LAB 2

Juhi Mehta
190905412
Roll No: 55
Batch B3

Solved Example for Concurrent TCP Client/Server

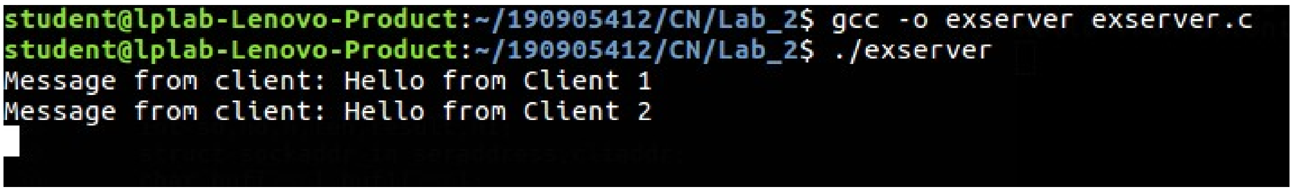
Code:

```
//server side

#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,result;
	struct sockaddr_in seraddress, cliaddr;
	char buf[256];

	sd=socket(AF_INET, SOCK_STREAM, 0);
	seraddress.sin_family=AF_INET;
	seraddress.sin_addr.s_addr=inet_addr("172.16.57.83");
	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));
			printf("Message from client: %s\n",buf);
			getchar();
		}
		close(nd);
	}
}
```

```
//client side

#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,result,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 to send: \n");
	gets(buf);
	n=write(sd,buf,strlen(buf));
	n1=read(sd,buf1,sizeof(buf1));
	printf("Message from server: %s\n",buf1);
	getchar();
}
```

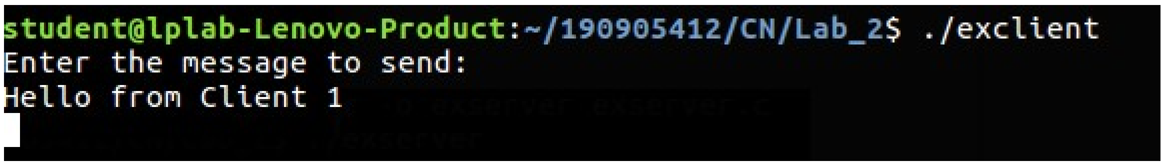
Output:

//server



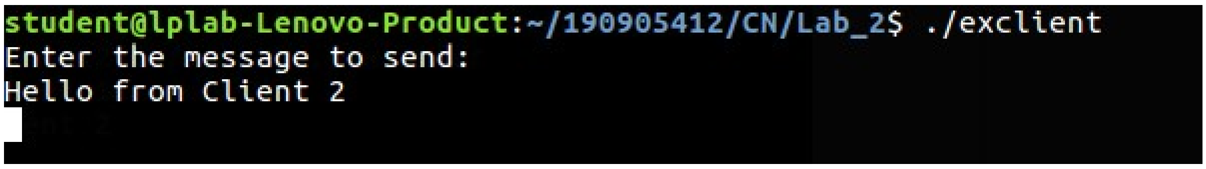
```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ gcc -o exserver exserver.c
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./exserver
Message from client: Hello from Client 1
Message from client: Hello from Client 2
```

//client 1



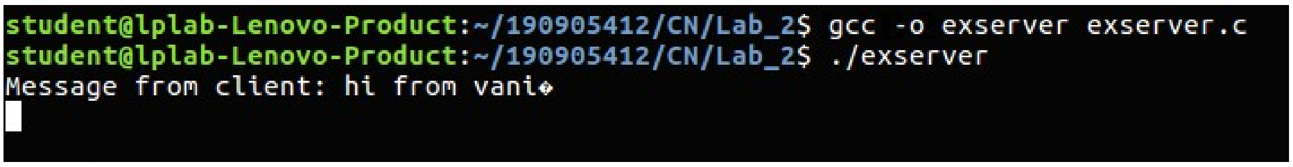
```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./exclient
Enter the message to send:
Hello from Client 1
```

//client 2



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./exclient
Enter the message to send:
Hello from Client 2
```

//When I was server



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ gcc -o exserver exserver.c
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./exserver
Message from client: hi from vani◆
```

