

CS 39006: Lab Test 2- Set A

Date: April 08, 2025

Important Note:

You have to follow the instructions and variable names given in this problem statement. Anything which is not given can be assumed; however, you should clearly write your assumptions at the beginning of the code.

Problem Statement:

You are required to implement a **TCP-based concurrent Echo Server** using C socket programming. The server should handle **multiple clients concurrently using `fork()`**, echo back messages sent by the client, and detect client disconnection **non-blockingly**.

Specifications:

1. Server Program (`tcp_echo_server.c`):

- Use **stream sockets (SOCK_STREAM)** with **IPv4 (AF_INET)**.
- Bind the server to **port 9090** on the local machine.
- Accept multiple clients **concurrently using `fork()`**.
- For each client:
 - Read messages from the client and echo them back.
 - Detect when the client disconnects and print a message "Client from <IP Address: Port> has disconnected normally". Note that the clients can disconnect through a CTRL+C interrupt. Here IP Address and Port are the address and port of the client. Note that you should not send these parameters explicitly to the server; the server should extract them from the connection information.
 - Set the server socket to **non-blocking mode** using `fcntl()` and handle the situation gracefully when `recv()` returns -1. When `recv()` returns with -1 due to blocking mode operation (EWOULDBLOCK is the error code), print a message "Client from <IP Address: Port> has disconnected abruptly".

2. Use `setsockopt()` to:

- Enable **SO_REUSEADDR**.
- Set the **receive buffer size** to 8 KB.
- Retrieve and print the buffer size using `getsockopt()`.

3. Client Program (`tcp_client.c`):

- Connects to the server at **127.0.0.1:9090**.
- Takes input from the user and sends it to the server.
- Receives the echoed message from the server and prints it.

- The above two steps are repeated until the user closes the connection abruptly through CTRL+C.

Function Prototypes at the Server: You need to implement the following functions apart from the main function.

```
/* Create a server socket by calling this from main with the port number*
```

```
int create_server_socket(int port);
```

```
/* handle a client connection after forking for every new connection. This  
function should implement echoing of the message and handling connection  
closure */
```

```
void handle_client(int client_fd);
```

```
/* setting the socket parameters using setsockopt() and retrieving the  
same using getsockopt()
```

```
void set_socket_options(int sockfd);
```

The client should implement everything under the main function.

Submission Instruction:

You have to submit the following two files: tcp_echo_server.c containing the server code and tcp_client.c containing the client code. **Submit a makefile to generate the executable files.** Put the files in a folder named LT2_SETA_<Your Roll Number> (for example, if your roll number is 22CS90098, then the folder name will be LT2_SETA_22CS90098. Compress the folder in a zip format and upload it on the MS Teams submission page. You have to follow the submission instructions exactly, otherwise a 20% penalty will be imposed.

Necessary Header Files:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <errno.h>  
#include <fcntl.h>  
#include <time.h>  
#include <arpa/inet.h>  
#include <netinet/in.h>  
#include <sys/socket.h>  
#include <sys/types.h>  
#include <sys/select.h>
```