Name-KISHAN KUMAR
Enrol. No.-17114046
Batch-CS2
Sub- CSN-361 LAB

Q.1. Write a socket program in C to connect two nodes on a network to communicate with each other, where one socket listens on a particular port at an IP, while other socket reaches out to the other to form a connection.

client.c

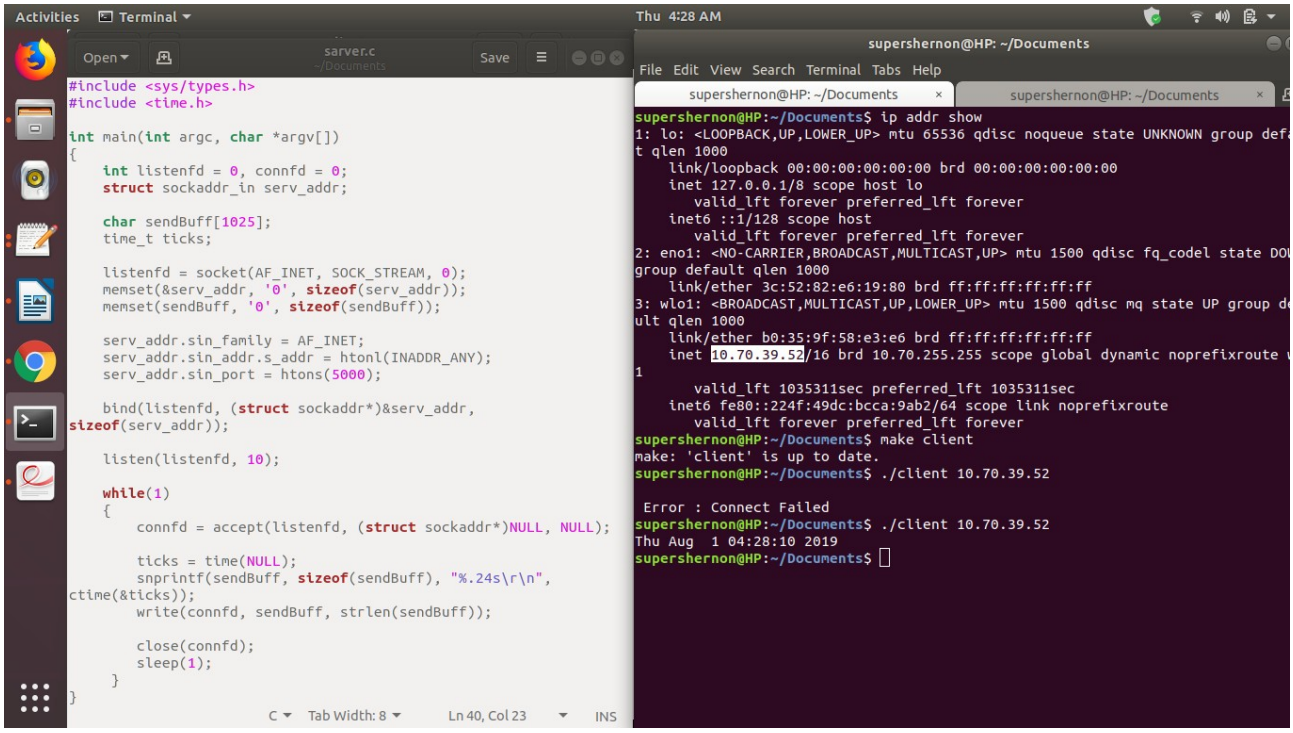
```c
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if(argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return 1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
    {
```

```
supershernon@HP:~/Documents$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defa
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOW
group default qlen 1000
    link/ether 3c:52:82:e6:19:80 brd ff:ff:ff:ff:ff:ff
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group de
ult qlen 1000
    link/ether b0:35:9f:58:e3:e6 brd ff:ff:ff:ff:ff:ff
    inet 10.70.39.52/16 brd 10.70.255.255 scope global dynamic noprefixroute w
1
       valid_lft 1035311sec preferred_lft 1035311sec
    inet6 fe80::224f:49dc:bcca:9ab2/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
supershernon@HP:~/Documents$ make client
make: 'client' is up to date.
supershernon@HP:~/Documents$ ./client 10.70.39.52

 Error : Connect Failed
supershernon@HP:~/Documents$
```