1) Server accepts integer array and sorts it and returns to client with pid

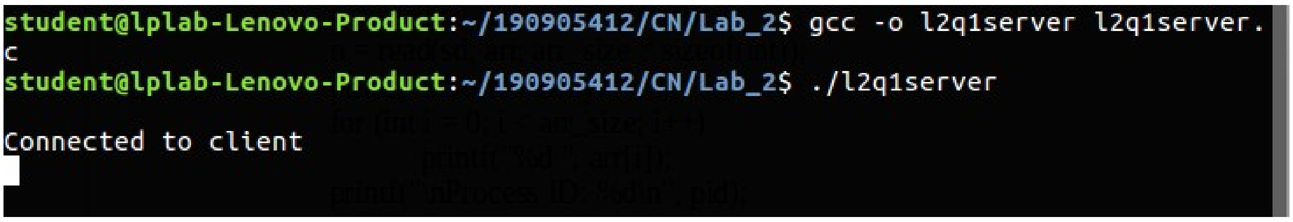
Code:

```
//server side

#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 compare(const void *a, const void *b)
{
        return (*(int *)a - *(int *)b);
}

int main()
{
        int sd, nd, len, n;
        struct sockaddr_in seraddress, cliaddr;
        int arr[20];
        int arr_size = 0;

        sd = socket(AF_INET, SOCK_STREAM, 0);
        seraddress.sin_family = AF_INET;
        seraddress.sin_addr.s_addr = inet_addr("172.16.57.83");
        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);
                printf("Connected to client");
                if (fork() == 0)
                {
                        close(sd);
                        int pid = getpid();
                        n = read(nd, &arr_size, sizeof(int));
                        n = read(nd, arr, arr_size * sizeof(int));

                        //Sort
                        qsort(arr, arr_size, sizeof(int), compare);
                        n = write(nd, &pid, sizeof(int));
                        n = write(nd, arr, arr_size * sizeof(int));
                        getchar();
                        close(nd);
                }
        }
}

//client side

#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;
        int arr[20];
        int arr_size, pid;

        sd = socket(AF_INET, SOCK_STREAM, 0);
        seraddress.sin_family = AF_INET;
        seraddress.sin_addr.s_addr = inet_addr("172.16.57.83");
        seraddress.sin_port = htons(10200);

        len = sizeof(seraddress);
        connect(sd, (struct sockaddr *)&seraddress, len);
        printf("Enter number of elements: \n");
```

```
	scanf("%d", &arr_size);
	printf("Enter elements: \n");
	for (int i = 0; i < arr_size; i++)
		scanf("%d", &arr[i]);
	n = write(sd, &arr_size, sizeof(int));
	n = write(sd, arr, arr_size * sizeof(int));
	n = read(sd, &pid, sizeof(int));
	n = read(sd, arr, arr_size * sizeof(int));
	printf("\nSorted array: ");
	for (int i = 0; i < arr_size; i++)
		printf("%d ", arr[i]);
	printf("\nProcess ID: %d\n", pid);
	getchar();
}
```

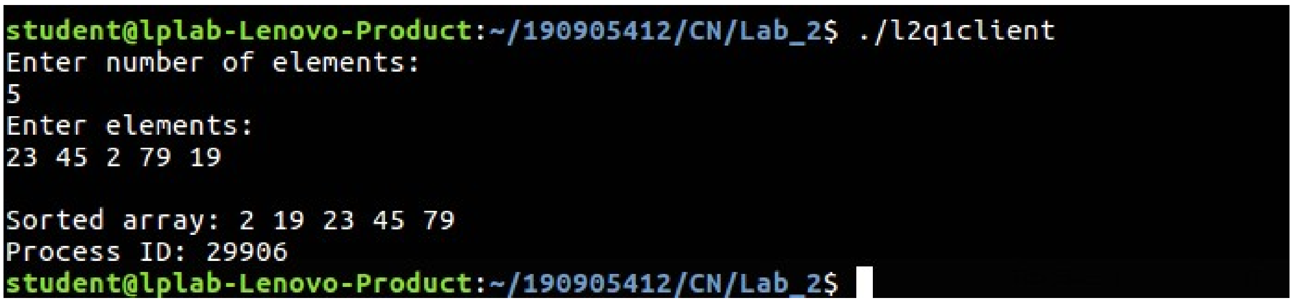
Output:

//When I was the server



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ gcc -o l2q1server l2q1server.
c
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./l2q1server

Connected to client
```

//When I was the client



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./l2q1client
Enter number of elements:
5
Enter elements:
23 45 2 79 19

Sorted array: 2 19 23 45 79
Process ID: 29906
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$
```

2) Calculator using Concurrent TCP Client Server Interaction

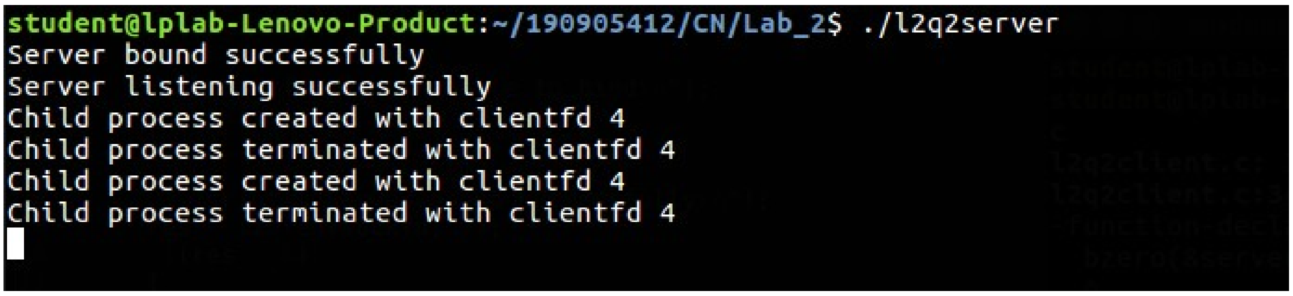
Code:

```
//server side

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#define PORT 5000

int calc(int a, int b, char operator)
```

```
{
	switch(operator)
	{
		case '+':
			return a + b;
		break;
		case '-':
			return a - b;
		break;
		case '/':
			return a / b;
		break;
		case '*':
			return a * b;
		break;
		default:return 0;
		break;
	}
}

void servfunc(int sockfd, struct sockaddr_in server_address)
{
	struct sockaddr_in client_address;
	int clientfd, a, b, res, size = sizeof(client_address);
	char op;
	while (1)
	{
		clientfd = accept(sockfd, (struct sockaddr *)&client_address, &size);
		if (fork() == 0)
		{
			printf("Child process created with clientfd %d\n", clientfd);
			close(sockfd);
			read(clientfd, (int *)&a, sizeof(int));
			read(clientfd, (int *)&b, sizeof(int));
			read(clientfd, (char *)&op, sizeof(char));
			res = calc(a, b, op);
			write(clientfd, (int *)&res, sizeof(int));
			close(clientfd);
			printf("Child process terminated with clientfd %d\n", clientfd);
			exit(0);
		}
		else
			close(clientfd);
	}
	printf("Server closing\n");
}

int main()
{
	int sockfd;
	struct sockaddr_in server_address;
	bzero(&server_address, sizeof(server_address));
	server_address.sin_family = AF_INET;
	server_address.sin_port = htons(PORT);
	server_address.sin_addr.s_addr = inet_addr("172.16.57.83");
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	int res = bind(sockfd, (struct sockaddr *)&server_address, sizeof(server_address));
	if(res < 0)
	{
		printf("Server unable to bind\n");
		exit(0);
	}
	else
		printf("Server bound successfully\n");
	res = listen(sockfd, 2);
	if(res < 0)
	{
		printf("Server unable to listen\n");exit(0);
	}
	else
		printf("Server listening successfully\n");
	servfunc(sockfd, server_address);
	close(sockfd);
}

//client side

#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>
#define PORT 5000

void clifunc(int sockfd)
{
	printf("This is the client by Juhi Mehta 190905412\n");
	int a, b;
	char c;
	printf("Enter the expression as you would on a Calculator: \n");
	scanf("%d%c%d", &a, &c, &b);
	write(sockfd, (int *)&a, sizeof(int));
	write(sockfd, (int *)&b, sizeof(int));
	write(sockfd, (char *)&c, sizeof(char));
	int res;
	read(sockfd, (int *)&res, sizeof(int));
	printf("%d %c %d = %d\n", a, c, b, res);
	printf("Client closing\n");
}

int main(int argc, char const *argv[])
{
	int sockfd;
	int len;
	struct sockaddr_in server_address;
	int result;
	char ch;
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	bzero(&server_address, sizeof(server_address));
	server_address.sin_family = AF_INET;
	server_address.sin_port = htons(PORT);
	server_address.sin_addr.s_addr = inet_addr("172.16.57.83");
	len = sizeof(server_address);
	result = connect(sockfd, (struct sockaddr *)&server_address, len);
	if(result == -1)
	{
		printf("Connection error\n");
		exit(0);
	}
	clifunc(sockfd);
	close(sockfd);
}
```

