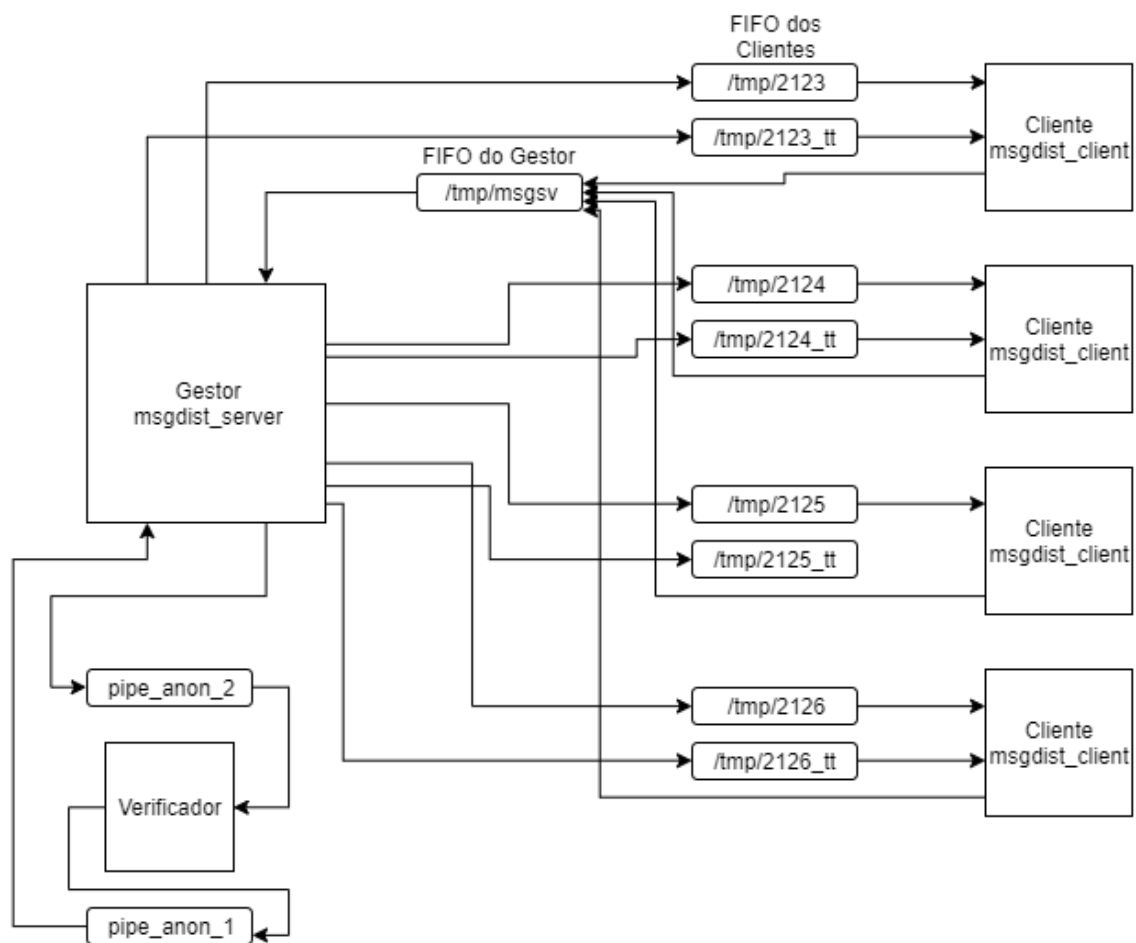


Figura 2 - Esquema comunicação gestor-cliente

5.1 Estruturas de dados utilizadas

A estrutura do COMMAND é uma estrutura de dados com um código, um remetente e um corpo de mensagem. Serve como base entre a comunicação do servidor e do cliente, indicando o código do comando (para saber do que se trata), o remetente (para saber de onde vem e a quem redirecionar) e o corpo de mensagem. Esta estrutura foi usada para facilitar a transferência de dados entre os clientes e o gestor.

```
typedef struct {  
    int cmd;  
    char From[MAX_FIFO];  
    CMD_UN Body;  
} COMMAND;
```

