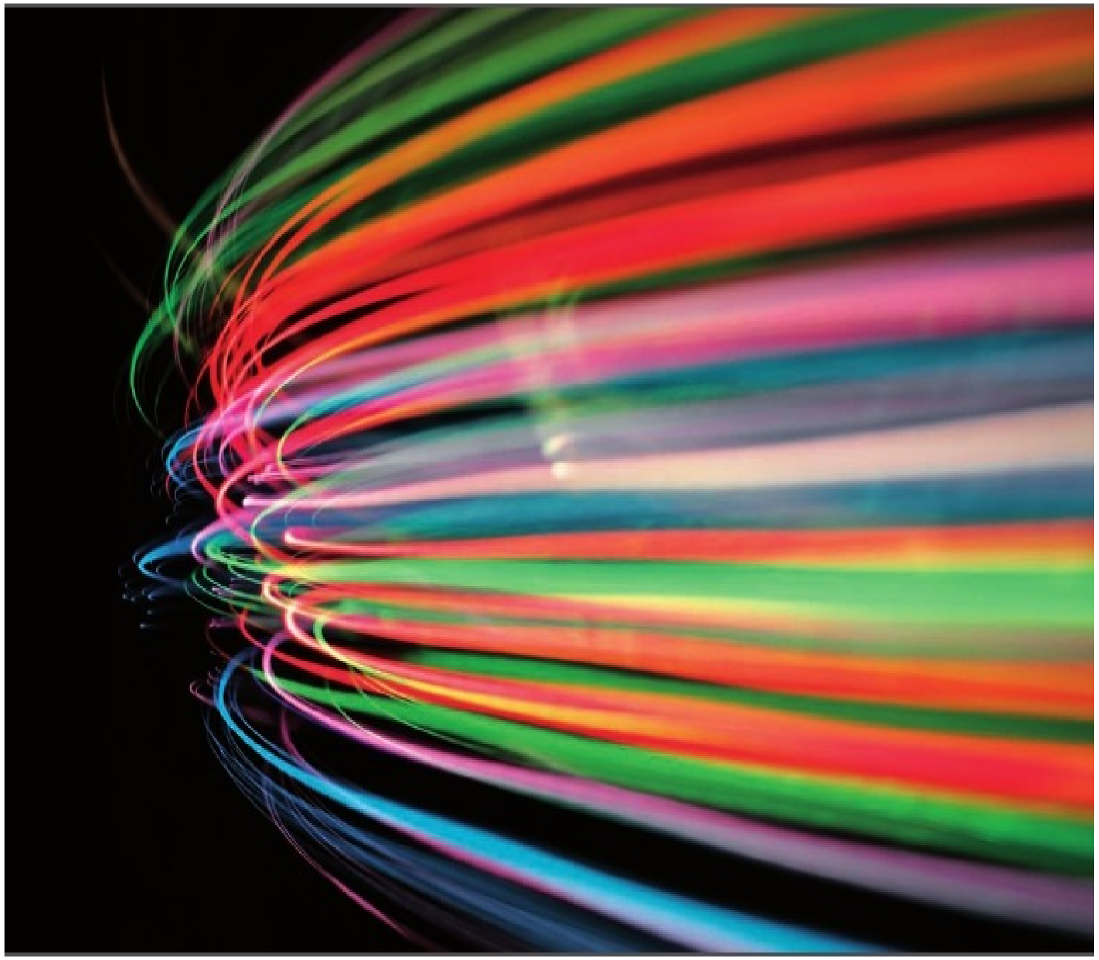
Université Léonard da Vinci



PROJECT

Network Programming

2020

Université Léonard da Vinci

Project network PROGRAMMING in Python or C

2020

General Project

This work is to be done in groups of 4 students (same group of tutorials). It must be returned no later than december 18, 2020 at 6 p.m.

- There are 3 projects to choose from and, in each group of tutorials, you cannot exceed 5 groups of 4 students (maximum) for the same project.
- Any postponement after this time slot will be penalized with a penalty of -5 points per day of delay.
- You will upload a .zip file to De Vinci Online, as a team, containing:
  - the source code in C (.c and .h files) or in Python
  - a tool installation document
  - the mini-report in PDF of 7 pages **minimum**

Note :

A rendering without the report and / or installation document will be noted 00/20.

