LAB 2

Sample Program - Write a TCP concurrent Echo server and simple client.

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>
#define PORTNO 10200
int main()
{
        int sockfd,newsockfd,portno,clilen,n=1;
        char buf[256];
        struct sockaddr_in servaddr,cliaddr;
        int i,value;
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = INADDR_ANY;
        servaddr.sin_port = htons(PORTNO);
        bind(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
        listen(sockfd,5);
        while(1){
                clilen = sizeof(clilen);
                newsockfd=accept(sockfd,(struct sockaddr *)&cliaddr,&clilen);
                if(fork()==0){
                        n = read(newsockfd,buf,sizeof(buf));
                        printf(" \nMessage from Client %s \n",buf);
                        n = write(newsockfd,buf,sizeof(buf));
                        close(newsockfd);
                        exit(0);
                }
                else{
                        close(newsockfd);
                }
```

```
	}
	return 0;
}
```

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>
#define PORTNO 10200
int main()
{
	int sockfd,newsockfd,portno,clilen,n=1;
	char buf[256];
	struct sockaddr_in servaddr,cliaddr;
	int i,value;
	sockfd = socket(AF_INET,SOCK_STREAM,0);
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = INADDR_ANY;
	servaddr.sin_port = htons(PORTNO);
	bind(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
	listen(sockfd,5);
	while(1){
		clilen = sizeof(clilen);
		newsockfd=accept(sockfd,(struct sockaddr *)&cliaddr,&clilen);
		if(fork()==0){
			n = read(newsockfd,buf,sizeof(buf));
			printf(" \nMessage from Server %s \n",buf);
			n = write(newsockfd,buf,sizeof(buf));
			close(newsockfd);
			exit(0);
		}
		else{
			close(newsockfd);
		}
	}
```

```
        return 0;
}
```

Q1) 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) {
    for(int i=0;i<4;i++) {
        int min=i;
        for(int j=i;j<5;j++) {
            if(arr[j]<arr[min]) min=j;
        }
        if(i!=min) {
            int temp=arr[i];
            arr[i]=arr[min];
            arr[min]=temp;
        }
    }
}
void main()
{
   int sd,nd,n,len,reult;
   struct sockaddr_in seradress,cliaddr;
   int buf[5];
   sd=socket(AF_INET,SOCK_STREAM,0);
   seradress.sin_family=AF_INET;
   seradress.sin_addr.s_addr=htonl(INADDR_ANY);
   seradress.sin_port=htons(10200);
   bind(sd,(struct sockaddr*)&seradress,sizeof(seradress));
```

```
  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));
        if(n==sizeof(buf))printf("Receieved array\n");
        sort(buf);
        n=write(nd,buf,sizeof(buf));
        if(n==sizeof(buf))printf("Sent array\n");
     }
     close(nd);
  }
}
```

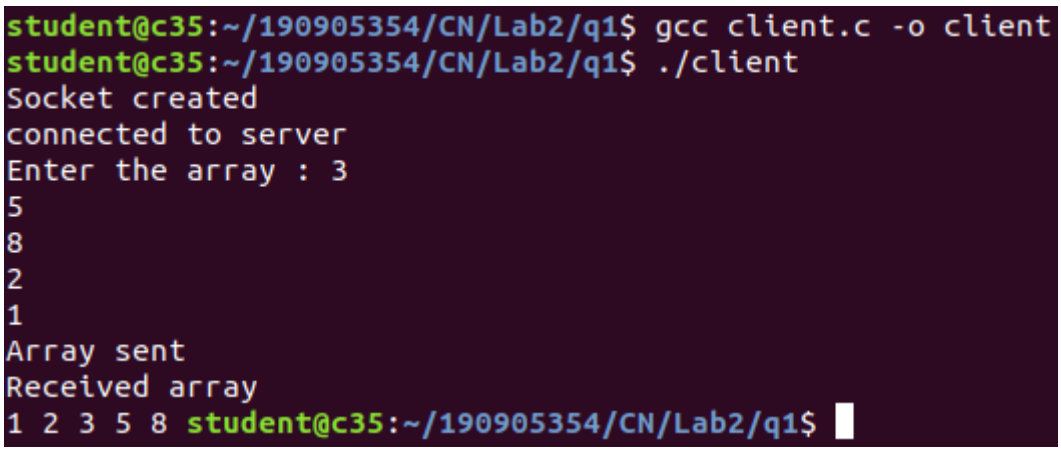
CLIENT -

