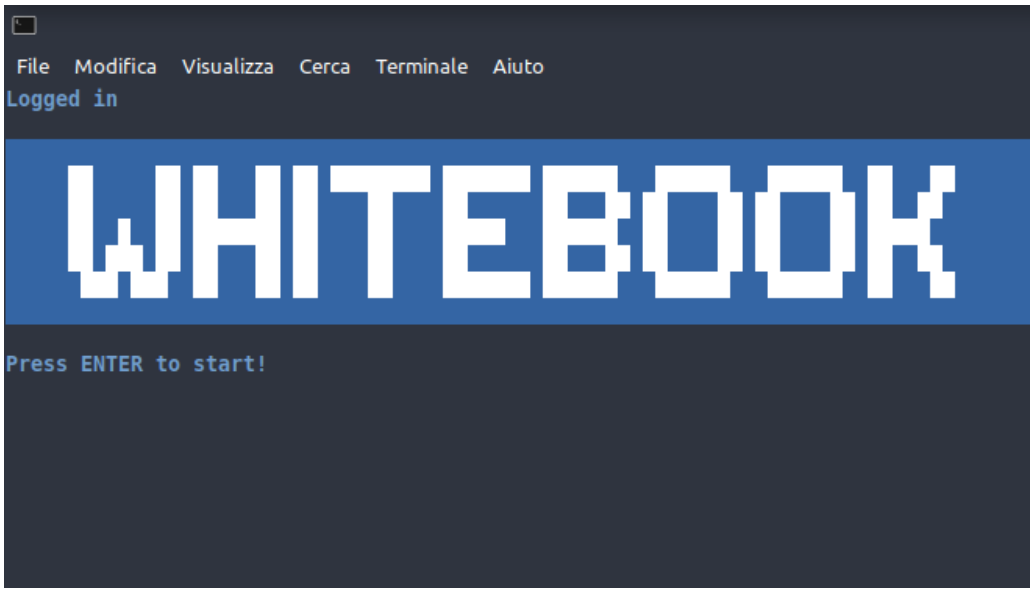
WhiteBook - Documentation

Elia Boninsegna
31-05-2020
Computer System And Programming Project

Introduction



"`WhiteBook`" is an application developed to be used with linux terminal. The main goal of the application is to create an interactive forum where each user can post messages and reply to messages sent from other users.
Messages in the application are divided into topics. Each user can decide to subscribe to a topic, moreover if the user is subscribed it can post and visualize messages in the topic.
A user can also see the status of a sent message.
This means that the application keeps track of the views of a message from other users subscribed to the topic, therefore each user can check which users have viewed a sent message and which have not.
Users in the application are managed by the administrator, so only the admin can register or delete users in the whiteboard.

Design choices

In the file "`struct.h`" is defined the whiteboard structure and all the functions implemented to interact with the whiteboard. The main "`whiteboard`" structure is divided in two arrays, that is "`users`" and "`topics`", and it is defined as follows:

whiteboard	
users	Array of structs "user", contains the list of the users in the whiteboard
topics	Array of structs "topic", contains the list of the topics in the whiteboard

The "user" struct is implemented as follows:

user	
username	Array of 32 chars containing the username of the users
password	Array of 32 chars containing the password of the user
subscriptions	Array of integer containing the list of the topics to which the user is subscribed

The "topic" struct is implemented as follows:

topic	
id	is a 4-character string that corresponds to the index of the topic in the list of topics padded with zeros. *Example: topic = topic_list[9] -> topic.id == "0009"*
name	Array of 32 characters containing the name of the topic
messages	array of "message" structures containing the list of messages in the topic
owner	index of the user who created the topic
subscripteds	array containing the indexes of users subscribed to the topic

Message struct is divided as follows:

message	
id	is an 8-character string, the first 4 characters correspond to the id of the topic in which the message is present, the other 4 characters correspond to the message index in the list of messages on the topic, padded with zeros. *Example:* *message.id=="00010002"* *topic.id == "0001"* *message==topic.messages[2]*

text	String containing the text of the message
in_reply_to	If the message is a reply to another message, it contains the index of the message to which it was replied
replies	Array containing the list of message indexes in response to the current message
visualized	Boolean value which is 0 if someone has not yet visualized the message, 1 if all users visualized the message. The message is considered as visualized when all users subscribed to the topic at the time of posting visualized the message.
visualizations	Array of size equal to the maximum number of users on the whiteboard, each index corresponds to a user. Each element of the array can take the following values: - 0: user visualized the message - 1: user didn't visualize the message - 2: user is not subscribed to the topic in which the message was posted

The application is a client-server application. The file "`server.c`" contains the code of the application's server which uses sockets to communicate with clients and "shared memory" to store whiteboard structure. Each time a client tries to connect, the server forks a new process that will handle the client. The subprocess handling the client can access and write data in the whiteboard structure using semaphores to avoid collisions. Each handling process will loop receiving commands by the client until the client sends the command "`exit`". Receiving this command the process closes the connection and dies.

