

Programming Assignment

Start Assignment

Due Nov 10 by 11:59pm **Points** 6 **Submitting** a file upload
File Types py, c, cc, cpp, txt, java, and zip **Available** Oct 20 at 12pm - Nov 12 at 11:59pm

- [Source code of the server in python](https://canvas.cityu.edu.hk/users/166222/files/11268098?verifier=ySHr2dYG8NoqD0Un7XJmq3H5gjzoODMU8KZECIxA&wrap=1)
(<https://canvas.cityu.edu.hk/users/166222/files/11268098?verifier=ySHr2dYG8NoqD0Un7XJmq3H5gjzoODMU8KZECIxA&wrap=1>)_ 
(https://canvas.cityu.edu.hk/users/166222/files/11268098/download?verifier=ySHr2dYG8NoqD0Un7XJmq3H5gjzoODMU8KZECIxA&download_frd=1)

Table of Contents

1. Project Description
2. Development Environment
3. Project Demo
4. List of Tasks
5. Project Requirements (Code Quality)
6. Assessment Tool
7. Submission
8. Getting Help

1. Project Description

In this project, you are required to implement a client's bulletin board system for a client-server application, where clients can communicate with the server through issuing different commands. The server runs on a specified port and supports 4 commands issued by the client: POST_STRING, POST_FILE, GET, and EXIT. The functionality of each of the commands is given as follows.

- **POST_STRING:** this command allows a client to send a text file to the server line by line. The client must type the text (using the standard input) line-by-line. The first line is the command itself (i.e. POST_STRING) and subsequent lines are treated as the text. The end of the text input is signalled by a special symbol &.

Example:

client: POST_STRING

client: Welcome socket programming in cs3201

client: I start writing an app code ABC

client: more text.

client: &

server: OK

Then, the message sent to the server is a string: POST_STRING\nWelcome socket programming in cs3201\nI start writing an app code ABC\nmore text.\n&\n

The server will acknowledge the receipt of the message with "OK"

- **POST_FILE:** this command allows a client to send a text file with length no more than 256 bytes from his/her local machine to the server, and the server responds to the command by asking the client the absolute path of the file in client machine. **You can edit the save path in server code.**

Example: Suppose a named text file is cs3201_project.py whose absolute path in the local machine is /user/file/cs3201_project.py

client: POST_FILE

server: please send the path name of your text file

client: /user/file/cs3201_project.py

If the file transfer is successful, the server replies with "OK"; otherwise, the operation fails, and the client needs to resend it again. For example, if the path does not exist or path typing errors, the server will not respond to the request.

- **GET:** upon receiving this command, the server will send **all** previously posted messages and files by the client and other clients back to the client as the standard output.
- **EXIT:** this informs the server to close the socket to the client. The server will acknowledge the receipt of the command with "OK".

Assume that the server is running on IP address 132.196.0.1 with port No. 16011. The user starts the client by running the command ./BulletinBoardClient 132.196.0.1 16011 in the terminal (Linux/macOS), or by double-clicking on the compiled executable BulletinBoardClient.exe (Windows).

After establishing the TCP connection to the socket of the server, the client program will repeatedly wait for a command from the user to execute until it receives command EXIT, and stops the client

session. For commands GET and EXIT, the client will just send the command to the server, whereas for POST_STRING, the client will send a message line by line, where the first line is the command (i.e. POST_STRING), and the following lines include the text, and ending with a special symbol &. Below is a possible communication sequence between the client and the server:

```
===== Initialize socket =====
input IP address : wrongip
input port number: wrongport
Error: connection is not built, try again
===== Initialize socket =====
input IP address : localhost
input port number: 16011
===== Input command =====
Input command:hello
server: ERROR - Command not understood
Input command:POST_STRING
===== Content (Type a lone '&' to end message) =====
client: Welcome socket programming
client: Text#!
client: more text.
client: &
server: OK
---
Sent 4 messages to (IP address:127.0.0.1, port number:16011)
Connect status: OK
Send status: OK
---
=====next command=====
Input command:GET
---Received Messages---
client: Welcome socket programming
client: Text#!
client: more text.
client: &
(IP address:127.0.0.1, port number:16011)
Connect status: OK
Send status: OK
=====next command=====
Input command:POST_FILE
server: please send your file and its path name
client:wrong path
Cannot find the file
server: The file not exist
=====next command=====
Input command:POST_FILE
server: please send your file and its path name
client:C:\Users\Administrator\Desktop\aniya.txt
Transfer file absolute path : C:\Users\Administrator\Desktop\aniya.txt
server: OK
=====next command=====
Input command:EXIT
server: OK
```

The IP address localhost used above means "this computer", and the image above show the result when server and client are executed on the same computer.

The **server program** can be found from the project demo file. You **only need to implement the client program**. Name your client **BulletinBoardClient**.

Your code for POST_STRING, POST_FILE/GET commands should print out some useful information, e.g., the information with POST_STRING/POST_FILE can be formatted as follows.

