Wboard - Documentation

Pierfrancesco Conti

Computer System and Programming

Professor: Giorgio Richelli

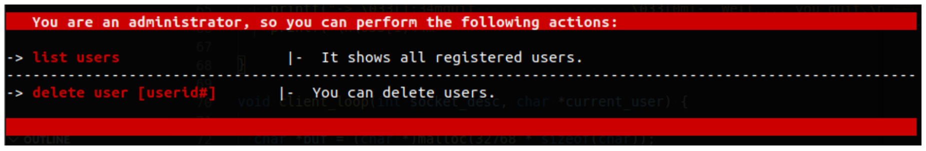
June 12, 2020

1 Introduction

Wboard is a whiteboard application designed for Linux terminal and developed by Pierfrancesco Conti using the C programming language for the Computer Systems and Programming course.

2 Design Choices

The Whiteboard structure is contained in a shared memory and is composed by the list of Users and the list of Topics (both set as shared memories). Among Users, **admin** is the only one with administrator privileges and can be authenticated with the password **admin**. It can perform actions on Users such as list or delete them.



```
  You are an administrator, so you can perform the following actions:

-> list users              |-  It shows all registered users.
------------------------------------------------------------------------------------------
-> delete user [userid#]   |-  You can delete users.
```

Figure 1: Administrator's view: Help command

The structure of each Topic was designed in the way described as following:

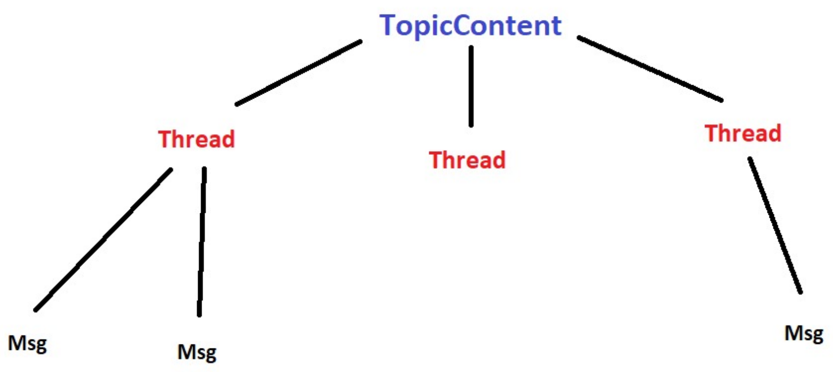
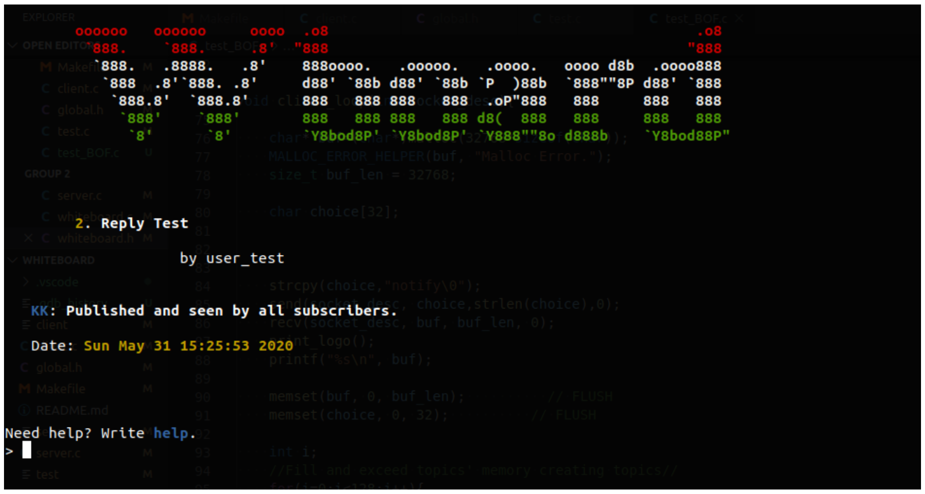


Figure 2: Topic's structure

Each Topic contains a list of Comments related to it. Each Comment is designed as another shared memory and contains the field *in_reply_to*, that is the integer which refers to the parent Comment. If this field is *-1*, then the Comment has no parent and it is automatically highlighted as a Thread. A Message is a Comment that is *in_reply_to* a Thread or to another Message.