A program with compilation warnings will be scored with a penalty of up to -5 points depending on the nature of the warnings.

A program that neither compiles nor executes will be rated 00/20.

Any cheating gives a score of 00/20 for the whole module.

General Project

Mini report

A mini-report of at least 7 pages is to be submitted with the code.

Must be included:
- the name and surname of the students who carried out the work;
- an explanation in French or English of the coding of each proposed function;
- a small diagram which completes the explanation.

Server project (DNS / DHCP trace)

- server under (Windows / Linux):

1) Connection to your DNS / DHCP log server (TCP)
2) Possibility to log in several @MACs managed by SQLite
3) Retrieval of DNS and DHCP requests in an SQLite database by @MAC
4) Possibility to display the logs by @MAC or by the OUI file (Manufacturer)
5) Possibility of sorting logs by (@IP, Date, Time ...)
6) Managing a list of unauthorized DNS
7) Notification if a DNS is in the list of unauthorized DNS
8) Notification if a DHCP @IP is in the list of unauthorized @MACs
9) Save logs to one file per day
10) Your original feature

Bonus:

1) Ability to read and send an alert email if for example DDOS attempt

Server Project (Mail)

- server under (Windows / Linux):

1) Connect your mail server (Google ...)
2) Possibility of having several email accounts in different servers managed by SQLite
3) Retrieving emails from an SQLite database
4) Ability to read emails
5) Possibility to send emails
6) Possibility of sorting emails by (DE, To, Subject ...)
7) Log log of connections to the mail server (login, @IP, date, etc.)
8) Saving emails to a file
9) Your original feature

Bonus:

1) Ability to read and send emails with attachments
2) Secure transmission by an end-to-end encryption algorithm

Server Project (Chat)

- server under (Windows / Linux):

1) Connection of Clients to Server in UDP or TCP
2) Ability to send and receive messages (chat) (280 characters per message)
3) Transfer files (jpeg or others) (client to client, client to server)
4) Multi-threaded more clients (50) per server
5) Command line on the server (for example #Kill Tim) to disconnect the client Tim
6) The server requires a username and password to connect. (use SQLite for password management) the database must be on the server
7) Log log (login, @IP, date, etc.) on the server
8) Your original feature

Bonus:

1) The server runs on Linux and Windows
2) Secure transmission by an end-to-end encryption algorithm

Server Project

- **Server function (command line):**

1) #Help (list command)
2) #Exit (server shutdown)
3) #Kill <user>
4) #ListU (list of users in a server)
5) #ListF (list of files in a server)
6) # Private <user> (private chat with another user)
7) #Alert <all users>

- **Client function (command line):**

1) #Help (list command)
2) #Exit (client exit)
3) #ListU (list of users in a server)
4) #ListF (list of files in a server)
5) #TrfU (Upload file transfer to a server)
6) #TrfD (transfer Download file to a server)

- # Private <user> (private chat with another user)
- #Public (back to the public)

1) #Ring <user> (notification if the user is logged in)
2) Your original orders

TIPS 1

- Sometimes an abnormal exit from a program
(for example, ctrl-c) does not properly release a port

- Finally (after a few minutes) the port may be released

- You can kill the process, or to reduce the likelihood of this problem including the following C code:
  - In the header:

```
#include <signal.h>
void cleanExit () {exit (0);}
```

  - In the socket code:

```
signal (SIGTERM, cleanExit);
signal (SIGINT, cleanExit);
```

TIPS 2

- Check out Beej's guide to network programming using Socket Internet

- Find the specification of the function to use for more information, or see the man pages.

TIPS 3

- How do I find the IP address of the machine my server program is running on?
  - Use 127.0.0.1 or localhost to test and access a server running on your local machine
  - To verify that the port is functional, use the netstat command
  - For a remote server running Linux, use the command: "ifconfig", for Windows: "ipconfig" in the terminal

REFERENCES

Below are the references for a more in-depth study of Socket programming with C:

- Beej's Guide to Network Programming Using Internet Sockets
  - 2019S1_C5_DOC_BGNET_Socket Programming_EN.pdf
- 2019S1_C5_DOC_TCP_IP_Sockets_in_C: _Practical_Guide_for_Programmers_EN.pdf
- The GNU C Reference Manual