IP Address: xxx.xxx.xxx.xxx Port Number: yyyyy

Connect status: OK/ERROR

Send status: OK/ERROR

where the number xxx.xxx.xxx.xxx and yyyyy are the IP Address and Port Number that are given as input parameters.

Note that the server program sends the text line-by-line continuously until the last line of the text consists of only a single # character. Therefore, a loop is needed in your program to receive the text file as shown in the following pseudo code fragment.

```
while (the current line <> '#'){  
    /* code for receiving the text from the server */  
  
    ...  
  
    /* code for saving the text in the file */  
  
    ...  
}
```

2. Development Environment

Hardware:

Two connected computers: one serves as the client and another serves as the server.

- If you use two laptops:
 - if you are in CityU, you can connect the two laptops to the campus wireless network so that they can communicate with each other ([How to do this? Links to an external site.](http://www.cityu.edu.hk/csc/deptweb/facilities/ctnet/wlan/wlanmain.htm) (<http://www.cityu.edu.hk/csc/deptweb/facilities/ctnet/wlan/wlanmain.htm>)).
 - if the campus wireless network is not available, you can set up a Wi-Fi ad-hoc network ([How to do this?\(Links to an external site.\)](http://compnetworking.about.com/od/wireless/ht/setupadhocwifi.htm) (<http://compnetworking.about.com/od/wireless/ht/setupadhocwifi.htm>)).
- If you use two desktops, you can directly connect the computers using cables and configure them in a subnet ([How to do this?\(Links to an external site.\)](https://helpdeskgeek.com/networking/connect-two-computers-using-a-crossover-cable/) (<https://helpdeskgeek.com/networking/connect-two-computers-using-a-crossover-cable/>)).

If you cannot access to two computers, it is suggested to install a virtual machine to test your program ([How to do this? \(https://canvas.cityu.edu.hk/courses/48996/files/9491620?wrap=1\)](https://canvas.cityu.edu.hk/courses/48996/files/9491620?wrap=1)).

(https://canvas.cityu.edu.hk/courses/48996/files/9491620/download?download_frd=1).

(<https://canvas.cityu.edu.hk/courses/36744/files/6529894/download?wrap=1>).

Software:



You can choose to code the client in either C/C++, Java, or Python, based on your own preference and familiarity.

- If you use **MS Windows**:
 - for C/C++: Visual Studio: <https://visualstudio.microsoft.com/downloads/> (Links to an external site.)
 - Install C++ support in Visual Studio: <https://docs.microsoft.com/en-us/cpp/build/vscpp-step-0-installation?view=vs-2017> (Links to an external site.)
- for Python: PyCharm: <https://www.jetbrains.com/pycharm/download/#section=windows> (Links to an external site.)
- If you use **Linux Ubuntu System** or **Mac OS X**:
 - for C/C++: The gcc/g++ compiler is usually installed by default.
- for Python:
 - PyCharm: (Mac OS) <https://www.jetbrains.com/pycharm/download/#section=mac> (Links to an external site.)
 - PyCharm: (Linux) <https://www.jetbrains.com/pycharm/download/#section=linux> (Links to an external site.)

*Remark: it is suggested that computers involved in the same working group should use the **same operating system** and apply the **same software tool** for the project design.*

3. Project Demo

Apart from the server program, we have also provided the following demo programs to demonstrate the application's operations:

- For Windows System, the demo program is provided as: [Demo program](https://canvas.cityu.edu.hk/users/166222/files/11268152?verifier=PqeBHK08BolpHLVC76LqBhmwojCQmeVZuJXpQaM&wrap=1)
(<https://canvas.cityu.edu.hk/users/166222/files/11268152?verifier=PqeBHK08BolpHLVC76LqBhmwojCQmeVZuJXpQaM&wrap=1>). 
(https://canvas.cityu.edu.hk/users/166222/files/11268152/download?verifier=PqeBHK08BolpHLVC76LqBhmwojCQmeVZuJXpQaM&download_frd=1)
- For Linux System, the demo program is provided as: [Demo program](https://canvas.cityu.edu.hk/courses/48996/files/9491616/download)
(<https://canvas.cityu.edu.hk/courses/48996/files/9491616/download>)
- For Mac OS, the demo program is provided as: [Demo program](https://canvas.cityu.edu.hk/users/166222/files/11281522?verifier=Hoa5Res40fBbSkLOgyvAJxKUR80aiJcYRJ2aFf26&wrap=1)
(<https://canvas.cityu.edu.hk/users/166222/files/11281522?verifier=Hoa5Res40fBbSkLOgyvAJxKUR80aiJcYRJ2aFf26&wrap=1>). 
(https://canvas.cityu.edu.hk/users/166222/files/11281522/download?verifier=Hoa5Res40fBbSkLOgyvAJxKUR80aiJcYRJ2aFf26&download_frd=1)
 - The Demo for Mac os uses port number 1200

Some notes about the provided **server program**:

- The default port number of the server for this demo is **16011**. **You cannot change the port number.**
- It will print out information about errors or commands that it receives from the clients; otherwise, it remains silent.
- When the server receives the GET command, it will print out all stored messages on its side.
- The server keeps listening even if the client disconnects from the server.