sarver.c

```c
#include <sys/types.h>
#include <time.h>

int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1025];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);

    bind(listenfd, (struct sockaddr*)&serv_addr,
sizeof(serv_addr));

    listen(listenfd, 10);

    while(1)
    {
        connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

        ticks = time(NULL);
        snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n",
ctime(&ticks));
        write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
    }
}
```

```
supershernon@HP:~/Documents$ make server
make: 'server' is up to date.
supershernon@HP:~/Documents$ ./server
```

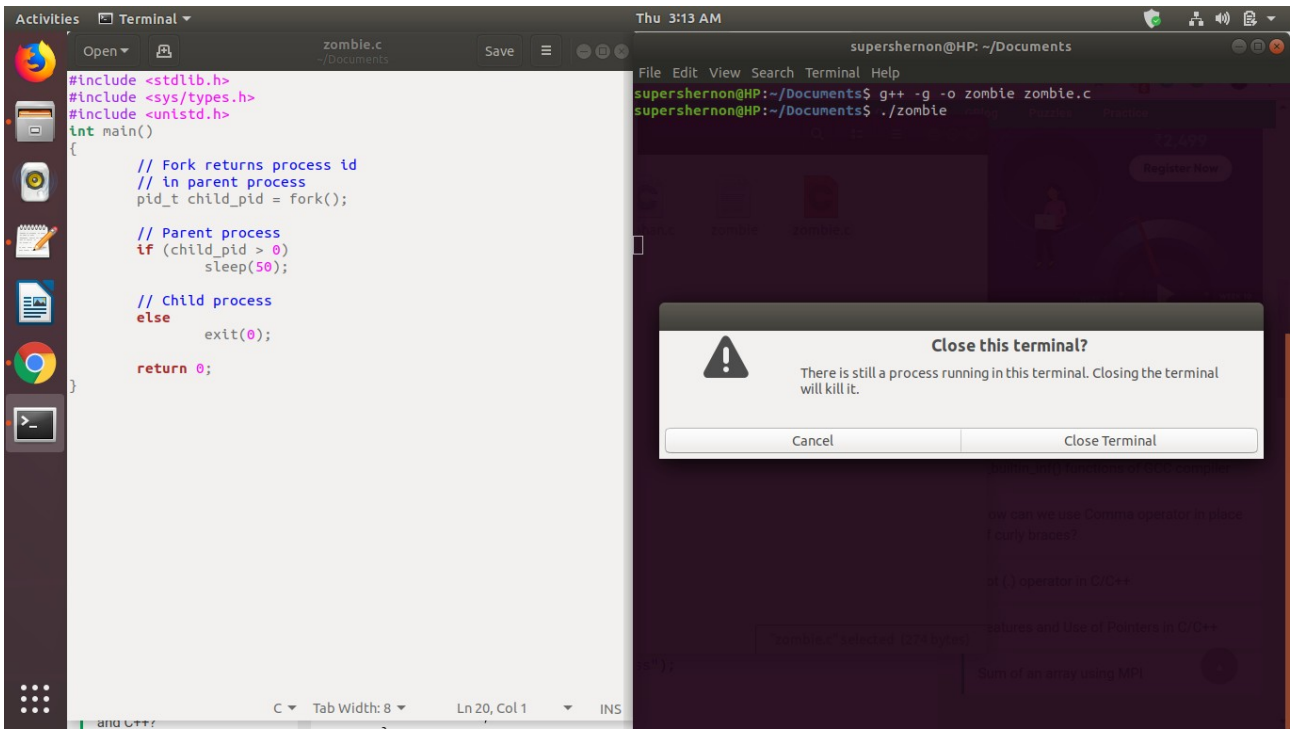


Algorithms used :
1. gethostbyname: to get the IP address of the host.
2. inet_addr: for proper conversion of the IP address returned.
3. socket: to create a socket of AF_INET address family.
4. getpid : system call of the process id.
5. in_cksum: code to calculate the checksum.
6. FD_ZERO: clear an fdset.
7. FD_SET: add a socket descriptor to the fdset.
8. select: select return values from different sockets without multithreading.
9. sendto: To send the data to the opened socket to the specified IP address.
10. recvfrom: To receive the data from the socket.
11. gettimeofday: To calculate the ping time.

