## EE533-LAB1

### 1. Experimental Overview

The primary objective of this experiment is to create two Ubuntu virtual machines using VMware, run the experimental code on each of them, optimize the code according to the experimental requirements, and resolve issues related to zombie processes and cyclic receiving.

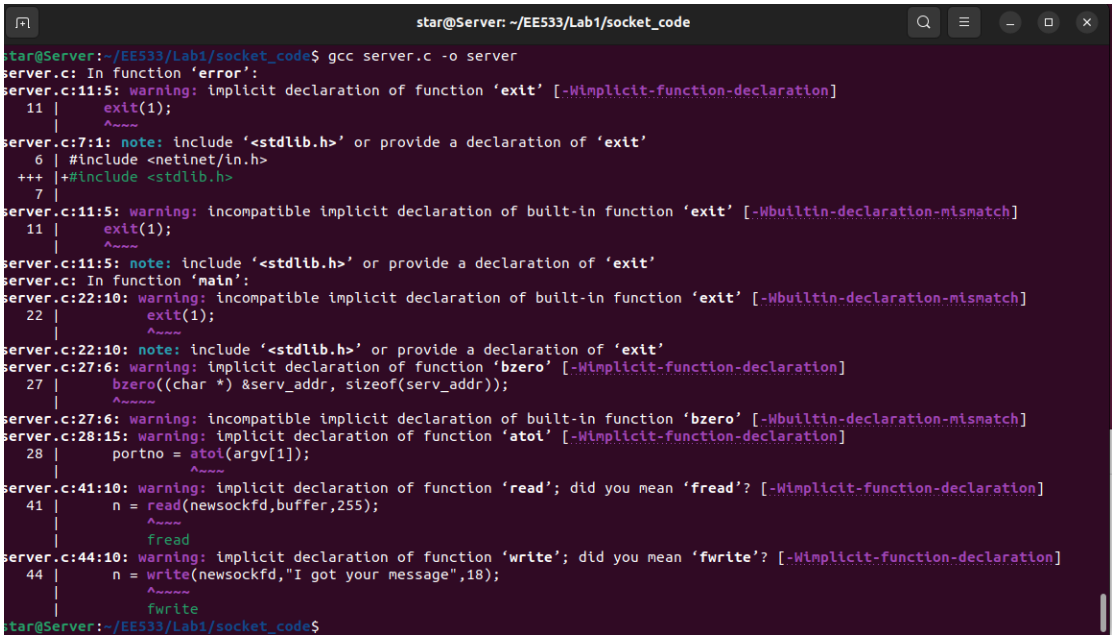
### 2. Core Concepts

**fork Function**: A system call in Linux for creating new processes. The parent process listens for new connections, while the child process handles communication with a single client, enabling multi-client concurrency.

**Zombie Process**: A residual process where child process resources are not reclaimed by the parent after termination. It can be resolved by using signal(SIGCHLD, SIG_IGN) to ignore the child process termination signal.

### 3. experiment

When running the original code, the following error occurred:

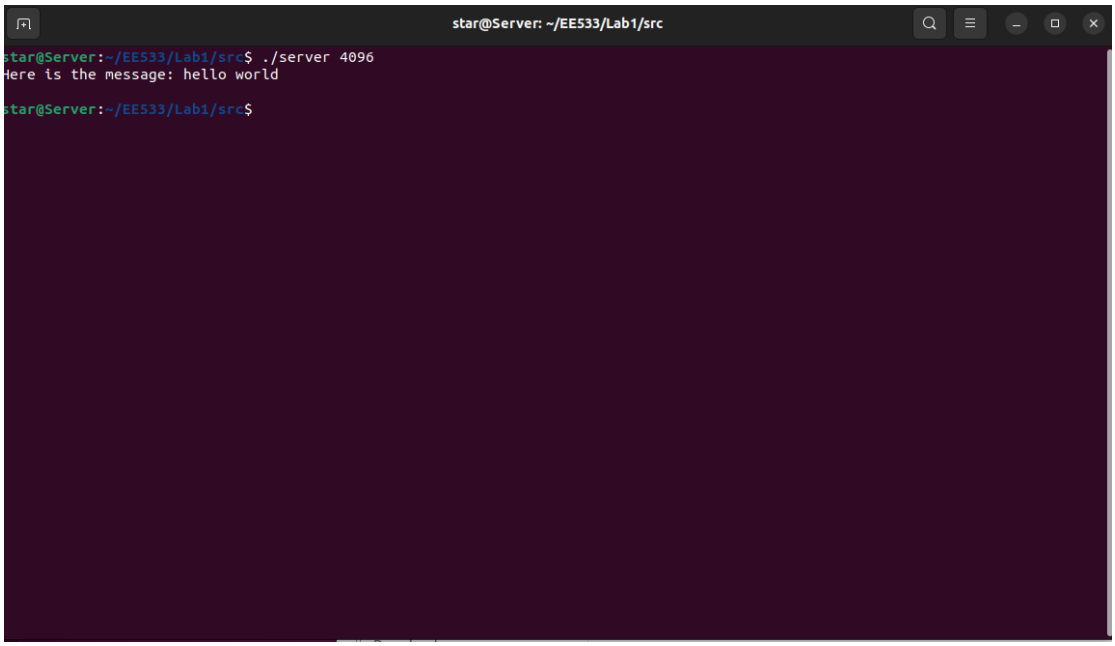


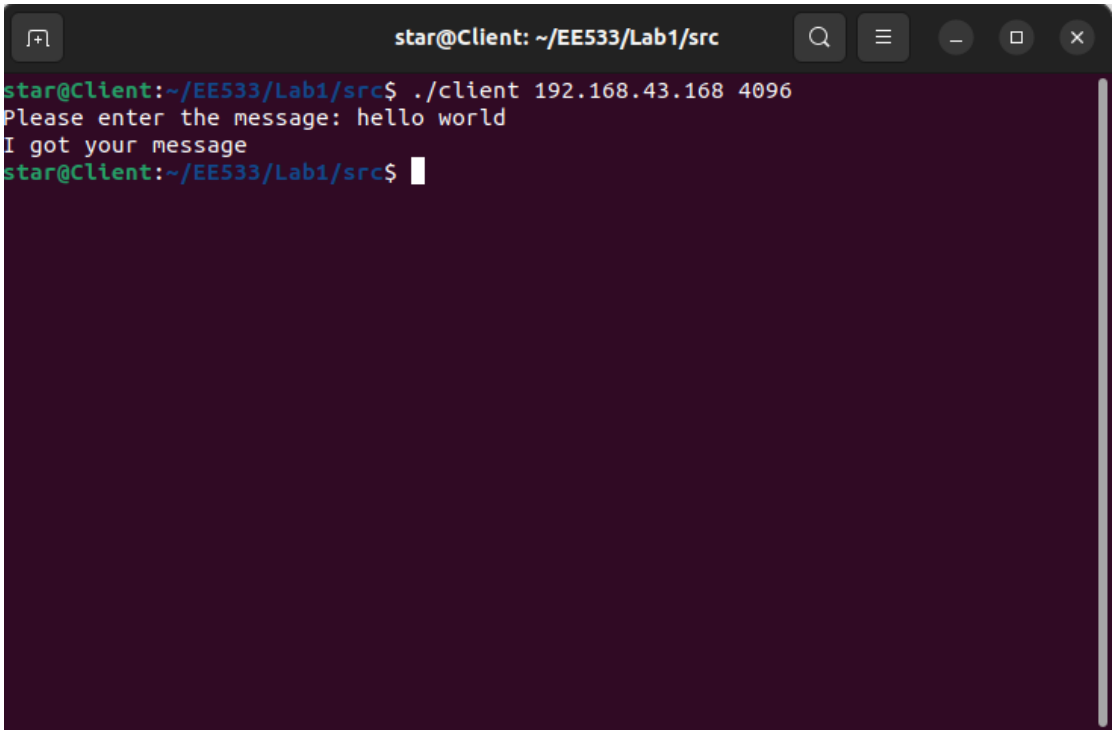

Based on the error messages, the following header files were added to the code to resolve the errors. (As required by the professor, I need to develop my own debugging tools to solve problems in future experiments.)

```
#include <stdlib.h>
#include <string.h>
```

```
#include <unistd.h>
```