It was chosen not to implement the **get** [**comment#**] command, because it is not necessary due to the fact that all messages are listed by the **topic** [**topic#**] command and the message can be displayed with all its info with the command **status** [**comment#**].



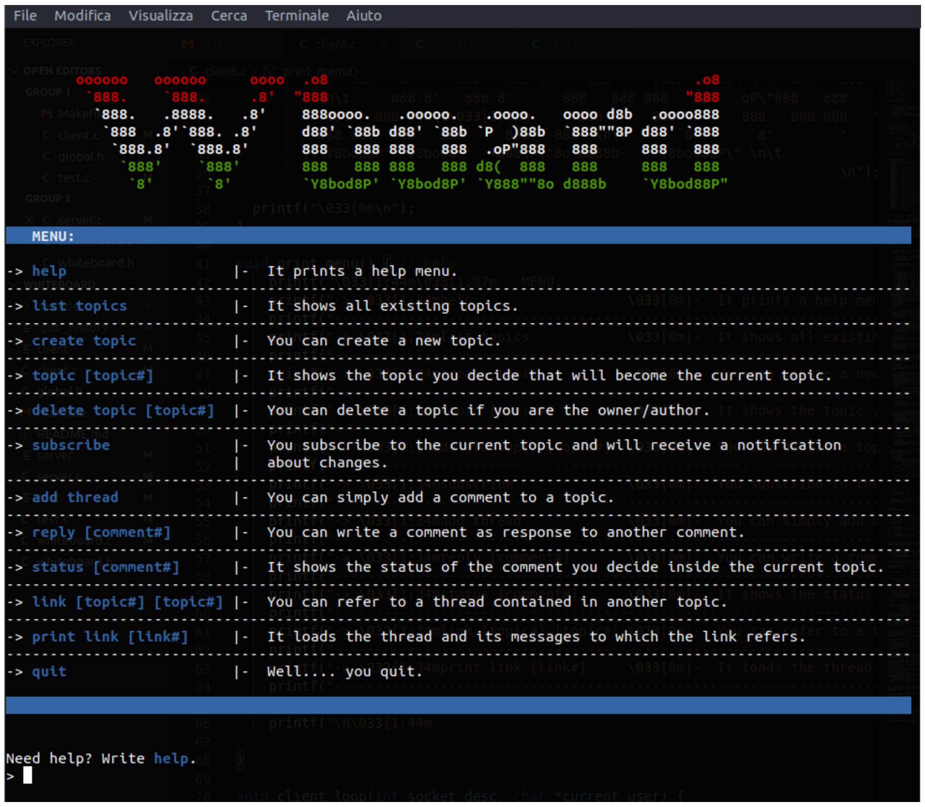
```
        oooooo   oooooo     oooo  .o8                                            .o8
         `888.    `888.     .8'  "888                                           "888
          `888.    .8888.   .8'    888oooo.   .ooooo.   .oooo.   oooo d8b  .oooo888
           `888   .8'`888. .8'     d88' `88b d88' `88b `P  )88b  `888""8P d88' `888
            `888.8'   `888.8'      888   888 888   888  .oP"888   888     888   888
             `888'     `888'       888   888 888   888 d8(  888   888     888   888
              `8'       `8'        `Y8bod8P' `Y8bod8P' `Y888""8o d888b    `Y8bod88P"


        2. Reply Test

                    by user_test


   KK: Published and seen by all subscribers.

   Date: Sun May 31 15:25:53 2020



Need help? Write help.
> 
```

Figure 3: Status message

Furthermore, it was chosen not to protect the size of the buffer with the empty/full semaphores, in order not to block the execution of the client that was trying to exceed this size. However another control was used to prevent the overflow and can be tested by executing the **./test_BOF** command. Finally, the subscription to a Topic allows a User to view its Comments and perform actions on it. Thanks to the subscription, after each authentication, the application notifies the user whether a new Comment is added to the subscripted Topic while it is not logged in.