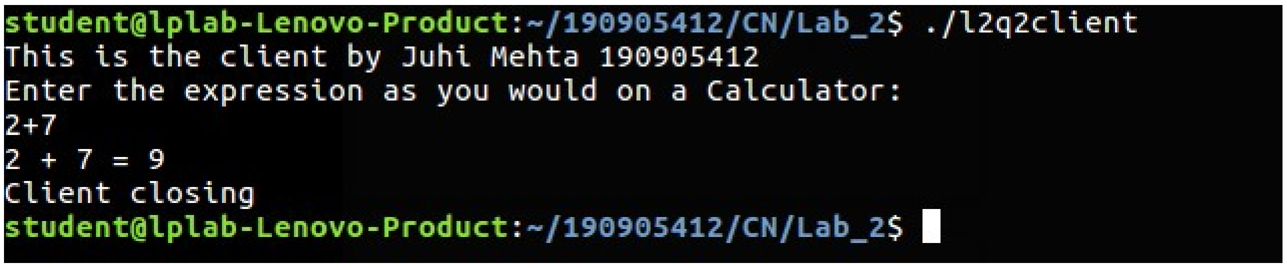
Output:

//When I was server



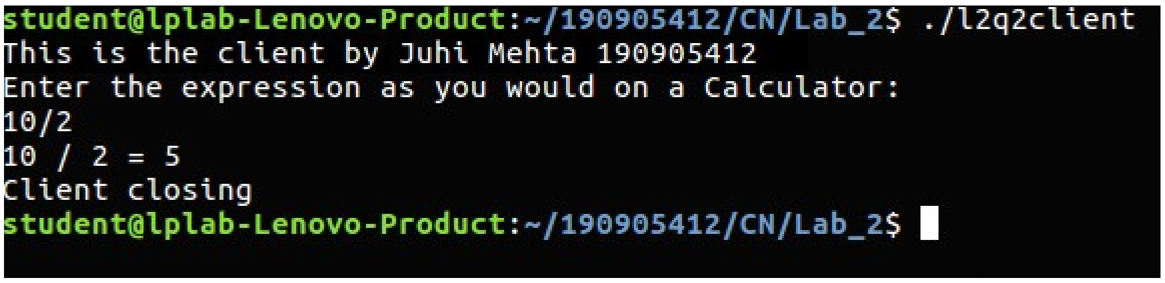
```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./l2q2server
Server bound successfully
Server listening successfully
Child process created with clientfd 4
Child process terminated with clientfd 4
Child process created with clientfd 4
Child process terminated with clientfd 4
```

//When I was client 1



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./l2q2client
This is the client by Juhi Mehta 190905412
Enter the expression as you would on a Calculator:
2+7
2 + 7 = 9
Client closing
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$
```

//When I was client 2



```
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$ ./l2q2client
This is the client by Juhi Mehta 190905412
Enter the expression as you would on a Calculator:
10/2
10 / 2 = 5
Client closing
student@lplab-Lenovo-Product:~/190905412/CN/Lab_2$
```