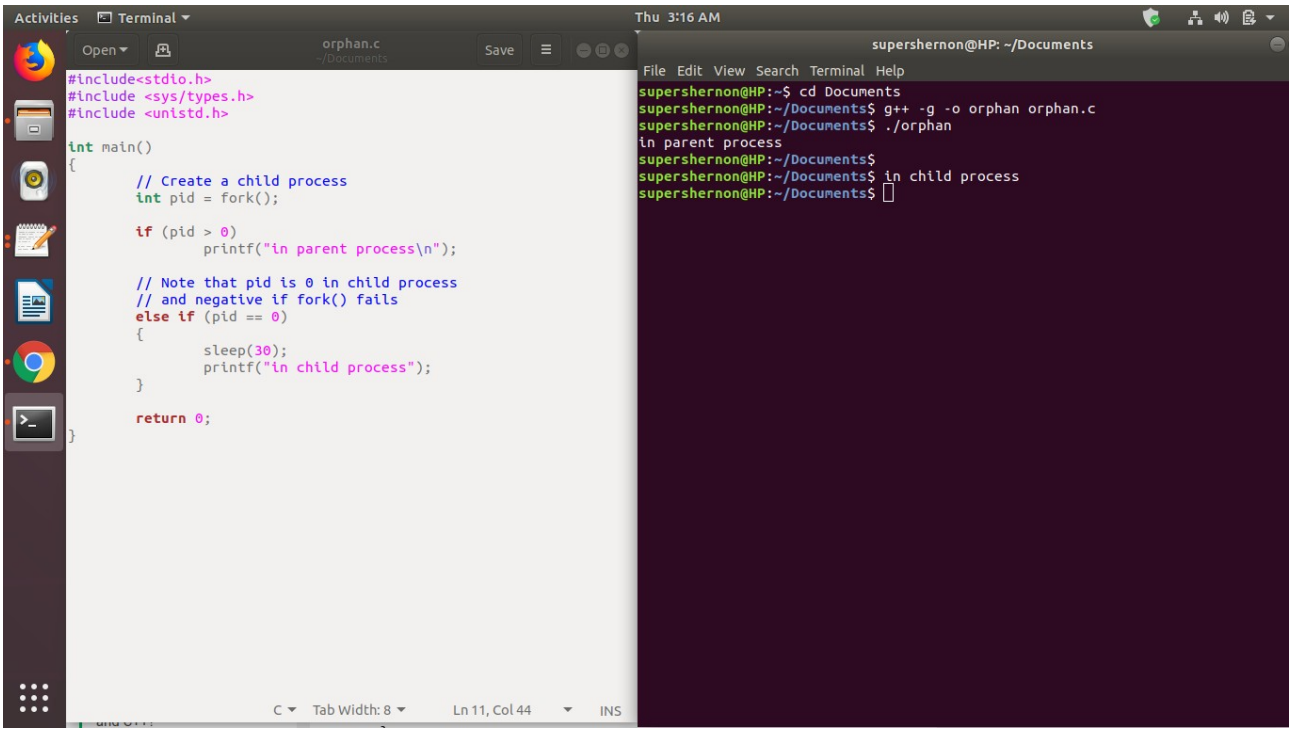
Data Structures used :
1. hostent: to store data about a specific host
2. sock_addr_in: to specify a transport address and port for the AF_INET address family.
3. ip: IP header.
4. icmp: icmp header.
5. timeval: checking interval for the socket.

Q.2. Write a C program to demonstrate both Zombie and Orphan process.

For Zombie process:



For Orphan process:



Algorithms used :
1. Busy waiting.

Data structures used:
1. Pid_t: C struct to store the process id.