```
#include <unistd.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 5
#define PORT 10200
#define SA struct sockaddr
void client_func(int sockfd)
{
  int buff[MAX];
  int n;
  bzero(buff, sizeof(buff));
  printf("Enter the array : ");
  for(int i=0;i<MAX;i++){
      scanf("%d",&buff[i]);
  }
  n = 0;
  n=write(sockfd, buff, sizeof(buff));
  if(n==sizeof(buff))
  {
     printf("Array sent\n");
  }
  bzero(buff, sizeof(buff));
  n=read(sockfd, buff, sizeof(buff));
```

```
  if(n==sizeof(buff))
  {
     printf("Received array \n");
     for(int i=0;i<MAX;i++){
         printf("%d ",buff[i]);
     }
  }
}
int main()
{
  int sockfd, connfd;
  struct sockaddr_in servaddr, cli;
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd == -1) {
     printf("socket creation failed\n");
     exit(0);
  }
  else{
     printf("Socket created\n");
  }
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servaddr.sin_port = htons(PORT);
  if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
     printf("connection failed\n");
     exit(0);
  }
  else{
     printf("connected to server\n");
  }
  client_func(sockfd);
  close(sockfd);
}
```

(PTO)

```
student@c35:~/190905354/CN/Lab2/q1$ gcc server.c -o server
student@c35:~/190905354/CN/Lab2/q1$ ./server
Receieved array
Sent array successfully
```



Q2) 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 <string.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAXSIZE 150
#define PORT 5000
#define MAXLINE 1000
typedef struct obj
{
   double a,b,r;
   char op;
   char ans[10];
}obj1,*obj_ptr;
int main()
{

   int sockfd,newsockfd,ret_val,sock_len, actual_len;
   int recvd_bytes,sent_bytes, sent_ans;
   struct sockaddr_in serveraddr,clientaddr;
```

```
    obj_ptr buff = (obj_ptr)malloc(sizeof(obj1));
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd==-1){
       printf("\nSocket creation failed");
    }
    serveraddr.sin_family=AF_INET;
    serveraddr.sin_port=htons(PORT);
    serveraddr.sin_addr.s_addr=htons(INADDR_ANY);
    bind(sockfd,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
    puts("Server Running");
    listen(sockfd,1);
    actual_len=sizeof(clientaddr);
    newsockfd=accept(sockfd,(struct sockaddr*)&clientaddr,&actual_len);
    do
    {
       recv(newsockfd,buff,sizeof(obj1),0);
       if(strcmp(buff->ans, "No") == 0)
       {
          puts("Stopping...");
          close(sockfd);
          close(newsockfd);
       }
       else
       {
          printf("Client [%s:%d] requested: %.2lf %c %.2lf\n",
inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port), buff->a, buff->op, buff-
>b);
          switch (buff->op)
          {
             case '+': buff->r = buff->a + buff->b;
                break;
             case '-': buff->r = buff->a - buff->b;
                break;
             case '*': buff->r = buff->a * buff->b;
                break;
             case '/': buff->r = buff->a / buff->b;
                break;
             case '%': buff->r = buff->a / buff->b;
                break;
             default: printf("Invalid operator");
                break;
          }
          sent_bytes = send(newsockfd,buff,sizeof(obj1),0);
       }
    }while(strcmp(buff->ans, "No") != 0);
```

```
   return 0;
}
```

CLIENT -

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#define MAXSIZE 150
#define PORT 5000
#define MAXLINE 1000
typedef struct obj
{
   double a,b,r;
   char op;
   char ans[10];
}obj1,*obj_ptr;
int main()
{
   int sockfd,ret_val;
   char ch;
   int recvd_bytes,sent_bytes, recvd_ans;
   struct sockaddr_in serveraddr;
   obj_ptr buff = (obj_ptr)malloc(sizeof(obj1));
   obj_ptr buff1 = (obj_ptr)malloc(sizeof(obj1));
   sockfd=socket(AF_INET,SOCK_STREAM,0);
   if(sockfd==-1){
      printf("\nSocket Creation failed");
   }
   printf("\nSocket ID : %d\n",sockfd);
   serveraddr.sin_family=AF_INET;
   serveraddr.sin_port=htons(PORT);
   serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
   ret_val=connect(sockfd,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
   if(ret_val==-1){
      printf("Connection failed");
   }
   do
```