The client code is defined in "`client.c`" file. The client performs a connection using sockets with the server and simply loops sending commands and receiving replies from the server. The available commands are the following:

- **create topic:**
  Command used to create a new topic. The server will ask the user for the name of the topic that the user wants to create and will automatically subscribe the user to the topic just created.
- **delete topic:**
  Command used to delete an existent topic. This command can be used only if the user is the creator of the topic.
- **list available topics:**
  Command used to show the user the complete list of topics on the whiteboard.
- **list subscripted topics:**
  Command used to show the list of topics where the user is subscripted.
  This command is not shown in the admin menu because the administrator can access automatically to all topics without subscribing.

- **subscribe to topic:**
  Command used to subscribe to a topic. The server will ask the user for the id of the topic and will perform the subscription.
- **create thread:**
  Command used to append a new message to the root of a topic:
  all messages under the topic root are considered as threads.
  The server will ask the user for the id of the topic where the message must be posted and for the text of the message.
  A user can post a message in a topic only if subscribed.
- **reply message:**
  Command used to reply an existent message. The server will ask the user for the id of the message to which he wants to reply and for the text of the new message.
  The user can reply to a message only if subscribed to the topic where the message is posted.
- **list messages:**
  Command used to view the list of the messages in a topic. Replies will be printed using tabs.
  A user can view the list of messages in a topic only if subscribed to the topic.
- **message status:**
  Command used to print the status of a sent message. The server will ask the user for the id of the message and, if the user coincides with the sender of the message it will send the list of the users who visualized the message and the list of the users who hasn't already done it.
- **exit:**
  Command used to exit from whitebook and close the connection with the server.

Only for user "admin":

- **add user:**
  Command used by the administrator to add a new user to the whiteboard structure
- **delete user:**
  Command used by the administrator to remove a user
- **show users:**
  Command used by the administrator to show the list of the users registered in the whiteboard structure

Test cases

In order to test the application, some tests were carried out considering:

- the limit cases applicable to each function
- the memory limits of the structure
- user input validation to avoid buffer overflows

The following lines show the tests performed on the functions in detail:

- **create topic:**
  Topics limit in the whiteboard is 32, if a user tries to create more topic the application will return an error code.
  The name of the topic cannot be null or larger than 32 characters to avoid buffer overflows. If a user tries to insert a larger topic name the application will return an error code.

- **delete topic:**
  Checks on trying to delete a missing topic or trying to delete a topic not created by the user have been performed.
  In these cases the application returns an error code, blocking the operation.

- **subscribe to topic:**
  Checks on trying to subscribe to a missing topic and trying to subscribe to an already subscribed topic have been carried out.
  In these cases the application returns an error code, blocking the operation.

- **create thread:**
  If a user tries to create a thread in a topic where the user is not subscripted or in a missing topic the application will return an error code.
  The length of the message can't be larger than 60 bytes or null. If a user tries to insert a message not considering those parameters the application will return an error code and block the operation.
  The number of messages in a topic can't be larger than 128. If the user tries to append more messages the application will return an error code.

- **reply message:**
  If a user tries to reply to a missing message the application will return an error code.
  The length of the message can't be larger than 60 bytes or null. If a user tries to insert a message not considering those parameters the application will return an error code and block the operation.
  The number of messages in a topic can't be larger than 128. If the user tries to append more messages the application will return an error code.

- **list messages:**
  If a user tries to list messages in a missing topic or in a topic where the user is not subscribed the application will return an error code.

- **message status:**
  If a user tries to see the status of a message not owned by the user or of a missing message, the application will return an error code.

- **add user:**
  In the application structure cannot be registered more than 128 users. If the admin tries to register more users the application will return an error message.
  Username and password cannot be null or larger than 32 bytes. If the admin tries to insert username or password not considering those parameters the application will return an error code and block the operation.
  If a regular user tries to use the functionality, the application will return an error code.

- **delete user:**
  If the administrator tries to delete a missing user the application will return an error message.
  The admin's account cannot be deleted. If the admin tries to delete his account the application will return an error message.
  If a regular user tries to use the functionality, the application will return an error message.

By using a test server (`test-server.c`) several tests have been performed on the saturation of the whiteboard structure. The behaviour of the server seems to be resistant to the saturation of the structure allocated in shared memory. For example reaching the limit of the topics number, the server will not permit to create anymore topics until someone deletes an old topic. The same tests have been carried out over the number of messages per topic and the number of users registered in the whiteboard.

Release notes

The server creates automatically two users:

- admin (password: “admin”) with administrative privileges
- test (password: “test”) with standard privileges

To compile the server you can use the “`make`” command. Two binaries will be generated, “`client`” and “`server`”.

- SERVER USAGE: “`./server`“
- CLIENT USAGE: “`./client`”

Project is stored on GitHub: `https://github.com/Ebloin/whiteboard-forum`