The original code of the experiment runs as follows, information is typed in the client side, and then received by the server side.




According to the experimental documentation, we should solve the problem of multi-client communication. We need to add an infinite loop to the main function of the experimental code and use the fork() function to implement multi-task sending and receiving.

When the pid is 0, it indicates that the current process is the parent process, and we need to continue calling accept() to keep listening for new client connections.

When the pid is not 0, it indicates the current process is the child process. After executing the dostuff() function, the process shall be terminated.

Additionally, we need to use the SIGCHLD signal to resolve the zombie problem.

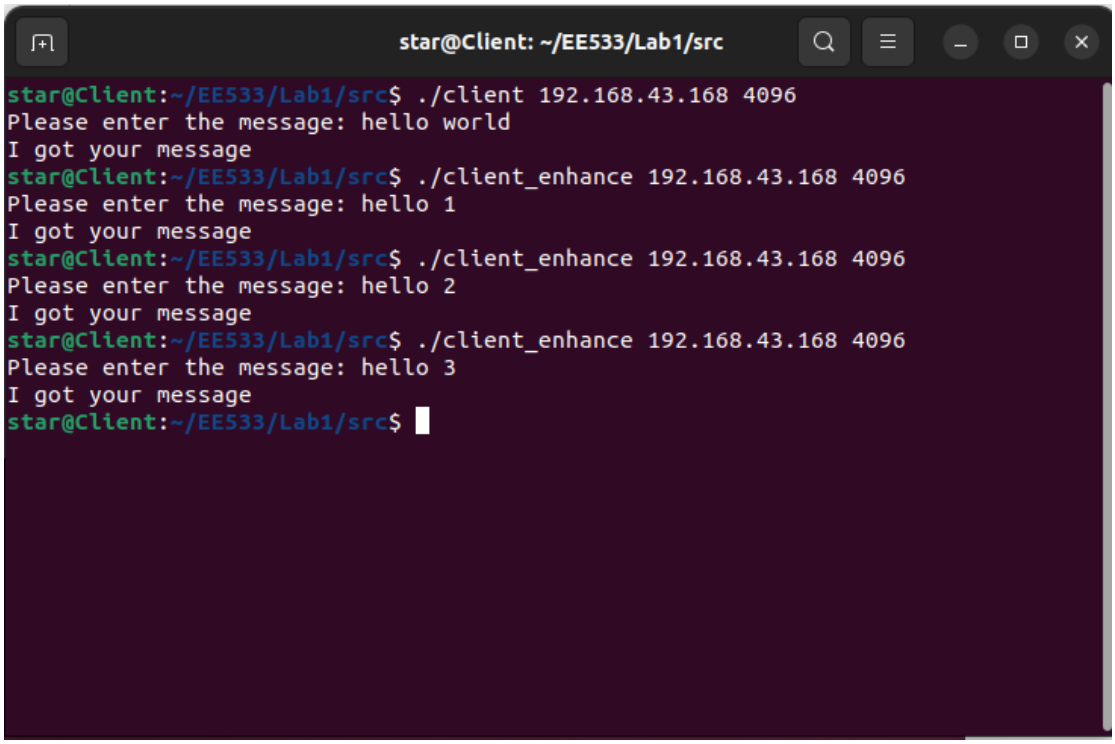
The complete code is shown as follows.

