1. Extend your echo Client Server message passing application to chat application.
   - Client and Server are able to send the message to each other until one of them quits or terminates.

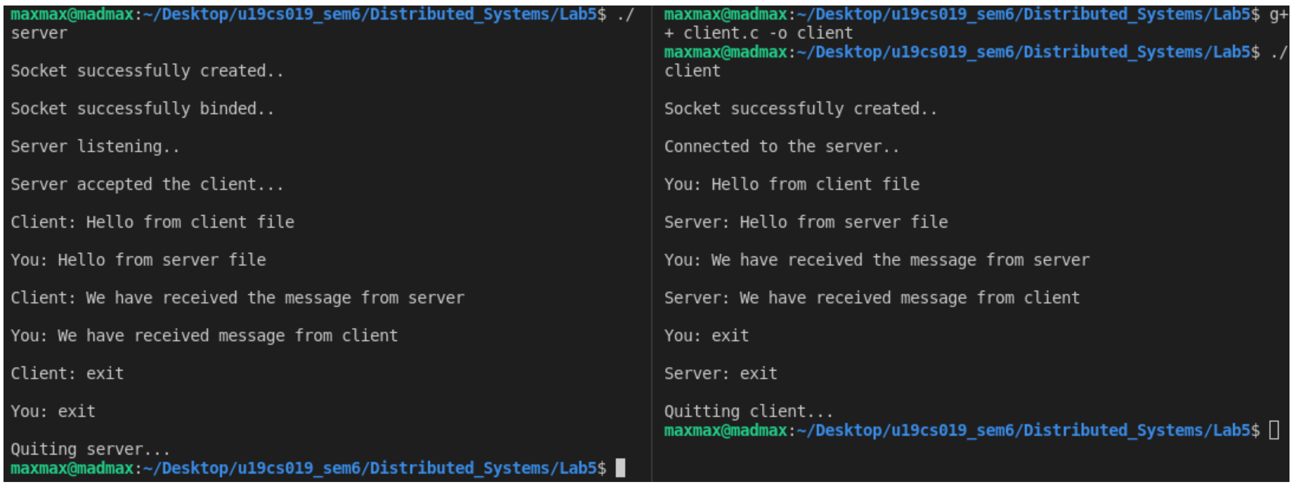
CODE => server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int connfd)
{
char buff[MAX];
int n;
while (1)
{
bzero(buff, MAX);
read(connfd, buff, sizeof(buff));
printf("\nClient: %s", buff);
printf("\nYou: ");
bzero(buff, MAX);
n = 0;
while ((buff[n++] = getchar()) != '\n')
;
write(connfd, buff, sizeof(buff));
if (strncmp("exit", buff, 4) == 0)
{
printf("\nQuiting server...\n");
break;
}
}
}
int main()
{
int sockfd, connfd, len;
struct sockaddr_in servaddr, cli;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1)
{
printf("\nSocket creation failed...\n");
exit(0);
}
else
printf("\nSocket successfully 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 ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
{
printf("\nSocket bind failed...\n");
exit(0);
}
else
printf("\nSocket successfully binded..\n");
if ((listen(sockfd, 5)) != 0)
{
printf("\nListen failed...\n");
exit(0);
}
else
printf("\nServer listening..\n");
len = sizeof(cli);
connfd = accept(sockfd, (SA *)&cli,(socklen_t*)&len);
if (connfd < 0)
{
printf("\nServer didn't accept the client...\n");
exit(0);
}
else
printf("\nServer accepted the client...\n");
func(connfd);
close(sockfd);
}
```

CODE => client.c

```
#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
char buff[MAX];
int n;
while (1)
{
bzero(buff, sizeof(buff));
printf("\nYou: ");
n = 0;
while ((buff[n++] = getchar()) != '\n')
;
write(sockfd, buff, sizeof(buff));
bzero(buff, sizeof(buff));
read(sockfd, buff, sizeof(buff));
printf("\nServer: %s", buff);
if ((strncmp(buff, "exit", 4)) == 0)
{
printf("\nQuitting client...\n");
break;
}
}
}
int main()
{
int sockfd, connfd;
struct sockaddr_in servaddr, cli;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1)
{
printf("\nSocket creation failed...\n");
exit(0);
}
else
printf("\nSocket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(PORT);
if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
{
printf("\nServer connection failed...\n");
exit(0);
}
else
printf("\nConnected to the server..\n");
func(sockfd);
close(sockfd);
}
```



```
maxmax@madmax:~/Desktop/u19cs019_sem6/Distributed_Systems/Lab5$ ./
server

Socket successfully created..

Socket successfully binded..

Server listening..

Server accepted the client...

