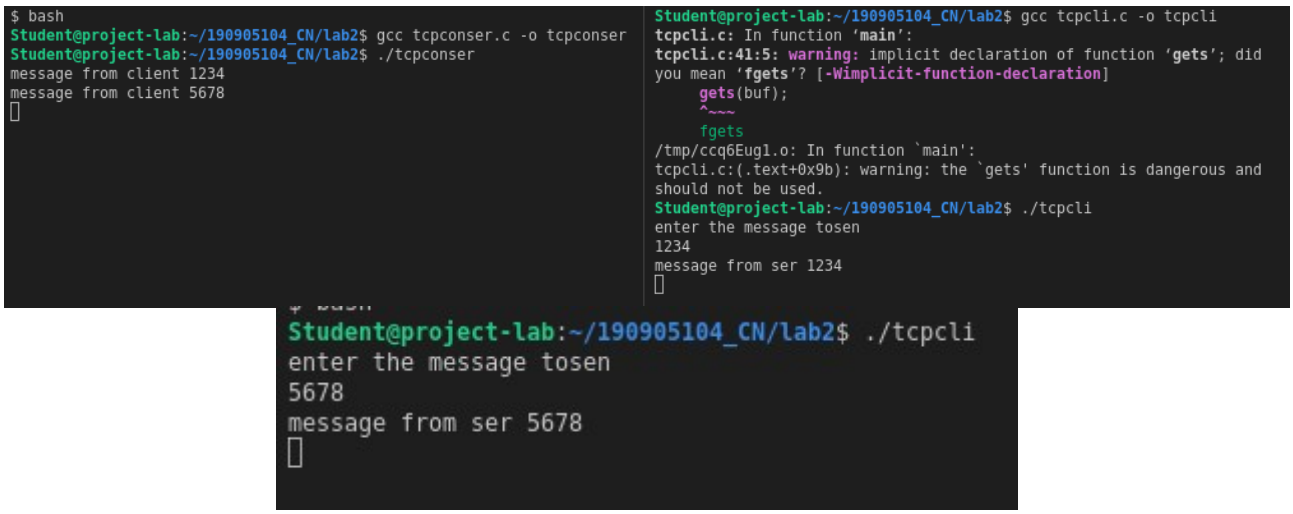
Parth Shukla
190905104
Lab 2

Write a server and client program to concurrent data transfer between client server using TCP protocol

// TCP server

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int main(){

    int sd,nd,n,len,reult;
    struct sockaddr_in seraddress, cliaddr;
    char buf[256], c;

    sd=socket(AF_INET, SOCK_STREAM, 0);

    seraddress.sin_family=AF_INET;
    seraddress.sin_addr.s_addr=INADDR_ANY;
    seraddress.sin_port=htons(10200);

    bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));

    listen(sd,5);
    len=sizeof(cliaddr);

    while(1){
        nd=accept(sd,(struct sockaddr*)&cliaddr,&len);
        if (fork()==0){
            close(sd);
            n=read(nd,buf,sizeof(buf));
            buf[n]='\0';

            printf("message from client %s\n",buf);
            n=write(nd,buf,strlen(buf));

            c = getchar();
            close(nd);
        }
```

```
    }
}

//TCP client

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int main(){
    int sd,nd,n,len,reult,n1;
    struct sockaddr_in seraddress, cliaddr;
    char buf[256], buf1[256];

    sd=socket(AF_INET, SOCK_STREAM,0);

    seraddress.sin_family=AF_INET;
    seraddress.sin_addr.s_addr=INADDR_ANY;
    seraddress.sin_port=htons(10200);

    len=sizeof(seraddress);

    connect(sd,(struct sockaddr*)&seraddress,len);

    printf("enter the message tosen \n");
    gets(buf);

    n=write(sd,buf,strlen(buf));

    n1=read(sd,buf1,sizeof(buf1));