Figura 3 - Estrutura COMMAND

A estrutura do USER é uma estrutura de dados que contém o nome do cliente e o nome do FIFO associado a esse utilizador. Esta estrutura foi utilizada para enviar os dados do cliente como o “Username” para o gestor.

```
typedef struct {  
    char Username[MAX_USER];  
    char FIFO[MAX_FIFO];  
} USER;
```

Figura 4 - Estrutura USER

A estrutura MESSAGE é uma estrutura de dados que contém um autor, um título, um tópico, uma duração e o texto da mensagem. Esta estrutura foi utilizada para enviar as mensagens entre o cliente e o servidor.

```
typedef struct {  
    int Duration;  
    char Author[MAX_USER];  
    char Topic[MAX_TPCTTL];  
    char Title[MAX_TPCTTL];  
    char Body[MAX_BODY];  
} MESSAGE;
```

Figura 5 - Estrutura MESSAGE

A union CMD_UN é usada para enviar uma informação de um tipo de dado, por exemplo um “USER”, uma “MESSAGE” ou um “Topico”. Foi optado por utilizar uma union, pois o tamanho fica sempre com o tamanho do maior tipo de dados que estiver definido na union e dá para enviar um tipo de dado numa comunicação entre o cliente e gestor através da estrutura COMMAND.

```
typedef union {  
    USER un_user;  
    MESSAGE un_msg;  
    char un_topic[MAX_TPCTTL];  
    int un_tt;  
} CMD_UN;
```

Figura 6 - Union CMD_UN

A estrutura SV_CFG é uma estrutura de dados, que contém um file descriptor do FIFO do gestor, um verificador PID, um pipe anónimo (ler – 0/escrever - 1), um id do próximo cliente, o tamanho do Malloc para os clientes, uma lista de clientes, um id da próxima mensagem, uma lista de mensagens, um id do próximo tópico, o tamanho do Malloc para os tópicos, uma lista de tópicos, um variável para controlar o filtro para usar na configuração do gestor, um número máximo de mensagens que o servidor pode conter, um número máximo de palavras banidas e um nome do arquivo/caminho do arquivo de palavras proibidas. Esta estrutura foi utilizada para facilitar a configuração do gestor.

```
typedef struct {
    int sv_fifo_fd;
    int sv_verificador_pid;
    int sv_verificador_pipes[2];

    int next_uid;
    int users_size;
    SV_USER *users;

    int n_msgs;
    int next_mid;
    SV_MSG *msgs;

    int next_tid;
    int topic_size;
    SV_TOPIC *topics;

    int use_filter;

    int maxmsg;
    int maxnot;
    char wordsnot[PATH_MAX];
} SV_CFG;
```

Figura 7 - Estrutura SV_CFG

A estrutura SV_USER é uma estrutura de dados que contém um id do cliente, um file descriptor do cliente, um cliente, um inteiro para saber se o cliente está vivo, o tamanho do Malloc para os tópicos e o id dos tópicos subscritos. Esta estrutura foi utilizada para guardar os dados do cliente.

```
typedef struct {
    int id;
    int user_fd;
    USER user;
    int alive;
    int sub_size;
    int *topic_ids;
} SV_USER;
```

Figura 8 - Estrutura SV_USER

A estrutura SV_TOPIC é uma estrutura de dados que contém com o id do tópico e com um o nome do tópico. Esta estrutura foi utilizada para guardar os dados dos tópicos.

```
typedef struct {  
    int id;  
    char topic[MAX_TPCTTL];  
} SV_TOPIC;
```

Figura 9 - Estrutura SV_TOPIC

A estrutura SV_MSG é uma estrutura de dados que contém um id da mensagem e com a respetiva mensagem. Esta estrutura foi utilizada para guardar os dados das mensagens.

```
typedef struct {  
    int id;  
    MESSAGE msg;  
} SV_MSG;
```

