











Server

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/socket.h> //socket

#include <arpa/inet.h> //inet\_addr

int clientcount = 0;

char server\_message[2000], client\_message[2000],client\_message2[2000];

void \*handleclients(void \*socket\_desc\_ptr)

{

int socket\_desc;

socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0);

int client\_sock = \*(int \*)socket\_desc\_ptr;

recv(client\_sock, client\_message, sizeof(client\_message), 0);

int j=0;

int index=0;

char servermsg[]="Processed by Server";

FILE\* f;

f=fopen("output.txt","w");

for(int i=0;i<strlen(client\_message);i++)

{

if(client\_message[i]>='A' && client\_message[i]<='Z')

{

client\_message2[index]=client\_message[i]+32;

index++;

}

else if(client\_message[i]>='a' && client\_message[i]<='z')

{

client\_message2[index]=client\_message[i]-32;

index++;

}else if(client\_message[i]=='\n')

{

strcat(client\_message2,servermsg);

}

else

{

client\_message2[index]=client\_message[i];

index++;

}

}

strcpy(server\_message, client\_message2);

fprintf(f, "%s", client\_message2);

fflush(f);

if (send(client\_sock, server\_message, strlen(server\_message), 0) < 0)

{

printf("Send Failed. Error!!!!!\n");

}

memset(server\_message, '\0', sizeof(server\_message));

memset(client\_message, '\0', sizeof(client\_message));

memset(client\_message2, '\0', sizeof(client\_message2));

clientcount--;

}

int main(void)

{

int socket\_desc, client\_sock, client\_size;

struct sockaddr\_in server\_addr, client\_addr;

pthread\_t clients[5];

memset(server\_message, '\0', sizeof(server\_message));

memset(client\_message, '\0', sizeof(client\_message));

// Creating Socket

socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0);

if (socket\_desc < 0)

{

printf("Could Not Create Socket. Error!!!!!\n");

return -1;

}

printf("Socket Created\n");

// Binding IP and Port to socket

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(2000);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

// BINDING FUNCTION

if (bind(socket\_desc, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

{

printf("Bind Failed. Error!!!!!\n");

return -1;

}

printf("Bind Done\n");

// Put the socket into Listening State

if (listen(socket\_desc, 1) < 0)

{

printf("Listening Failed. Error!!!!!\n");

return -1;

}

while (1)

{

if (clientcount <= 5)

{

printf("\nListening for Incoming Connections.....\n");

// Accept the incoming Connections

client\_sock = accept(socket\_desc, (struct sockaddr \*)&client\_addr, &client\_size);

printf("Client Connected with IP: %s and Port No: %i\n", inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));

pthread\_create(&clients[clientcount], NULL, &handleclients, &client\_sock);

clientcount++;

}

else

{

printf("\nPlease wait for other users to leave.");

sleep(20);

}

}

// Closing the Socket

close(client\_sock);

close(socket\_desc);

return 0;

}

Client

#include <stdio.h>

#include <string.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h> //socket

#include <arpa/inet.h> //inet\_addr

int main(void)

{

int socket\_desc;

struct sockaddr\_in server\_addr;

char server\_message[2000], client\_message[2000];

char filecontent[500];

socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0);

if (socket\_desc < 0)

{

printf("Could Not Create Socket. Error!!!!!\n");

return -1;

}

printf("Socket Created\n");

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(2000);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if (connect(socket\_desc, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0)

{

printf("Connection Failed. Error!!!!!");

return -1;

}

printf("Connected\n");

FILE \*f;

char temp[100];

f = fopen("input.txt", "r");

while (fgets(temp, sizeof(temp), f) != NULL)

{

strcat(filecontent, temp);

}

strcpy(client\_message,filecontent);

// Send the message to Server

if (send(socket\_desc, client\_message, strlen(client\_message), 0) < 0)

{

printf("Send Failed. Error!!!!\n");

return -1;

}

// Receive the message back from the server

if (recv(socket\_desc, server\_message, sizeof(server\_message), 0) < 0)

{

printf("Receive Failed. Error!!!!!\n");

return -1;

}

printf("Server Message: %s\n", server\_message);

memset(server\_message, '\0', sizeof(server\_message));

memset(client\_message, '\0', sizeof(client\_message));

// Closing the Socket

close(socket\_desc);

return 0;

}