```
    {
        printf("Do you want to request? Yes/Stop\n");
      scanf("%c",&ch);
      scanf("%[^\n]%*c",(buff->ans));
      if(strcmp(buff->ans,"No")==0)
      {
         puts("Stopping");
         sent_bytes=send(sockfd,buff,sizeof(buff),0);
         close(sockfd);
      }
      else
      {
        printf("Enter the maths - a op b : \n");
        scanf("%lf %c %lf",&buff->a, &buff->op, &buff->b);
        sent_bytes=send(sockfd,buff,sizeof(obj1),0);
                recvd_bytes=recv(sockfd,buff1,sizeof(obj1),0);
        printf("Result = %.2lf \n",buff1->r);
      }
    }while(strcmp(buff->ans, "No") != 0);
    return 0;
}
```

```
student@c35:~/190905354/CN/Lab2/q2$ gcc server.c -o server
student@c35:~/190905354/CN/Lab2/q2$ ./server
Server Running
Client [127.0.0.1:44688] requested: 3.00 + 4.00
Client [127.0.0.1:44688] requested: 5.00 * 7.00
Client [127.0.0.1:44688] requested: 5.00 - 3.00
Client [127.0.0.1:44688] requested: 12.00 / 6.00
```

```
student@c35:~/190905354/CN/Lab2/q2$ gcc client.c -o client
student@c35:~/190905354/CN/Lab2/q2$ ./client

Socket ID : 3
Continue? Yes/No
Yes
Enter the maths - a op b :
3 + 4
Result = 7.00
Continue? Yes/No
Yes
Enter the maths - a op b :
5 * 7
Result = 35.00
Continue? Yes/No
Yes
Enter the maths - a op b :
5 - 3
Result = 2.00
Continue? Yes/No
Yes
Enter the maths - a op b :
12 / 6
Result = 2.00
Continue? Yes/No
No
Stopping..
student@c35:~/190905354/CN/Lab2/q2$
```

Q3) Implement simple TCP daytime server using select().

SERVER -

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    time_t rawtime;
    struct tm * timeinfo;
    char *reply;
    int server_sockfd, client_sockfd;
    int server_len, client_len;
    struct sockaddr_in server_address;
```

```
   struct sockaddr_in client_address;
   int hour,mins,sec,pid;
   server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
   server_address.sin_family = AF_INET;
   server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
   server_address.sin_port = 9734;
   server_len = sizeof(server_address);
   bind(server_sockfd, (struct sockaddr *)&server_address, server_len);
   listen(server_sockfd, 5);
   while(1)
   {
      char ch;
      printf("server waiting\n");
      client_len = sizeof(client_address);
      client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_address,
&client_len);
      char * ip_add =inet_ntoa(client_address.sin_addr);
      int port=client_address.sin_port;
      printf("IP:%s  Port:%d\n", ip_add,port);
      time ( &rawtime );
      timeinfo = localtime ( &rawtime );
      reply = asctime(timeinfo);
      printf ( "The time is: %s", reply );
      hour =  timeinfo->tm_hour;
      mins = timeinfo->tm_min;
      sec = timeinfo->tm_sec;
      pid = getpid();
      write(client_sockfd, &hour, 1);
      write(client_sockfd, &mins, 1);
      write(client_sockfd, &sec, 1);
      write(client_sockfd, &pid, 1);
      exit(0);
   }
   return 0;
}
```

CLIENT -

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
```

```
#include <time.h>
int main()
{
   int sockfd;
   int len;
   struct sockaddr_in address;
   struct tm * timeinfo;
   int result;
   char *reply;
   int hr,mins,sec,pid;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = inet_addr("127.0.0.1");
   address.sin_port = 9734;
   len = sizeof(address);
   result = connect(sockfd, (struct sockaddr *)&address, len);
   if(result == -1)
   {
      printf("Failed\n");
      exit(0);
   }
   printf(" Getting the time\n");
   read(sockfd, &hr , 1);
   read(sockfd, &mins , 1);
   read(sockfd, &sec , 1);
   read(sockfd, &pid , 1);
   printf("%d:%d:%d", hr, mins, sec);
   printf(" The process id: %d \n",pid);
   close(sockfd);
   return 0;
}
```

```
student@c35:~/190905354/CN/Lab2/q3$ gcc server.c -o server
student@c35:~/190905354/CN/Lab2/q3$ ./server
server waiting
IP:127.0.0.1  Port:33984
The time is: Sat Oct 23 14:48:58 2021
student@c35:~/190905354/CN/Lab2/q3$
```

```
student@c35:~/190905354/CN/Lab2/q3$ gcc client.c -o client
student@c35:~/190905354/CN/Lab2/q3$ ./client
 Getting the time
14:1824856624:22074 The process id: 1709640683
student@c35:~/190905354/CN/Lab2/q3$
```