Figura 10 - Estrutura SV_MSG

A estrutura CL_CFG é uma estrutura de dados que contém um file descriptor do fifo do gestor, um file descriptor do fifo do cliente, um segundo file descriptor do fifo do cliente que server para facilitar a troca de alguns dados, nomeadamente a lista de títulos e a lista de tópicos, o número de mensagens não lidas, o nome do fifo e o nome do segundo fifo do cliente, o nome do utilizador do cliente e por fim as janelas do interface do cliente. Esta estrutura foi utilizada para facilitar na configuração do cliente.

```
typedef struct {  
    int sv_fifo_fd;  
    int cl_fifo_fd;  
    int cl_fifo_tt_fd;  
    int cl_unr_msg;  
    char cl_fifo[MAX_FIFO];  
    char cl_fifo_tt[MAX_FIFO];  
    char cl_username[MAX_USER];  
  
    CL_WIN win[NUM_WIN];  
} CL_CFG;
```

Figura 11 - Estrutura CL_CFG

A estrutura CL_WIN é uma estrutura de dados que contém a janela da interface do cliente, tendo a sua altura e a sua largura. Esta estrutura serve para configurar a janela da interface do cliente.

```
typedef struct {  
    WINDOW *w;  
    int height;  
    int width;  
} CL_WIN;
```

Figura 12 - Estrutura CL_WIN

6. Funcionalidades

6.1 Gestor

6.1.1 Comandos administrador

O gestor disponibiliza diversas funcionalidades ao administrador. Para se saber quais as funcionalidades disponíveis, existe o comando “help”, o qual apresenta os diversos comandos e uma breve descrição acerca dos mesmos. De seguida é apresentada a lista dos comandos ao efetuar o comando “help”.

```
Admin > help  
Here's a list of all commands currently available:  
filter [on/off] - Turns message filtering on or off.  
users - Lists all users currently connected to the server.  
topics - Lists all topics currently stored on the server.  
msg - Lists all messages currently stored on the server.  
topic [topic_name] - Lists all messages with the topic_name.  
del [message_id] - Deletes the message associated with the message_id.  
kick [user_id] - Kicks the user associated with the user_id.  
prune - Deletes topics with no messages associated and cancel subscriptions to those topics.  
verify [message] - Uses the 'verificador' program to verify the message for banned words.  
cfg - Shows the server's configuration.  
view [message_id] - View message with the message_id.  
subs [user_id] - View subscriptions from the user with the user_id.  
shutdown - Shuts the server down.  
exit - Same as shutdown.  
help - Shows this.  
Admin > _
```

Figura 13 - Lista de Comandos do Gestor

O comando “filter” pode ser utilizado com ou sem argumentos. Caso o administrador não forneça argumentos, o gestor indica se o filtro está ligado ou desligado. Caso o administrador forneça argumentos “on/off”, o filtro irá atribuir o ligado ou desligado conforme o argumento. Este comando serve para ligar/desligar o filtro que irá utilizar o “verificador” para verificar as mensagens.

```
Admin > filter  
Filter is enabled.  
To use 'filter', try 'filter [on/off]'  
Admin > filter on  
Filter is already enabled.  
Admin > filter off  
Disabled word filter.  
Admin > filter on  
Enabled word filter.  
Admin > filter wadoiahw  
Unknown command...  
Admin > _
```

Figura 14 - Comando “filter”

O comando “users” apresenta uma lista dos utilizadores que estão atualmente conectados ao gestor.

```
Admin > users daw
Unknown command...
Admin > users
Current number of users: 0
Admin > [INFO] User 'User XPT0' has connected.
Admin > users
Current number of users: 1
ID: 1 Username: User XPT0 FIFO: /tmp/2015
Admin > [INFO] User 'User XPT0(0)' has connected.
Admin > users
Current number of users: 2
ID: 1 Username: User XPT0 FIFO: /tmp/2015
ID: 2 Username: User XPT0(0) FIFO: /tmp/2191
Admin >
```

Figura 15 - Comando “users”

O comando “topics” apresenta uma lista de todos os tópicos que estão armazenados no gestor.

```
Admin > topics
Current number of topics: 0
Admin > [INFO] Topic 'Topic XPT0' does not exist therefore it was added to the server.
Admin > [INFO] User 'User XPT0' added a message to the server with 0 bad words.
Admin > topics
Current number of topics: 1
ID: 1 Topic: Topic XPT0
Admin > topics idwaodw
Unknown command...
Admin > _
```

Figura 16 - Comando “topics”

