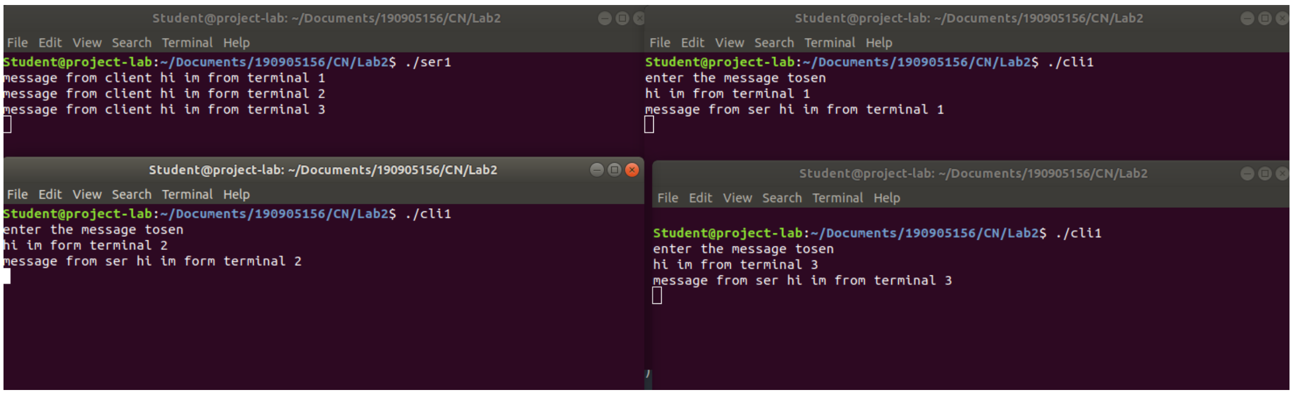
CN Lab 2

Name:Rhea Adhikari
Class :Section D
Reg No:190905156
Roll No:23

Q) Socket Programming in ‘C’ using TCP- Concurrent Client-Server Programs

Solved Example



```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./ser1
message from client hi im from terminal 1
message from client hi im form terminal 2
message from client hi im from terminal 3
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli1
enter the message tosen
hi im from terminal 1
message from ser hi im from terminal 1
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli1
enter the message tosen
hi im form terminal 2
message from ser hi im form terminal 2
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli1
enter the message tosen
hi im from terminal 3
message from ser hi im from terminal 3
```

client.c

```
#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();
}
```

server.c

```
#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];
        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));
                        getchar();
                        close(nd);
                }
        }
}
```

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.c

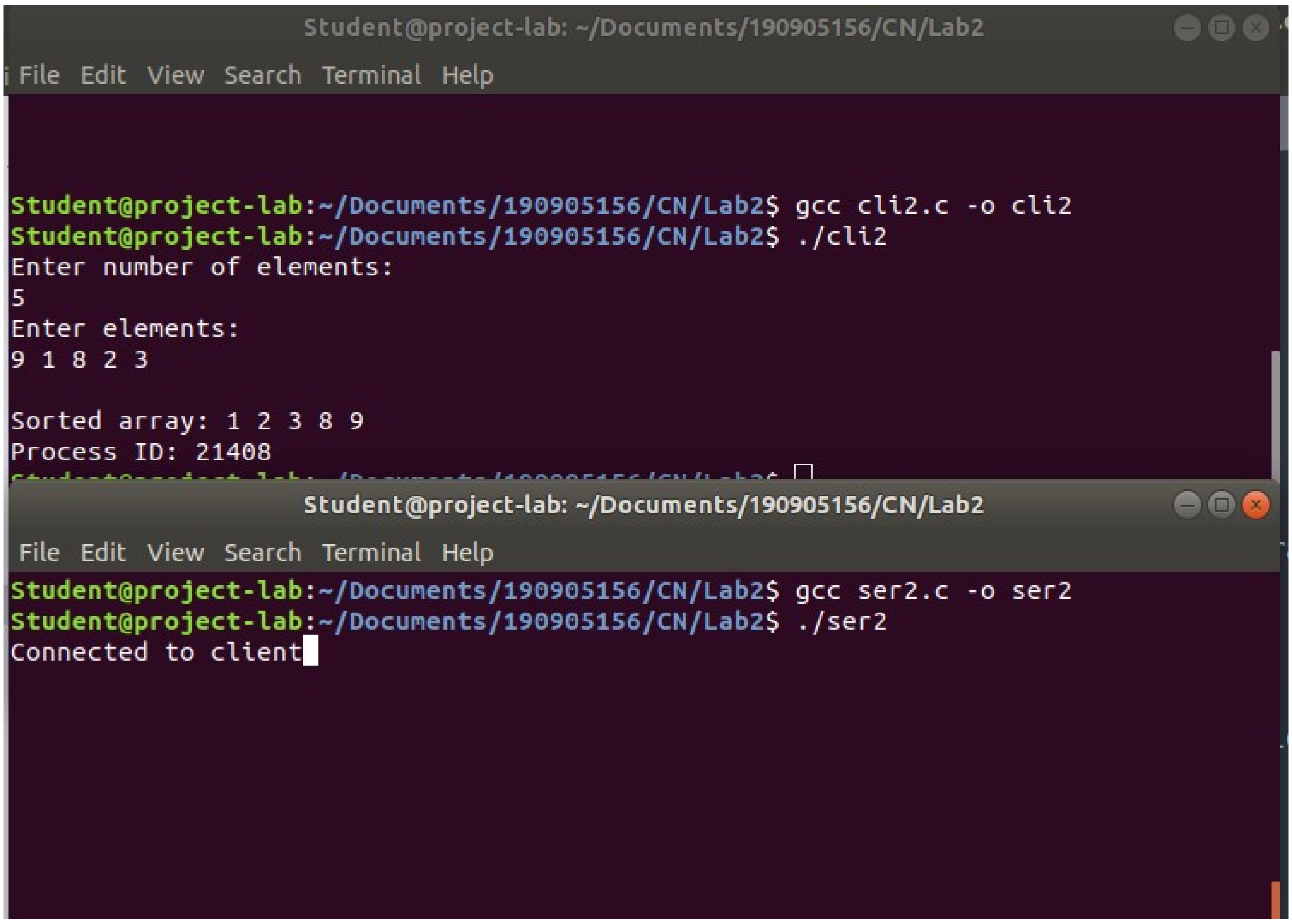
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