4. List of Tasks

Task 1: Socket initialization

Description: initialize and create the TCP socket.

Task 2: Initiate the connection request

Description:

- Input the server IP and port.
- Initiate a connection on a socket.

Task 3: Send command to the server

Description: send the command to the server.

- POST_STRING: the program continues to accept the user input until entering a line that consists only of a "&" followed by a newline character '\n'. Then, send this command and these messages to the server.
- POST_FILE: send this command to the server, and the server responds to ask for the absolute address of the local file.
- GET: send this command to the server.
- EXIT: send this command to the server.
- Other cases: send this command to the server.

Task 4: Receive the message from the server.

Description: receive the message from the server and perform the corresponding action.

- POST_STRING: if "OK" is received, it indicates that the server has successfully received all of the messages from clients. Print out "OK" and then allow the user to input the next command.
- POST_FILE: the server asks for the absolute path name of the text file. If the file transfer is successful, it acknowledges with "OK", otherwise, there is no response from the server.

- GET: receive all messages from server and print them to the screen. Then, allow the user to input the next command.
- EXIT: if "OK" is received, it indicates that the server has successfully received this command. Then, the socket will be closed (see Task 5).
- Other cases: receive the error message "ERROR - Command not understood" that is sent by the server. Print out the error message and allow the user to input another command.

Task 5: Close socket

Description: when the client sent the "EXIT" command and received "OK" from the server in response, the connection socket is closed.

The input must include the IP Address and Port Number of the server.

How to find your IP address:

- Linux or Mac: Type ifconfig in terminal
- Windows: Network & Internet settings -> network properties

5. Project Requirements (Code Quality)

We cannot specify completely the coding style that we would like to see but your submission should include the following:

- Proper decomposition of a program into subroutines (and multiple source code files when necessary) -- A 500 line program as a single routine would not suffice.
- Comment -- judiciously, but not profusely. Comments also serve to help a marker in addition to yourself. To further elaborate:
 - Your favorite quote from Star Wars or Douglas Adams' Hitch-hiker's Guide to the Galaxy does not count as comments. In fact, they simply count as anti-comments and will result in reduction of marks.
 - Comment your code in English. It is the language of instruction of the university.
- Proper variable names -- leia is not a good variable name, it never was and never will be.
- Use a small number of global variables, if any. Most programs need a very small number of global variables if any. (If you have a global variable named temp, think again.)
- The return values from all function calls should be checked and all values should be dealt with appropriately.

6. Assessment Tool

- You **must make it clear how to compile and run your code** by explaining it in your readme text file. The development environment should also be indicated in this file.
- TAs are not supposed to fix bugs, neither in your source code nor in your Makefile.

- If an executable file cannot be generated or executed successfully, it will be considered as unsuccessful.
- Any program that does not print the output in the format required in Section 1, will be considered as unsuccessful.
- If the program can be run successfully and the output is in the correct form, your program will be graded according to the following marking scheme in *Table 1*.
- **Table 1: Marking scheme.**

Components	Weight
Initialize socket	10%
Connect to server	10%
POST_STRING command	15%
POST_FILE command	10%
GET command	15%
EXIT command	10%
Other command	10%
Error handling	10%
programming style and in-program comments	10%

- Attention:
 - **Error handling refers to the ability of the client to detect incorrect inputs, e.g., illegal input IP address, incorrect instructions, the input file does not exist, etc.**
 - **Programming style and in-program comments means that your code should have a clear structure, detailed comments, and its usage instructions.**

7. Submission

- This assignment is to be done **individually or by a group of two students**. You are encouraged to discuss the high-level design of your solution with your classmates but you must implement the program on your own. Academic dishonesty such as copying another student's work or allowing another student to copy your work, is regarded as a serious academic offence.
- Each submission consists of two files: a source program file (e.g. .cpp, .c, .py or .java file), a readme (.txt) file telling us how to compile your code, and a Makefile if applicable.

- Use your student ID to name your submitted files, such as 5xxxxxxx-5xxxxxxx.cpp, 5xxxxxxx-5xxxxxxx-readme.txt, 5xxxxxxx.txt for submission.
- Submit the files to Canvas.
- No late submission will be accepted.

8. Getting Help

- If you have any programming issues, please feel free to contact the following TA:
- ZHAO Guanyi guanyzhao3-c@my.cityu.edu.hk (<mailto:guanyzhao3-c@my.cityu.edu.hk>)
- All contents in this page including source code and documents can only be used for course CS3201, October, 2023.

https://canvas.cityu.edu.hk/courses/36744/files/6529894/download?download_frd=1