TÉCNICO LISBOA

Redes de Computadores

1st Semester 2019/2020
LEIC Alameda

Programming using the *Sockets* interface

"RC Forum"

1. Introduction

The goal of this project is to develop a simple online forum to share questions about the study of Redes de Computadores. Users can be either students or instructors. Users can suggest topics, post questions composed of text and possibly also an image to illustrate the question, and post answers. The forum is moderated to approve new users.

The development of this project requires implementing: (i) a *Forum Server* (FS); (ii) the *User Application* (user). The forum server and the various user application instances are intended to operate on different machines connected to the Internet.

The operation is as follows.
The FS will be running in a machine with known IP address and port.
Any user must first register with the FS, using as identity the IST student number (a 5-digit number), and have the registration approved (this can be done by a moderator, or simply by checking a previously available list of enrolled students). Questions, and the corresponding answers, are stored by the FS in a separate directory for each topic. For each question there may be multiple answers.

A user can perform the following operations:
1 – Register as a new user.
2 – Request the list of topics available in the FS.
3 –Request, for a given topic, the list of available question titles and the indication if they have been replied.
4 – Retrieve the text and images of a previous question and the corresponding replies available.
5 – Submit an answer to an existing question.
6 – Propose a new topic.
7 – Submit a new question for an existing topic.

If a question has received many replies, the corresponding retrieval will recover up to 10 answers (the most recent).

Topics are identified by a string of no more than 10 alphanumeric characters without spaces. Question titles should have at most 10 alphanumeric characters without spaces. Answers are identified by the question title concatenated with an underscore and a 2-digit number, assigned sequentially.

The questions received for each topic are stored by the FS in the corresponding folder, using a file named with the question title and extension ".txt". If the question is accompanied by an image, the image is also stored with the same name and the original file extension (different from ".txt"). When the FS server starts, the questions and answers already available in the respective directories will be available for consultation by the users.

The project tests will include the FS server and several instances of the user application.
For the implementation, the application layer protocols operate according to the client-server paradigm, using the transport layer services made available by the socket interface, using the TCP and UDP protocols.
The management of the users' registration and all consultation of previously uploaded topics and respective questions titles will be supported on communications using the UDP protocol. The communications between a user and the FS for the submission of new questions or new answers, as well as for the retrieval of previously uploaded questions and the corresponding answers, will use the TCP protocol.

2. Project Specification

2.1 User Application

The program implementing the user application should be invoked using the command:

`./user [-n FSIP] [-p FSport]`,

where:

FSIP this is the IP address of the machine where the forum server (FS) runs. This is an optional argument. If this argument is omitted, the FS should be running on the same machine.

FSport this is the well-known port where the FS server accepts user requests, either in UDP or in TCP. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the group number.

Once the user program is running, it waits for the user to indicate the action to take, notably:

- *register userID / reg userID* – following the *register* instruction the user application sends a registration message to the FS server, using the UDP protocol, sending the user's identification *userID* (the 5-digit IST student number), for validation by the FS. Alternatively, the short form *reg* can be used. The *userID* is stored in memory for usage during this user's session. The result of the FS validation should be displayed to the user.

- *topic_list / tl* – following the *topic_list* instruction the user application should contact the FS server, using the UDP protocol, asking for the list of topics available in the server. Alternatively, the short form *tl* can be used. The received reply should be displayed to the user as a numbered list.

- *topic_select topic / ts topic_number* – following the *topic_select* instruction the user application stores the selected topic *topic* as the (locally) active topic. In alternative, the short form *ts* can be used with the topic being indicated by its *topic_number* as displayed in the list of available topics. No communication is involved for this instruction.

- *topic_propose topic / tp topic* – following this instruction the user application contacts the FS server, using the UDP protocol, suggesting a new topic *topic* for questions. The short form *tp* can also be used. The reply from the FS confirming or not the acceptance of the new topic should be displayed to the user.

- *question_list / ql* – following this instruction the user application should contact the FS server, using the UDP protocol, asking for the list of questions available for the previously (locally) selected topic (*topic*). The short form *ql* can also be used. The received list of question titles should be displayed to the user as a numbered list, together with the number of available replies for each question.

- *question_get question / qg question_number* – following this instruction the user application should establish a TCP connection with the FS server, to ask for the file(s) (text or text + image) related to question with title *question*, of the (locally) active topic *topic*, as well as the corresponding answer files available (a set of text or text + image files). The short form *qg* can also be used, in which case the question is indicated by the number *question_number* previously used to display the list of available question titles. The received files should be stored in a directory with the topic name. Additionally, the question title *question* is (locally) selected as the currently active question.

- *question_submit question text_file [image_file.ext]/ qs question text_file [image_file.ext]* – following this instruction the user application should establish a TCP connection with the FS server submitting a new question related to the (locally) active topic *topic*, including: the question title *question*, the file(s) (*text_file* or *text_file* + *image_file.ext*, where *ext* is the extension of the image file – the text.file extension is assumed to be *txt*). Alternatively, the instruction can use the short form *qs*. The reply from the FS confirming or not the acceptance of the question should be displayed to the user.