int cmpfunc(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 = 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);
		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), cmpfunc);
			n = write(nd, &pid, sizeof(int));
			n = write(nd, arr, arr_size * sizeof(int));
			getchar();
			close(nd);
		}
	}
}
```

client.c

```
#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 = INADDR_ANY;
	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();
}
```



```
Student@project-lab:~/Documents/190905156/CN/Lab2$ gcc cli2.c -o cli2
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli2
Enter number of elements:
5
Enter elements:
9 1 8 2 3

Sorted array: 1 2 3 8 9
Process ID: 21408
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ gcc ser2.c -o ser2
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./ser2
Connected to client
```

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 theclient which is displayed on the client screen. Then both the processes terminate.

Server.c

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

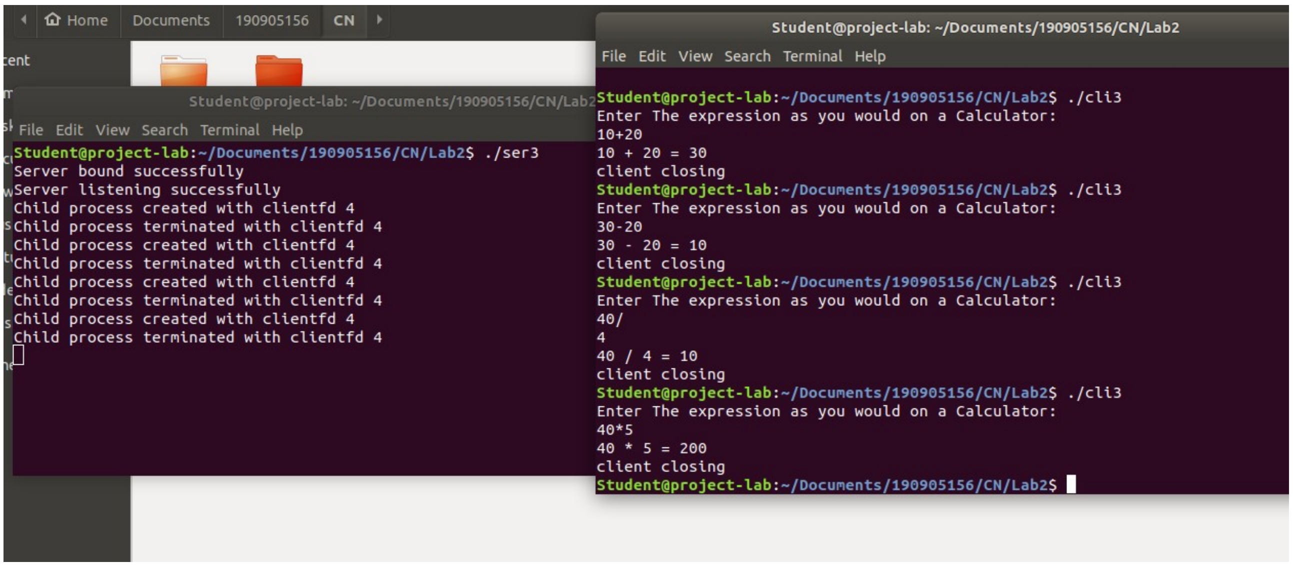
```
#include <unistd.h>
#define PORT 5000

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 =
htonl(INADDR_ANY);
        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 listne\n");
		exit(0);
	}
	else
		printf("Server listening successfully\n");

	servfunc(sockfd, server_address);
	close(sockfd);
}
```

client.c

```
#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)
{
        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 = htonl(INADDR_ANY);
        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);
}
```