    buf1[n1]='\0';
    printf("message from ser %s\n",buf1);
    getchar();
}
```



1) Write a TCP concurrent client server program where server accepts integer array from client and sorts it and returns it to the client along with process id.

// Server

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>


void sort(int arr[],int n){
        for(int i =0;i<n-1;i++){
                for(int j = 0;j<n-i-1;j++){
                        if(arr[j] > arr[j+1]){
                                int temp=arr[j];
                                arr[j] = arr[j+1];
                                arr[j+1] = temp;
                        }
                }
        }
}

int main(){
   int sd, nd, len, n, size;
   struct sockaddr_in seraddress, cliaddr;
```

```
    char buf[256], c;
    int buffer_int[256];
    printf("Initiating server\n");
    sd=socket(AF_INET, SOCK_STREAM, 0);
    if(sd == -1){
                    printf("Error in creating socket");
        exit(1);
            }
    seraddress.sin_family=AF_INET;
    seraddress.sin_addr.s_addr=INADDR_ANY;
    seraddress.sin_port=htons(10200);

    bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));

    listen(sd,5);
    len=sizeof(cliaddr);

    while(1){
        nd = accept(sd, (struct sockaddr *)&cliaddr, &len);
        puts("Connected to client");

        if(fork() == 0){
            // child process, only concerned with data transfer
            close(sd);
            int pid = getpid();
            n = read(nd, &size, sizeof(int));
            n = read(nd, buffer_int, size * sizeof(int));
            printf("array from client:\n");
            for(int i =0; i<size; i++){
                    printf("%d\t",buffer_int[i]);
            }
            sort(buffer_int, size);
            printf("\nProcess ID:%d",pid);
            n = write(nd, &pid, sizeof(int));
            n = write(nd, buffer_int, size * sizeof(int));
            getchar();
            close(nd);
        }
    }
}

// Client

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
```

```c
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int main(){
    int sd, len, n;
    struct sockaddr_in seraddress, cliaddr;
    int buffer_int[256];
    int size, pid;

    sd = socket(AF_INET, SOCK_STREAM, 0);
    if(sd == -1){
                    printf("Error in creating socket");
        exit(1);
            }
    seraddress.sin_family = AF_INET;
    seraddress.sin_addr.s_addr = INADDR_ANY;
    seraddress.sin_port = htons(10200);
    len = sizeof(seraddress);
    connect(sd, (struct sockaddr *)&seraddress, len);

    printf("Enter number of elements: \n");
    scanf("%d", &size);
    printf("Enter %d elements: \n", size);

    for (int i = 0; i < size; i++){
        scanf("%d", &buffer_int[i]);
    }

    n = write(sd, &size, sizeof(int));
    n = write(sd, buffer_int, size * sizeof(int));
    // printf("-------");
    n = read(sd, &pid, sizeof(int));
    n = read(sd, buffer_int, size * sizeof(int));
    printf("\nSorted array: ");
    for (int i = 0; i < size; i++){
        printf("%d ", buffer_int[i]);
    }
    printf("\nProcess ID: %d\n", pid);
    getchar();
}
```

```
Student@project-lab:~/190905104_CN/lab2$ gcc q1ser.c -o q1ser
Student@project-lab:~/190905104_CN/lab2$ ./q1ser
Initiating server
Connected to client
array from client:
4       3       7       1       5
Process ID:8905
```

```
Student@project-lab:~/190905104_CN/lab2$ gcc q1cli.c -o q1cli
Student@project-lab:~/190905104_CN/lab2$ ./q1cli
Enter number of elements:
5
Enter 5 elements:
4 3 7 1 5

Sorted array: 1 3 4 5 7
Process ID: 8905
```

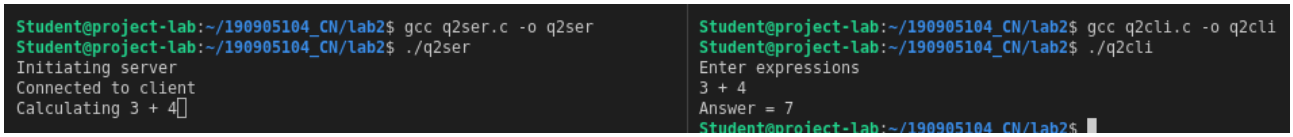
2) Implement concurrent Remote Math Server To perform arithmetic operations in the server and display the result at the client. The client accepts two integers and an operator from the user and sends it to the server. The server then receives integers and operator. The server will performs the operation on integers and sends result back to the client which is displayed on the client screen. Then both the processes terminate.

// Server

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int calc(int a, int b, char c){
   switch(c){
      case '+':
         return a + b;
         break;
      case '-':
         return a - b;
         break;
      case '/':
         return a / b;
         break;
      case '*':
         return a * b;
         break;
      default:
         return 0;
         break;
   }
}
int main(){
   int sd, nd, len, n, size;
   struct sockaddr_in seraddress, cliaddr;
   char buf[256], c;
   int buffer_int[256];
   printf("Initiating server\n");
   sd=socket(AF_INET, SOCK_STREAM, 0);
   if(sd == -1){
                  printf("Error in creating socket");
      exit(1);
         }
```