3 Usage

The application can be compiled by the **make** command and started by **./server**. Each client can be executed by the **./client** command (from localhost by default). Automatically, the client communicates with the server's port that is listening for incoming connections. Now you can choose between the **register** and **authenticate** commands. After authentication, you are in the application and you can type **help** to list all the commands you could insert.



```
   oooooo   oooooo     oooo  .o8                                        .o8
    `888.    `888.     .8'  "888                                       "888
     `888.   .8888.   .8'    888oooo.   .ooooo.   .oooo.   oooo d8b  .oooo888
      `888  .8'`888. .8'     d88' `88b d88' `88b `P  )88b  `888""8P d88' `888
       `888.8'  `888.8'      888   888 888   888  .oP"888   888     888   888
        `888'    `888'       888   888 888   888 d8(  888   888     888   888
         `8'      `8'        `Y8bod8P' `Y8bod8P' `Y888""8o d888b    `Y8bod88P"

 MENU:

-> help                  |-  It prints a help menu.
-------------------------------------------------------------------------------
-> list topics           |-  It shows all existing topics.
-------------------------------------------------------------------------------
-> create topic          |-  You can create a new topic.
-------------------------------------------------------------------------------
-> topic [topic#]        |-  It shows the topic you decide that will become the current topic.
-------------------------------------------------------------------------------
-> delete topic [topic#] |-  You can delete a topic if you are the owner/author.
-------------------------------------------------------------------------------
-> subscribe             |-  You subscribe to the current topic and will receive a notification
                         |   about changes.
-------------------------------------------------------------------------------
-> add thread            |-  You can simply add a comment to a topic.
-------------------------------------------------------------------------------
-> reply [comment#]      |-  You can write a comment as response to another comment.
-------------------------------------------------------------------------------
-> status [comment#]     |-  It shows the status of the comment you decide inside the current topic.
-------------------------------------------------------------------------------
-> link [topic#] [topic#] |-  You can refer to a thread contained in another topic.
-------------------------------------------------------------------------------
-> print link [link#]    |-  It loads the thread and its messages to which the link refers.
-------------------------------------------------------------------------------
-> quit                  |-  Well.... you quit.


Need help? Write help.
> 
```

Figure 4: Client's view - Help command

There is a list of topics that compose the whiteboard (see them by the **list topics** command) and each of them is made-up by an ID, a title, its author, its content and all the comments referred to it. Some of these fields are hidden to better organize the view, but you can see them by choosing the topic you are interested in by typing **topic [topic#]** (i.e. **topic 3** to see info about the topic with id=3). The one you have chosen will be set as the current one and you can perform actions on it: subscribe to it, add a comment or other actions on its comments.

Ways to add comments: **add thread** to append a new thread to the current topic and **reply [comment#]** to append a new message to any existing comment inside the topic). It is also possible to link a Thread contained in another Topic, by executing the command **link [topic#] [thread#]** and you can follow this link by typing **print link [link#]** Type **status [comment#]** to show some info about the comment you decided. To quit, type **quit**. ATTENTION, this is not an exhaustive description: for further commands, please refer to the **help** command.

4 Code

4.1 Macro modules and their interaction

The code is composed by the files described by the Table 1. *whiteboard.c*, *whiteboard.h* and *server.c* are the ones used to manage the application by server-side, while *client.c*, *test_BOF.c* and *test.c* are different type of clients, used to interact with the server through pre-formatted commands.

Wboard	
File names	**Description**
whiteboard.c/.h	They contain the whole Whiteboard's structure and the relative functions to manage it server-side.
server.c	It forks a process for each connected client, calls Whiteboard's functions and sends back the result.
client.c	After authentication process, a client chooses a command to send to the server and receives the response.
test.c	Automated client that rapidly interacts with the server to test its responses.
test_BOF.c	Automated client that rapidly tries to exceed Wboard buffers' sizes and shows the results.
global.h	Simple library shared by both clients and server useful for errors handling and providing connection info.