O comando “msg” apresenta uma lista de todas as mensagens que estão armazenadas no gestor.

```
Admin > msg odiwa
Unknown command...
Admin > msg
Current number of messages: 1
ID: 1 Title: Title XPT0 Author: User XPT0 Time left: 1410065331 seconds
Admin > _
```

Figura 17 - Comando “msg”

O comando “topic” necessita de um argumento, sendo este o nome do tópico. Este comando apresenta uma lista de todas as mensagens com esse nome do tópico.

```
Admin > topic
Incomplete command.
To use 'topic', try 'topic [topic_name]'
Admin > topic Topic
[WARNING] Invalid topic name.
Admin > topic Topic XPT0
Messages with topic 'Topic XPT0':
ID: 1 Title: Title XPT0 Author: User XPT0 Time left: 1410065178 seconds
Admin >
```

Figura 18 - Comando “topic”

O comando “del” necessita de um argumento, sendo este o id da mensagem a eliminar. Este comando elimina a mensagem com esse id enviado por argumento.

```
Admin > del
Incomplete command.
To use 'del', try 'del [message_id]'
Admin > del 0
[WARNING] Invalid ID.
Admin > del -1
[WARNING] Invalid ID.
Admin > del 1
[INFO] Message deleted.
Admin >
```

Figura 19 - Comando "del"

O comando “kick” necessita de um argumento, sendo este o id do utilizador que pretender forçar a sua desconexão.

```
Admin > kick
Incomplete command.
To use 'kick', try 'kick [user_id]'
Admin > kick 0
[WARNING] Invalid ID.
Admin > kick 1
[INFO] User kicked.
Admin > users
Current number of users: 1
ID: 2 Username: User XPT0(0) FIFO: /tmp/2682
Admin > _
```

Figura 20 - Comando "kick"

O comando “prune” não necessita de argumentos e serve para apagar os tópicos que não têm mensagens associadas e cancelar as subscrições a esses tópicos.

```
Admin > prune kjdwa
Unknown command...
Admin > prune
[Server] The prune command deleted 1 topics.
[Server] The prune command canceled 1 subscriptions.
Admin >
```

Figura 21 - Comando "prune"

O comando “verify” necessita de um argumento, sendo este um texto. Este comando serve para utilizar o programa “verificador” para verificar o número de palavras banidas.

```
Admin > verify
Incomplete command.
To use 'verify', try 'verify [text]'
Admin > verify Ola, esta mensagem de teste serve para demonstrar o comando a funcionar. 22
[Server] Bad word count: 1
[Server] Valid message.
Admin >
```

Figura 22 - Comando "verify"

O comando “cfg” não necessita de argumentos e este comando mostra a configuração do servidor.

```
Admin > cfg dwaiuo
Unknown command...
Admin > cfg
      MAXMSG: 50
      MAXNOT: 5
      WORDSNOT: bannedwords
      Current users: 1
      Current topics: 0
      Current messages: 0
      Server PID: 2674
      'Verificador' PID: 2675
Admin > _
```

Figura 23 - Comando “cfg”

O comando “view” necessita de um argumento, sendo este o id da mensagem que deseja ver.

```
Admin > view
Incomplete command.
To use 'view', try 'view [message_id]'
Admin > view 0
[WARNING] Invalid ID.
Admin > view -1
[WARNING] Invalid ID.
Admin > view 1
      ID: 1
      Time left: 1410065018
      Author: User XPT0
      Topic: Topic XPT0
      Title: Title XPT0
      Body:
Hello there!
- XPT0 Master
Admin > _
```

Figura 24 - Comando “view”

O comando “subs” necessita de um argumento, sendo este o id do utilizador. Este comando mostra as subscrições que esse utilizador contém.

```
Admin > subs
Incomplete command.
To use 'subs', try 'subs [user_id]'
Admin > subs 0
[WARNING] Invalid ID.
Admin > subs 1
      User is subscribed to 0 topics.
Admin > subs 2
      User is subscribed to 1 topics.
      ID: 1 - 'Topic XPT0'
Admin > _
```

Figura 25 - Comando “subs”

O comando “shutdown” e “exit” não necessita de argumentos e serve para desligar o gestor e avisar todos os clientes conhecidos que o gestor se vai desligar.

```
Admin > shutdown
[Server] Server shutting down. Please wait.
[INFO] Named pipe closed.
[INFO] Named pipe deleted.
@server:~/_Shared/TrabPrat$ _
```

Figura 26 - Comando “shutdown”

6.1.2 O que o gestor permite?

O gestor mantém a informação acerca de que clientes e utilizadores se encontram a interagir com ele, a partir do momento em que o cliente se conecta ao gestor. Assim que os clientes se desconectarem o gestor deixará de ter a informação acerca dos clientes.