Client: Hello from client file

You: Hello from server file

Client: We have received the message from server

You: We have received message from client

Client: exit

You: exit

Quiting server...
maxmax@madmax:~/Desktop/u19cs019_sem6/Distributed_Systems/Lab5$
```

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Distributed_Systems/Lab5$ g+
+ client.c -o client
maxmax@madmax:~/Desktop/u19cs019_sem6/Distributed_Systems/Lab5$ ./
client

Socket successfully created..

Connected to the server..

You: Hello from client file

Server: Hello from server file

You: We have received the message from server

Server: We have received message from client

You: exit

Server: exit

Quitting client...
maxmax@madmax:~/Desktop/u19cs019_sem6/Distributed_Systems/Lab5$
```

2. Using the Client-Server communication mechanism get the load status of other nodes in your network (identify the states of other nodes in the system – Overload, Moderate, Lightly).

- Implement the Client-Server model. Run the client and server instance on same machine and pass the message from client to server or server to client
- Get the CPU load of the client or server and state that either it is under loaded or overloaded.

The client server communication mechanism has the limitation that it only handles one connection at a time and then terminates. A real-world server should run indefinitely and should have the capability of handling a number of simultaneous connections, each in its own process.

CODE => server.c

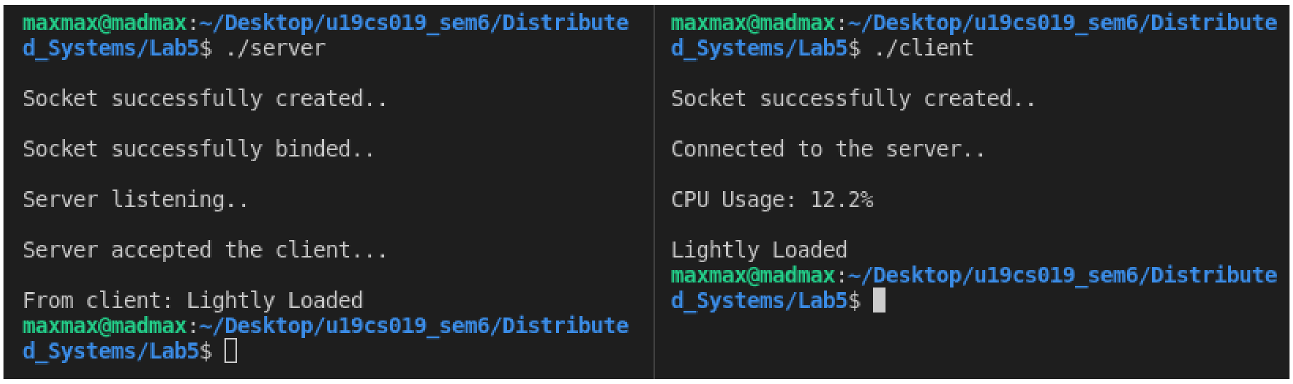
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int connfd)
{
char buff[MAX];
int n;
bzero(buff, MAX);
read(connfd, buff, sizeof(buff));
printf("\nFrom client: %s", buff);
}
int main()
{
int sockfd, connfd, len;
struct sockaddr_in servaddr, cli;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1)
{
printf("\nSocket creation failed...\n");
exit(0);
}
else
printf("\nSocket successfully 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 ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
{
printf("\nSocket bind failed...\n");
exit(0);
}
else
printf("\nSocket successfully binded..\n");
if ((listen(sockfd, 5)) != 0)
{
printf("\nListen failed...\n");
exit(0);}
else
printf("\nServer listening..\n");
len = sizeof(cli);
connfd = accept(sockfd, (SA *)&cli, (socklen_t*)&len);
if (connfd < 0)
{
printf("\nServer didn't accept the client...\n");
exit(0);
}
else
printf("\nServer accepted the client...\n");
func(connfd);
close(sockfd);
}
```

CODE => client.c

```
#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
FILE *fp;
fp = popen("./cpu.sh", "r");
char buff[1024];
if (fp)
{
while (1)
{
char *line;
line = fgets(buff, sizeof(buff), fp);
if (!line)
{
break;
}
printf("\n%s", line);
}
pclose(fp);
}
write(sockfd, buff, sizeof(buff));
}
```

```c
int main()
{
int sockfd, connfd;
struct sockaddr_in servaddr, cli;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1)
{
printf("\nSocket creation failed...\n");
exit(0);
}
else
printf("\nSocket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(PORT);
if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
{
printf("\nServer connection failed...\n");
exit(0);
}
else
printf("\nConnected to the server..\n");
func(sockfd);
close(sockfd);
}
```



```
maxmax@madmax:~/Desktop/u19cs019_sem6/Distribute
d_Systems/Lab5$ ./server

Socket successfully created..

Socket successfully binded..

Server listening..

Server accepted the client...

From client: Lightly Loaded
maxmax@madmax:~/Desktop/u19cs019_sem6/Distribute
d_Systems/Lab5$
```

```
maxmax@madmax:~/Desktop/u19cs019_sem6/Distribute
d_Systems/Lab5$ ./client

Socket successfully created..

Connected to the server..

CPU Usage: 12.2%

Lightly Loaded
maxmax@madmax:~/Desktop/u19cs019_sem6/Distribute
d_Systems/Lab5$
```