Table 1: Files and their utilities

4.2 Whiteboard structure

- Whiteboard:
  - List of Topics.
  - List of Users registered to the application.
- Topic:
  - ID number: to be referred by app's functionalities.
  - Title.
  - Author.
  - Content.
  - List of Comments.
  - List of Links.
  - Timestamp.
  - List of subscribed users.
  - Pointer to next topic.
- User:
  - ID number.
  - Username
  - Password
  - Pointer to next user.
- Subscribers' pool:
  - User ID.
  - List of Topics to which the user is subscribed.
  - Pointer to next item.
- Comment:
  - ID number: to be referred by app's functionalities.
  - Reply-id: ID number of the comment to which it replies.
  - List of comments that reply to this.
  - Comment's Status.
  - Author.
  - Content.
  - Timestamp.
  - ID of the Comment to which the current Comment is a reply (-1 if none, so the current comment is a Thread).
  - IDs of replies to the current Comment.
  - IDs of Users that see the current Comment.
  - Pointer to next Comment.
- Link:
  - ID number: to be referred by app's functionalities.
  - ID of the Topic of the Thread to which the Link is referred.
  - ID of the Thread to which the Link is referred.
  - Pointer to the next Link.

5 Test cases

Tests are performed by the *test* client. It executes some client's commands in sequence, in order to output server's responses and let the programmer quickly check their results. In particular the following commands are executed:

- help
- create topic
- list topics
- topic 0 - ERROR: cannot see comments. [1]
- add thread - ERROR: action not permitted. [1]
- subscribe
- add thread
- reply 0
- topic 0
- topic 1
- link 1 0
- link 99 99 - ERROR: topic/thread not found. [1]
- topic 1
- quit

For readability reasons, some commands have been removed and the only way to test them is to run **./client** and try them out manually. Furthermore the test user does not have administrative privileges, so it cannot perform actions on users such as **list users** and **delete users**.

Another test that was carried out, is about the resistance of the application on attempts to exceed the size of the most important buffers such as Topics, Users and Comments. This test can be performed by executing the client **./test_BOF**. These buffers will be filled and the results will be printed on stdout.

[1]These ERRORs occur when executed **./test** command at first. Their presence represent a CORRECT behaviour of the application.

6 Last Changes

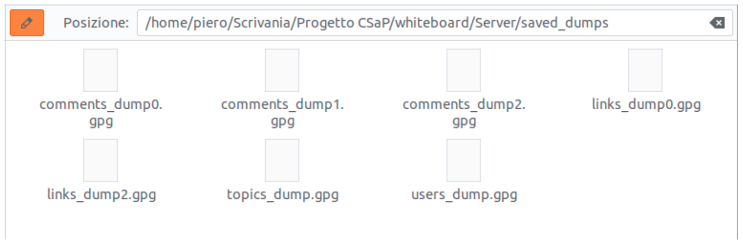
About the second release of this application, this chapter reports further functionalities that have been implemented.

6.1 Whiteboard Saving & Loading

Thanks to Saving and Loading procedures, the whiteboard can preserve its data after the server crashes or shuts down. This can be done by writing each object inside a file, using the function *save_wb*(), and reading it after the server restarts, thanks to *load_wb*()). To reload all the users inside the whiteboard, it was created a new simple server-side client, that is automatically executed by the application, to perform quick registration procedures (**./loader**).

6.2 Data encryption

The files, created through data saving procedure, are instantly encrypted, so no one can harm the integrity of the whiteboard after accessing the server. In order to load this data, files are decrypted (to be readable by the application) and immediately re-encrypted. In particular, the encryption/decryption in PGP was made getting help from the **gpg** Unix tool. Unfortunately, due to the quantity of the files to encrypt/decrypt, these procedures may affect application performance. To get all files into the directory, the library *dirent.h* was employed.



Posizione: /home/piero/Scrivania/Progetto CSaP/whiteboard/Server/saved_dumps

Figure 5: Encrypted files inside the "saved_dumps" directory.

6.3 Corrections to Shared Memory usage

Now the Shared memory is only attached to the parent process before the forks, during the initial creation of the Whiteboard, in order to correctly inherit the memory addresses from child processes. To make this possible, the list of Comments has been made static within the Topic's structure.

6.4 Added comments to the code

Further comments have been added, in order to make the code more readable.