- *answer_submit text_file [image_file] / as text_file [image_file]* – following this instruction the user application should establish a TCP connection with the FS server, submitting the file(s) (*text_file* or *text_file* + *image_file*) providing an answer to the (locally) active question with title *question*, of topic *topic*. Alternatively, the instruction can use the short form *as*. The reply from the FS confirming or not the acceptance of the answer should be displayed to the user.

- *exit* – the user application terminates.

2.2 *Forum Server* (FS)

The program implementing the *Forum Server* should be invoked using the command:

`./FS [-p FSport]`,

where:

FSport is the well-known port where the FS server accepts requests, both in UDP and TCP. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the number of the group.

The forum server (FS) makes available two server applications, one in UDP and the other in TCP, with well-known ports *FSport*, to answer user requests.
The FS server outputs to the screen a short description of the received requests and the IP and port originating those requests.
Each received request should start being processed once it is received.

3. Communication Protocols Specification

3.1 *User–FS* Protocol (in UDP)

The part of the interaction between the user application and the FCS server relying on the UDP protocol uses the following request and reply protocol messages:

a) `REG` *`userID`*
Following the *`register/reg`* instruction, the user application sends the user ID *`userID`* (the 5-digit IST student number) for validation by the FS server.

b) `RGR` *`status`*
In reply to a `REG` request the FS server replies in UDP indicating the status of the registration request. If the `REG` request was successful (valid user ID) the *`status`* is "`OK`"; if the user ID is not accepted the *`status`* is "`NOK`".

c) `LTP`
Following the *`topic_list/tl`* instruction, the user application requests the FS server to return the list of topics available.

d) `LTR` *`N`* `(`*`topic:userID`* `)*`
In reply to a `LTP` request the FS server replies with the list of the *`N`* topics available, followed by a set of *`N`* strings with each of the topic descriptions and the ID of the user who proposed that topic (*`topic:userID`*). The *`N`* strings are separated by single spaces. The protocol message is terminated with a newline character `\n`. If no topics are yet available in the FS the reply will be `LTR` *`0`*.

e) `PTP` *`userID topic`*
Following the *`topic_propose/tp`* instruction, the user application identifies the user with its ID *`userID`* and requests the FS server to create the new topic *`topic`*.

f) `PTR` *`status`*
The FS reply to a `PTP` request is the new topic creation confirmation. If the `PTP` request was successful the *`status`* will be "`OK`", if the topic already existed the *`status`* is "`DUP`", if the topic list is full the *`status`* is "`FUL`", otherwise the *`status`* will be "`NOK`".

g) `LQU` *`topic`*
Following the *`question_list/ql`* instruction, the user application requests the FS server to return the list of question titles available for the selected topic *`topic`*.

h) `LQR` *`N`* `(`*`question:userID:NA`* `)*`
In reply to a `LQU` request the FS server replies with the number *`N`* of question titles available for the selected topic, followed by a set of *`N`* strings with each of the question titles (*`question`*). The ID of the user that proposed the question (*`userID`*) and the number of available answers for that question (*`NA`*). The *`N`* strings are separated by single spaces. The protocol message is terminated with a newline character `\n`. If no question titles are yet available for this topic the reply will be `LQR` *`0`*.

If an unexpected protocol message is received, the reply will be "`ERR`".

In the above messages the separation between any two items consists of a single space. Each request or reply message ends with the character "\n".

3.2 *User–FS* Protocol (in TCP)

The part of the interaction between the user application and the FS server relying on the TCP protocol uses the following request and reply protocol messages:

a) GQU *topic question*
Following the *question_get/qg* instruction, the user application opens a TCP connection with the FS server and sends a request asking for the file(s) (text or text + image) related to question with title *question*, for the selected topic *topic*, as well as the corresponding answer files available (a set of text or text + image files).

b) QGR *qUserID qsize qdata qIMG [qiext qisize qidata] N (AN aUserID asize adata aIMG [aiext aisize aidata])**
The FS reply to a GQU request contains:

- *qUserID* is the ID of the user who submitted the question
- the size *qsize*, in bytes, of the question text file
- the question text file data *qdata*
- a binary flag *qIMG* indicating if an image is available for the question
- if *qIMG* = 1:
    - the 3 byte extension *qiext* of the question image file
    - the size *qisize*, in bytes, of the question image file
    - the image data *qidata*
- the number *N* of available answers (up to the 10 most recent ones)

For each of the available answers:

- the answer number *AN* (a 2 digit number)
- *aUserID* is the ID of the user who submitted the answer
- the size *asize*, in bytes, of the answer text file
- the answer text file data *adata*
- a binary flag *aIMG* indicating if an image is available for this answer
- if *aIMG* = 1:
    - the 3 byte extension *aiext* of the answer image file
    - the size *aisize*, in bytes, of the answer image file
    - the answer data *aidata*

