B4 - Network Programming

B-NWP-400

my Teams

Collaborative communication application

0.8

my Teams

binary name: myteams_server, myteams_cli
repository name: NWP_myteams_*$ACADEMICYEAR*
language: C
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The goal of this project is to create a **server** and a **CLI client**.
You MUST create your own protocol and describe it in a RFC's style documentation.
You MUST create your own data model in compliance with the given library technical properties.
You MUST implement requested commands in the **CLI client**.
You MUST use the given server and client libraries to print every events and data.
The network communication will be achieved through the use of TCP sockets.
You MUST push the given logging library and its includes at the root of the repo in a subfolder `libs` like `NWP_myteams_*$ACADEMICYEAR*/libs/myteams/[extracted files]`.

Server

```
Terminal
~/B-NWP-400> ./myteams_server -help
USAGE: ./myteams_server port

      port  is the port number on which the server socket listens.
```

The server MUST be able to handle several clients at the same time by using **select** for command management.
When the server is shut down it MUST save its internal information in the current folder. (Think about Ctrl-c)
When the server starts it must look if the save file exists and load it if it does.

A good use of **select** is expected for both reading and writing on sockets.
Any bad use of **select** would cause point loss

The use of **fork** and **threads** is prohibited

Take a look at **man 3 uuid** and **man 7 queue**.

Think about **strace** to debug your program.

Features

Your server MUST be able to manage a collaborative communication application like the well known Microsoft Teams ®.
A collaborative communication application is a service able to manage several communication teams, where discussion are organised like following:

- threads (initial post and additional comments) in a specific channel
- discussion (personnal messages)

Here are the features intended to be implemented :

- Creating/Joining/Leaving a team
- Creating a user
- Creating a channel in a team
- Creating a thread in a channel
- Creating a comment in a thread
- Saving & restoring users, teams, channels, threads & associated comments
- Personnal discussion (from a user to an other)
- Saving & restoring personnal discussion

LOG

The server MUST use the given logging library to print EVERY requested events on the standard error output as described :
Please refer to the library's header file to see the list of events to handle.

DO NOT WRITE ANYTHING ON THE ERROR OUTPUT!
The given library will handle it.

If you need to display error information use for this project the standard output.

SECURITY

There is no password authentication required for this subject but you should always develop with security in mind.
A user that is not logged in should **NOT** be able to see connected users for example.
Someone that is **NOT** subscribed to a team should not be able to create a thread.
Someone that is **NOT** subscribed in a team should not receive events related to that team (new threads etc…).

Command Line Interface (CLI) Client

```
Terminal
~/B-NWP-400> ./myteams_cli -help
USAGE: ./myteams_cli ip port
       ip    is the server ip address on which the server socket listens
       port  is the port number on which the server socket listens
```

Features

Your client will handle the following commands from the standard input :

- /help : show help
- /login ["user_name"] : set the user_name used by client
- /logout : disconnect the client from the server
- /users : get the list of all users that exist on the domain
- /user ["user_uuid"] : get details about the requested user
- /send ["user_uuid"] ["message_body"] : send a message to specific user
- /messages ["user_uuid"] : list all messages exchanged with the specified user
- /subscribe ["team_uuid"] : subscribe to the events of a team and its sub directories (enable reception of all events from a team)
- /subscribed ?["team_uuid"] : list all subscribed teams or list all users subscribed to a team
- /unsubscribe ["team_uuid"] : unsubscribe from a team
- /use ?["team_uuid"] ?["channel_uuid"] ?["thread_uuid"] : Sets the command context to a team/channel/thread
- /create : based on the context, create the sub resource (see below)
- /list : based on the context, list all the sub resources (see below)
- /info : based on the context, display details of the current resource (see below)

/create

When the context is not defined (/use):

- /create ["team_name"] ["team_description"] : create a new team

When team_uuid is defined (/use "team_uuid"):

- /create ["channel_name"] ["channel_description"] : create a new channel

When team_uuid and channel_uuid are defined (/use "team_uuid" "channel_uuid"):

- /create ["thread_title"] ["thread_message"] : create a new thread

When team_uuid, channel_uuid and thread_uuid are defined (/use "team_uuid" "channel_uuid" "thread_uuid"):

- /create ["comment_body"] : create a new reply

/LIST

When the context is not defined (/use):

- /list : list all existing teams

When team_uuid is defined (/use “team_uuid”):

- /list : list all existing channels

When team_uuid and channel_uuid are defined (/use “team_uuid” “channel_uuid”):

- /list : list all existing threads

When team_uuid, channel_uuid and thread_uuid are defined (/use “team_uuid” “channel_uuid” “thread_uuid”):

- /list : list all existing replies

/INFO

When the context is not defined (/use):

- /info : display currently logged-in user details

When team_uuid is defined (/use “team_uuid”):

- /info : display currently selected team details

When team_uuid and channel_uuid are defined (/use “team_uuid” “channel_uuid”):

- /info : display currently selected channel details

When team_uuid, channel_uuid and thread_uuid are defined (/use “team_uuid” “channel_uuid” “thread_uuid”):

- /info : display currently selected thread details

General Informations

Please note that all arguments of the existing commands should be quoted with double quotes.
A missing quote should be interpreted as an error.
Please note that all the names, descriptions and message bodies have a pre-defined length which will be as follow:

- MAX_NAME_LENGTH 32
- MAX_DESCRIPTION_LENGTH 255
- MAX_BODY_LENGTH 512

You don't have to spend too much time on the information display.
You MUST therefore use the functions provided by the logging library (and only these) to display the information given by the server.
Please look at the display library header, to find the variables of each data structure.