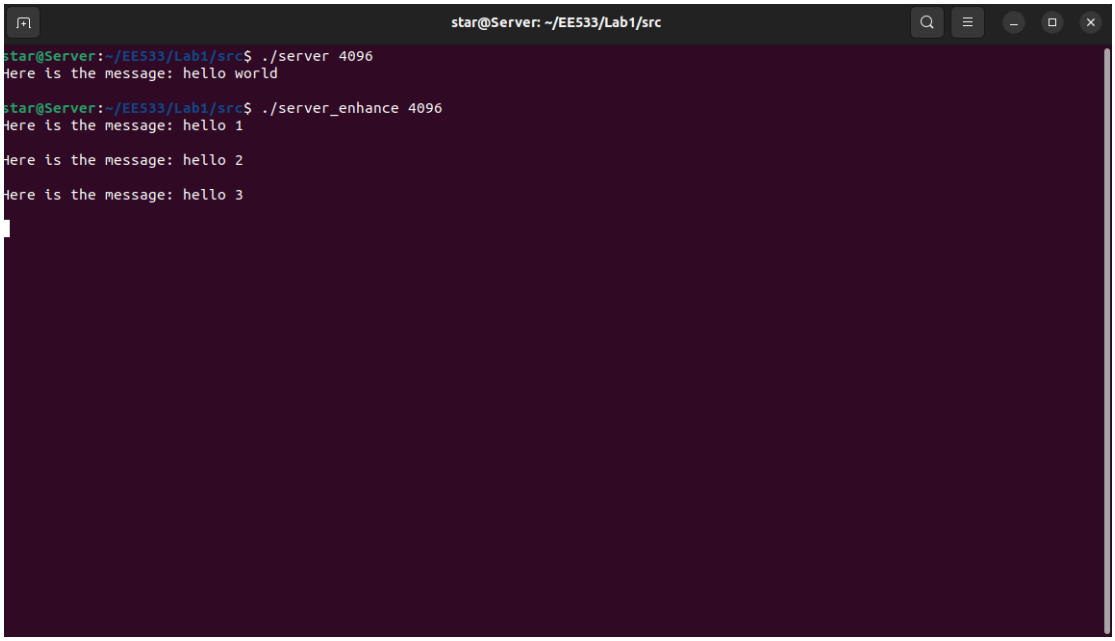
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
void error(char *msg)
{
    perror(msg);
    exit(1);
}
void dostuff(int newsockfd)
{
    char buffer[256];
    int n;
    bzero(buffer,256);
    n = read(newsockfd,buffer,255);
    if (n < 0)
    {
 error("ERROR reading from socket");
    }
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0)
    {
         error("ERROR writing to socket");
    }
}
int main(int argc, char *argv[])
{
     int newsockfd, sockfd, portno, clilen;

     struct sockaddr_in serv_addr, cli_addr;
```

```
     clilen = sizeof(cli_addr);

     signal(SIGCHLD, SIG_IGN);

     if (argc < 2)
     {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }

     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
     {
        error("ERROR opening socket");
     }

     bzero((char *) &serv_addr, sizeof(serv_addr));

     portno = atoi(argv[1]);

     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);

     if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)
) < 0)
     {
      error("ERROR on binding");
     }
     listen(sockfd,5);

     int pid;
     while(1)
     {
      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen
);
      if (newsockfd < 0)
      {
           error("ERROR on accept");
      }
 pid = fork();

 if(pid < 0)
 {
```

```
        error("ERROR on fork");
    }
    if(pid == 0)
    {
        close(sockfd);
        dostuff(newsockfd);

        exit(0);
    }
    else
    {
        close(newsockfd);
    }
      }
      return 0;
}
```

The verification of the improved code is shown as follows: the server can repeatedly receive multiple messages.




star@Server: ~/EE533/Lab1/src
star@Server:~/EE533/Lab1/src$ ./server 4096
Here is the message: hello world
star@Server:~/EE533/Lab1/src$ ./server_enhance 4096
Here is the message: hello 1
Here is the message: hello 2
Here is the message: hello 3