The received files should be stored in a directory with the topic name. The question is stored in a file named with the question title and extension ".txt". If the question is accompanied by an image, the image is also stored with the same name and the original file extension *ext*. The answers are stored in a similar way, with the name being composed of the question title concatenated with an underscore and the 2-digit number *AN*.
If the GQU request cannot be answered (e.g., no such query or topic are available) the reply will be "QGR EOF". If the GQU request is not correctly formulated the reply is "QGR ERR".
After receiving the complete reply message, the user application closes the TCP connection with the FS.

c) QUS q*UserID* topic question qsize qdata qIMG [iext isize idata]
Following the *question_submit/qs* instruction, the user application opens a TCP connection with the FS server, identifies the user with its ID *qUserID*, and sends a question with title question, for the topic *topic*.
The QUS request additionally contains:

- the size *qsize*, in bytes, of the question text file
- the question text file data *qdata*
- a binary flag q*IMG* indicating if an image is being sent with the question
- if q*IMG* = 1:
    - the 3 byte extension *iext* of the question image file
    - the size *isize*, in bytes, of the question image file
    - the question data *idata*

d) QUR *status*
In reply to a QUS request, the FS server replies in TCP indicating the status of the file transfer. If the QUS request was successful the *status* will be "OK", if the question already existed the *status* is "DUP", if the question list is full the *status* is "FUL", otherwise the *status* will be "NOK".
After receiving the complete reply message, the user application closes the TCP connection with the FS.

e) ANS *aUserID* topic question asize adata aIMG [iext isize idata]
Following the *answer_submit/as* instruction, the user application opens a TCP connection with the FS server, identifies the user with its ID *aUserID*, and sends an answer to the question with title question, of topic topic.
The ANS request additionally contains:

- the size *asize*, in bytes, of the answer text file
- the answer text file data *adata*
- a binary flag *aIMG* indicating if an image is being sent with the answer
- if *aIMG* = 1:
    - the 3 byte extension *iext* of the answer image file
    - the size *isize*, in bytes, of the answer image file
    - the answer data *idata*

f) ANR *status*
In reply to a ANS request, the FS server replies in TCP indicating the status of the file transfer. If the ANS request was successful the *status* will be "OK", if the answer list is full the *status* is "FUL", otherwise the *status* will be "NOK".
After receiving the complete reply message, the user application closes the TCP connection with the FS.

If an unexpected protocol message is received, the reply will be "ERR".
In the above messages the separation between any two items consists of a single space.
Each request or reply message ends with the character "\n".

4. Development

4.1 Development and test environment

Make sure your code compiles and executes correctly in the development environment available in lab LT5.

4.2 Programming

The operation of your program, developed in the *C programming language*, should be based on the following set of system calls:

- Computer name: `gethostname()`.
- Remote computer IP address from its name: `getaddrinfo()`.
- UDP server management: `socket()`, `bind()`, `close()`.
- UDP client management: `socket()`, `close()`.
- UDP communication: `sendto()`, `recvfrom()`.
- TCP server management: `socket()`, `bind()`, `listen()`, `accept()`, `fork()`, `close()`.
- TCP client management: `socket()`, `connect()`, `close()`.
- TCP communication: `write()`, `read()`.
- Multiple inputs multiplexing: `select()`.

4.3 Implementation notes

Developed code should be adequately structured and commented.
The `read()` and `write()` system calls may read and write, respectively, a smaller number of bytes than solicited – you need to ensure that your implementation still works correctly.
Both the client and server processes should terminate gracefully at least in the following failure situations:

- wrong protocol messages received from the corresponding peer entity;
- error conditions from the system calls.

5 Bibliography

- W. Richard Stevens, Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2nd edition, Prentice-Hall PTR, 1998, ISBN 0-13-490012-X, chap. 5.
- D. E. Comer, Computer Networks and Internets, 2nd edition, Prentice Hall, Inc, 1999, ISBN 0-13-084222-2, chap. 24.
- Michael J. Donahoo, Kenneth L. Calvert, TCP/IP Sockets in C: Practical Guide for Programmers, Morgan Kaufmann, ISBN 1558608265, 2000
- On-line manual, `man` command
- Code Complete - http://www.cc2e.com/
- http://developerweb.net/viewforum.php?id=70

6 Project Submission

6.1 Code

The project submission should include the source code of the programs implementing the *user* and the *FS server*, as well as the corresponding *Makefile*.
The makefile should compile the code and place the executables in the current directory.

6.2 Auxiliary Files

Together with the project submission you should also include any auxiliary files needed for the project operation together with a *readme.txt* file.

6.3 Submission

The project submission is done by e-mail to the lab teacher, **no later than October 18, 2019, at 23:59 PM**.
You should create a single `zip` archive containing all the source code, makefile and all auxiliary files required for executing the project. The archive should be prepared to be opened to the current directory and compiled with the command `make`.
The name of the archive should follow the format: **`proj_"group number".zip`**

7 Questions

You are encouraged to ask your questions to the teachers in the scheduled foreseen for that effect.

8 Open Issues

You are encouraged to think about how to extend this protocol in order to make it more generic. For instance, how could an interactive chat functionality be added?