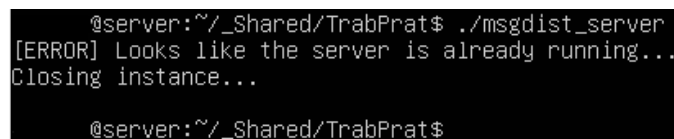
O gestor recebe mensagens enviadas pelos clientes e armazena-as durante o tempo indicado no campo “duração” da mensagem e elimina-as automaticamente quando a duração se esgotar. O número máximo de mensagens a armazenar é fixo durante a execução do gestor e é fornecido pelo valor da variável de ambiente MAXMSG. Este mantém ainda a informação acerca dos tópicos das mensagens enviadas pelos clientes. Cada mensagem recebida pelo gestor é analisada com o auxílio do programa independente “verificador”, que recebe o corpo da mensagem e identifica o número de palavras proibidas nele existente. Se o número de palavras proibidas exceder o valor indicado na variável de ambiente MAXNOT, então a mensagem é rejeitada e o cliente é avisado. As palavras proibidas estão no ficheiro de texto cujo nome está na variável de ambiente WORDSNOT.

O gestor mantém a informação acerca de que tópicos estão subscritos por que clientes/utilizadores. Assim que os utilizadores cancelarem a subscrição, o gestor descarta essa informação. Este avisa os clientes acerca da existência de novas mensagens recebidas nos tópicos por eles subscritos.

O gestor deteta automaticamente que o cliente deixou de existir com uma precisão com cerca de 10 segundos.

Este ainda reporta de imediato no “stderr” todas as ações feitas que alteram a lista de mensagens ou tópicos, e as ações que alteram o conjunto de clientes/utilizadores.

Apenas deve existir um processo gestor em execução. Ao detetar uma nova instância termina de imediato.



```
@server:~/_Shared/TrabPrat$ ./msgdist_server
[ERROR] Looks like the server is already running...
Closing instance...

@server:~/_Shared/TrabPrat$
```

Figura 27 - Dupla instância do gestor

6.2 Cliente

O utilizador indica o seu username ao programa cliente como argumento da linha de comandos. Caso exista um username com esse mesmo nome, o gestor acrescenta um número ao username e avisa o cliente.

```
@server:~/_Shared/TrabPrat$ ./msgdist_client
You need to specify a username!
Try ./msgdist_client [username]
@server:~/_Shared/TrabPrat$ ./msgdist_client "User XPT0"
Username has been changed due to username being already in use.
New username: User XPT0(0).
Press enter key to continue...
```

Figura 28 - Inicialização do cliente

Para o cliente foi desenvolvida uma interface em ncurses, com o objetivo de melhorar a interação do cliente com o utilizador.



Figura 29 – Interface do Cliente

6.2.1 O que o cliente permite?

O cliente permite enviar uma nova mensagem ao gestor, especificando todos os campos. Esses campos são: a duração da mensagem até ser descartada pelo gestor, o tópico ao qual a mensagem está associada, o título da mensagem e o corpo da mesma.



Figura 30 - Interface do Cliente (nova mensagem)

O cliente poderá consultar a lista dos tópicos atualmente existentes, tenham os tópicos mensagens ou não.



Figura 31 - Interface do Cliente (ver tópicos)

O cliente poderá ainda consultar a lista de títulos de todas as mensagens.



Figura 32 - Interface do Cliente (ver títulos)

O cliente pode consultar uma mensagem, com um determinado id, podendo ver assim o seu autor, o seu tópico, o seu título e o corpo da mensagem.