```
    seraddress.sin_family=AF_INET;
    seraddress.sin_addr.s_addr=INADDR_ANY;
    seraddress.sin_port=htons(10200);

    bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));

    listen(sd,5);
    len=sizeof(cliaddr);

    while(1){
        nd = accept(sd, (struct sockaddr *)&cliaddr, &len);
        puts("Connected to client");

        if(fork() == 0){
            // child process, only concerned with data transfer
            close(sd);
            n = read(nd, buffer_int, 2*sizeof(int));  // read integers
            n = read(nd, buf, 1*sizeof(char));
            printf("Calculating %d %c %d", buffer_int[0], buf[0], buffer_int[1]);
            int ans = calc(buffer_int[0], buffer_int[1], buf[0]);
            n = write(nd, &ans, sizeof(int));

            getchar();
            close(nd);
        }
    }
}

// Client

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int main(){
    int sd, len, n;
    struct sockaddr_in seraddress, cliaddr;
    char buf[256];
    int buffer_int[256];
    int size, pid;

    sd = socket(AF_INET, SOCK_STREAM, 0);
    if(sd == -1){
```

```
                printf("Error in creating socket");
        exit(1);
            }
    seraddress.sin_family = AF_INET;
    seraddress.sin_addr.s_addr = INADDR_ANY;
    seraddress.sin_port = htons(10200);
    len = sizeof(seraddress);
    connect(sd, (struct sockaddr *)&seraddress, len);
    printf("Enter expressions\n");
    scanf("%d %c %d", &buffer_int[0], &buf[0], &buffer_int[1]);

    n = write(sd, &buffer_int, 2*sizeof(int));
    n = write(sd, &buf, sizeof(char));
    int ans;
    n = read(sd, &ans, sizeof(int));
    printf("Answer = %d\n", ans);
    getchar();
    exit(0);
}
```



3) Implement simple TCP daytime server using fork.

// Server

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

int main(){
    int sd, nd, len, n, size;
    struct sockaddr_in seraddress, cliaddr;
    char buf[256], c;
    int buffer_int[256];
    printf("Initiating server\n");
    sd=socket(AF_INET, SOCK_STREAM, 0);
```

```
    if(sd == -1){
                    printf("Error in creating socket");
       exit(1);
          }
    seraddress.sin_family=AF_INET;
    seraddress.sin_addr.s_addr=INADDR_ANY;
    seraddress.sin_port=htons(10200);

    bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));

    listen(sd,5);
    len=sizeof(cliaddr);

    while(1){
       nd = accept(sd, (struct sockaddr *)&cliaddr, &len);
       puts("Connected to client");

       if(fork() == 0){
          // child process, only concerned with data transfer
          close(sd);
          int a[5];
          n = read(sd, &a, sizeof(int));
          printf("%c\n", a[0]);
          if(a[0] == 1){
             time_t cur_time;
             struct tm* timeinfo;
             time(&cur_time);
             timeinfo = localtime(&cur_time);
             int h = timeinfo->tm_hour;
             int m = timeinfo->tm_min;
             int s = timeinfo->tm_sec;
             printf("Time is %d:%d:%d\n", h, m, s);

             n = write(nd, &h, sizeof(int));
             n = write(nd, &m, sizeof(int));
             n = write(nd, &s, sizeof(int));
             getchar();
             close(sd);
             exit(0);
          }
       }
    }
}

// Client

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
```

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int main(){
   int sd, len, n;
   struct sockaddr_in seraddress, cliaddr;
   int buf[256];
   char c;

   sd = socket(AF_INET, SOCK_STREAM, 0);
   if(sd == -1){
      printf("Error in creating socket\n");
      exit(1);
   }
   seraddress.sin_family = AF_INET;
   seraddress.sin_addr.s_addr = INADDR_ANY;
   seraddress.sin_port = htons(10200);
   len = sizeof(seraddress);
   connect(sd, (struct sockaddr *)&seraddress, len);

   printf("Making request for time\n");
   int a[] = {1};
   n = write(sd, &a, sizeof(int));

   int h, m, s;
   n = read(sd, &h, sizeof(int));
   n = read(sd, &m, sizeof(int));
   n = read(sd, &s, sizeof(int));
   printf("Time is %d:%d:%d\n", h, m, s);
   getchar();
   exit(0);
}
```

```
^C
Student@project-lab:~/190905104_CN/lab2$ gcc q3ser.c -o q3ser
Student@project-lab:~/190905104_CN/lab2$ ./q3ser
Initiating server
Connected to client

Time is 9:55:31
```

```
^C
Student@project-lab:~/190905104_CN/lab2$ ./q3cli
Making request for time
Time is 9:55:31
```