```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./ser3
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
Child process created with clientfd 4
Child process terminated with clientfd 4
Child process created with clientfd 4
Child process terminated with clientfd 4
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli3
Enter The expression as you would on a Calculator:
10+20
10 + 20 = 30
client closing
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli3
Enter The expression as you would on a Calculator:
30-20
30 - 20 = 10
client closing
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli3
Enter The expression as you would on a Calculator:
40/
4
40 / 4 = 10
client closing
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli3
Enter The expression as you would on a Calculator:
40*5
40 * 5 = 200
client closing
Student@project-lab:~/Documents/190905156/CN/Lab2$
```

3. Implement simple TCP daytime server using fork.

client.c

```
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
#include <unistd.h>

int main()
{
   int sockfd;
   int len;
   struct sockaddr_in address;
   struct tm * timeinfo;
   int result;
   char *reply;
   int hour,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)
   {
      perror("oops: client2");
      exit(1);
   }

   printf(" Sending request to get the time\n");

   read(sockfd, &hour , 1);
   read(sockfd, &mins , 1);
```

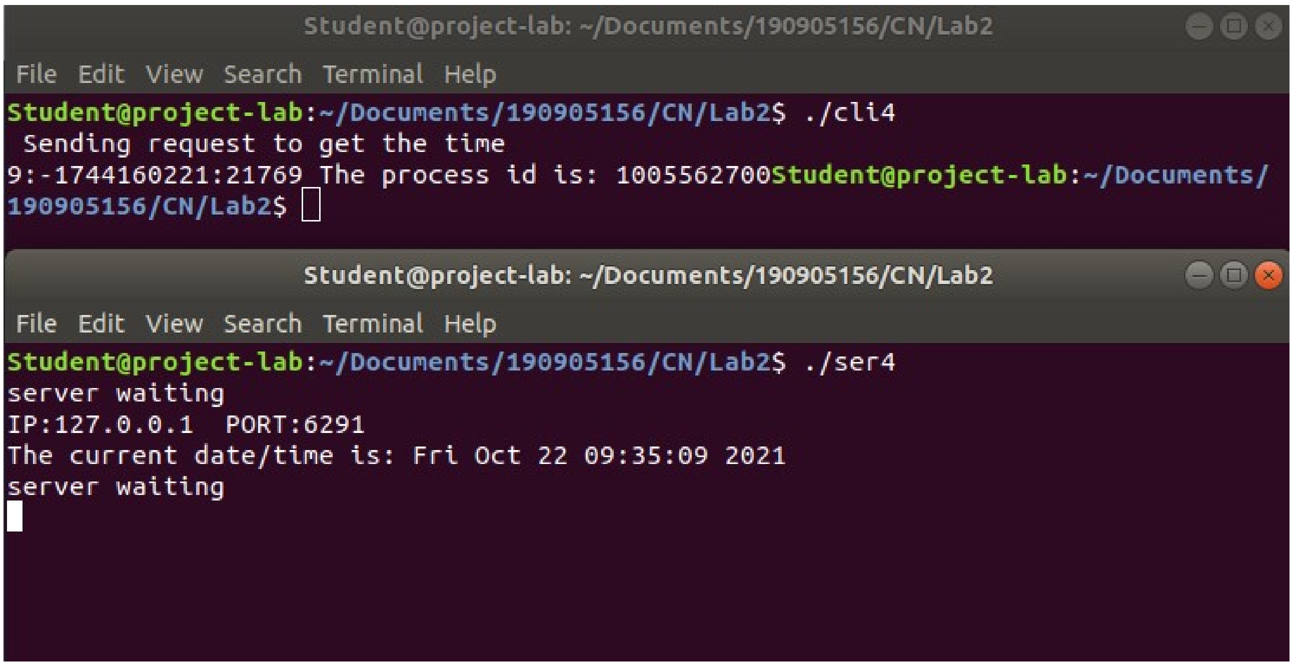
```
    read(sockfd, &sec , 1);
    read(sockfd, &pid , 1);
    printf("%d:%d:%d", hour, mins, sec);
    printf(" The process id is: %d",pid);

    close(sockfd);
    exit(0);

    return 0;
}
```

server.c

```
#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 current date/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);


    }


    return 0;
}
```



```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./cli4
 Sending request to get the time
9:-1744160221:21769 The process id is: 1005562700Student@project-lab:~/Documents/
190905156/CN/Lab2$
```

```
Student@project-lab:~/Documents/190905156/CN/Lab2$ ./ser4
server waiting
IP:127.0.0.1  PORT:6291
The current date/time is: Fri Oct 22 09:35:09 2021
server waiting
```
Student@project-lab: ~/Documents/190905156/CN/Lab2
File Edit View Search Terminal Help
Student@project-lab: ~/Documents/190905156/CN/Lab2
File Edit View Search Terminal Help