Figura 33 - Interface do Cliente (ver mensagem 1)

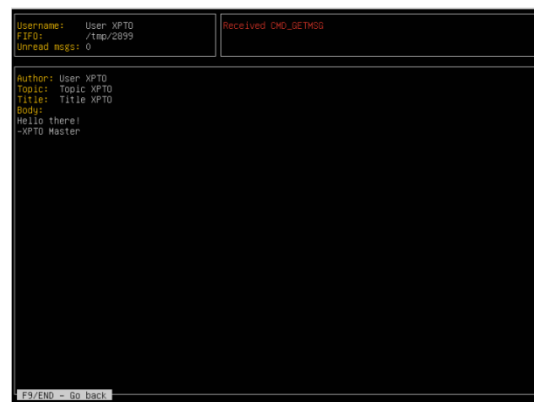


Figura 34 - Interface do Cliente (ver mensagem 2)

O cliente pode subscrever ou cancelar uma subscrição de um tópico.



Figura 35 - Interface do Cliente (subscrever/cancelar subscrição 1)



Figura 36 - Interface do Cliente (subscrever/cancelar subscrição 2)

6.3 Interação automática Cliente-Servidor

O cliente recusar-se-á a prosseguir a sua execução se perceber que o gestor não está em execução.

```
@server:~/_Shared/TrabPrat$ ./msgdist_client "User XPT0"  
Server is not online...  
Please try another time.  
@server:~/_Shared/TrabPrat$ _
```

Figura 37 – Gestor Offline

Quando o utilizador encerra o seu processo cliente, este deverá informar o gestor.

```
Admin > [INFO] User 'User XPT0' has disconnected.  
Admin >
```

Figura 38 - Utilizador desconnectou-se

O gestor, ao terminar, informará os clientes conhecidos, e que se encontram em execução, que vai terminar. Os clientes deverão também terminar, informando o utilizador.

```
Server has shutdown.  
@server:~/_Shared/TrabPrat$
```

Figura 39 - Gestor terminou

7. Testes efetuados

7.1 Makefile

Para efetuar os testes ao makefile foi utilizado o comando make sem nenhum parâmetro e verificou-se que este executa a primeira instrução (all), depois foi experimentado o make com as diversas instruções e foi verificado que todas elas funcionaram corretamente.

7.2 Leitura de variáveis de ambiente

Para efetuar os testes com as variáveis de ambiente foi executado o servidor sem mexer nas variáveis de ambiente e este definiu os valores predefinidos para essas variáveis, de seguida foi utilizado o comando export para cada uma das variáveis de ambiente e foi constatado que estavam a funcionar corretamente.

7.3 Variáveis de linha de comando

Para efetuar os testes com as variáveis de linha de comandos, o cliente foi executado sem argumentos e o programa cliente ensina o utilizador a iniciar o programa, de seguida foi colocado um argumento e esse argumento foi definido como “username” do cliente que iniciou o programa.

7.4 Comando filter e verificador

Para efetuar os testes do comando filter e do verificador foi utilizado um comando (verify) para facilitar os testes de ligação com o verificador, de seguida foi ligado o filtro e verificou-se que o verificador devolvia o número de palavras banidas e por fim foi desligado o filtro e foi constatado que o verificador já não devolvia o número de palavras banidas.

7.5 Comando shutdown

Para efetuar os testes ao comando shutdown, foi verificado que o comando shutdown avisa todos os utilizadores conhecidos que o gestor se irá desligar, de seguida termina todas as threads auxiliares e fecha a ligação com os FIFO's e elimina os ficheiros dos mesmos, depois desliga o verificador e por fim termina o processo.

7.6 Restantes comandos e funcionalidades

Os restantes comandos e funcionalidades foram testados e provados que estavam a funcionar corretamente com base no que foi dito anteriormente neste relatório.

8. Conclusão

O principal objetivo deste projeto é colocar em prática todos os conhecimentos adquiridos na disciplina de Sistemas Operativos.

Todas as funcionalidades obrigatórias de implementação para a meta final foram realizadas com sucesso.

A realização deste projeto permitiu consolidar diversas competências nomeadamente: threads, mutexes, ncurses, pipes anónimos e named pipes.

Durante o processo de realização deste projeto foram surgindo algumas dúvidas, que conseguiram ser superadas através do auxílio do professor da disciplina e das fichas de trabalho por ele disponibilizadas.