

# CSE320 : System Fundamentals II

## Assignment 6

**DUE: 18-Nov-2024**

### Mastering Advanced C Programing Concepts - Exception, System Call and Process Control

#### Overview/Goals

The objective of this assignment is to reinforce your understanding of advanced C programming concepts. You will work on completing a simple 'echo' server application.

The assignment will give you experience with:

1. Understanding the network socket API.
2. Using `fork()` and `exec__()` to spawn child processes to handle requests.

#### Basic Program Specifications

The Socket API was introduced in the lecture and a simple Echo Server example demonstrated a TCP server-client implementation in C. This involves two main components: a server program that listens for incoming connections and a client program that initiates a connection to the server.

Using the example, you should implement a server-client program to send a message (max 50 characters) from a client to a server. The server receives it and print to the terminal. The server also can send a message back to the client. 1

In the server-client program, you need to open two terminals and execute the client and the server, respectively. For example, when you execute the echo server example, you can observe this behavior below on the server and client terminals.

Server:

```
â given ./server
pid: 64191, client: 127.0.0.1:15065
pid: 64191 done
```

Client

```
â given ./client
server: 127.0.0.1:28695
Hello
Hello
I'm Tony
I'm Tony
Nice to meet you!
Nice to meet you!
â given
```

Download assignment6.zip that includes server.c, client.c, util.c and util.h. Compile the code with your implementation and execute it. You should capture a screenshot of the execution result from the C code.

## Tasks

1. Download assignment6.zip provided in the assignment
2. Unzip the file which contains server.c, client.c, util.c, util.h, and a Makefile.
3. Build the code and test it. Capture a screenshot of the execution.
4. Implement remaining code with bug fixes and error checking for client-server echo application.

## Implementation

You must address the following:

1. Error handling: - Correct or insert any missing error cases and handle properly. You can print a error message in the error cases.
2. Exit condition:
  - When sending a “exit” message from the client or the server, exit both of them.
  - If the message is more than 50 characters long, print an error message and terminate both the server and the client.
3. Comment:
  - Add comments on any of your lines. Describe why you added them in details.
4. Compile:
  - Use Makefile to compile: Ex) make server; make client
5. Test modified code

## Testing the code

Example Session:

Client:

```
â Assign6 ./client
CSE320:socket successfully created.
CSE320:connected to the server.
CSE320:send a message : Hello
What is your name?
CSE320:received a message: Hello
CSE320:send a message : CSE320:received a message: This is the server!
CSE320:send a message : This is Tony!
CSE320:received a message: What about you?
CSE320:send a message : Good to see you.
CSE320:received a message: Good to see you to!
CSE320:send a message : Good
CSE320:received a message: Do you have anything interesting to say?
CSE320:send a message : Nope
Bye
CSE320:received a message: Okay
CSE320:send a message : CSE320:received a message: Bye
CSE320:send a message : exit
CSE320:exit client
â Assign6
```

Server:

```
â Assign6 ./server
CSE320:Socket successfully created.
CSE320:Socket successfully bound.
CSE320:server listening.
CSE320:server accepting client connection.
pid: 18887, client: 127.0.0.1:193
CSE320:received a message: Hello
CSE320:send a message: Hello
CSE320:received a message: What is your name?
CSE320:send a message: This is the server!
What about you?
CSE320:received a message: This is Tony!
CSE320:send a message: CSE320:received a message: Good to see you.
CSE320:send a message: Good to see you to!
Do you have anything interesting to say?
CSE320:received a message: Good
CSE320:send a message: CSE320:received a message: Nope
CSE320:send a message: Okay
CSE320:received a message: Bye
CSE320:send a message: Bye
CSE320:received a message: exit
CSE320:exit server
status: 0
```

## Reflection

- Document your code thoroughly with comments explaining each section.
- Prepare a report PDF document summarizing everything such as each step or functions that you implemented, challenges faced, and what you've learned during this assignment.


## Submission Instructions

**Deliverable Files:** Completed `client.c`, `server.c`, `util.c`, `util.h`, the provided Makefile, and your brief report packaged in a single zip or tgz (gzipped tar) file.

**Important:** Remove any debug output before submitting. This can make it difficult to judge the accuracy of the code against the expected results.

**Make sure you test the code in an Ubuntu environment. That is where I will test the code and if it does not build, has warnings on the build or crashes, some points will be lost.**

Please follow this procedure for submission:

1. Place the deliverable files into a folder by themselves. The folder's name should be `CSE320_HW6_<yourname>_<yourid>`. So if your name is Alice Kim and your id is 12345678, the folder should be named `'CSE320_HW6_AliceKim_12345678'`
2. Compress the folder and submit the zip file.
  - a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brightspace.
  - b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Brightspace.
3. Navigate to the course Brightspace site. Click **Assignments** in the top navbar menu. Look under the category 'Assignments'. Click **Assignment6**.
  - a. Scroll down and under **Submit Assignment**, click the **Add a File** button.
  - b. Click **My Computer** (first item in list).
  - c. Find the zip file and drag it to the 'Upload' area of the presented dialog box.
  - d. Click the **Add** button in the dialog.
  - e. You may write comments in the comment box at the bottom.
  - f. Click **Submit**.  Be sure to do this so I can retrieve the submission!

## Grading Criteria

The assignment will be graded for the following items:

Any errors in the compile reduce the grade by 50%.

Any warnings in the compile (but the code builds completely) reduces the grade by 30%.

If code crashes due to issues with improper use of data types, pointers, C language features, or the System API calls, this will reduce the grade by 30%.

If code does not crash but gives incorrect results or results that do not match the assignment specification, it will reduce the grade by 20%

If code is not well commented with reasonable variable names and consistent spacing and indentation, the grade may be reduced by up to 5%.

If the Assignment submission does not follow all directives (debut output is removed, includes required files, etc., is packaging (name of zip or tgz file.)...) the assignment may lose up to 5%.

Quality and completeness of report is worth 10% of the total points.