**Computer Networks Lab 2**

**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 Side program file called “q1server.c”:**

//Server program

#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 Side program file called “q1client.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);

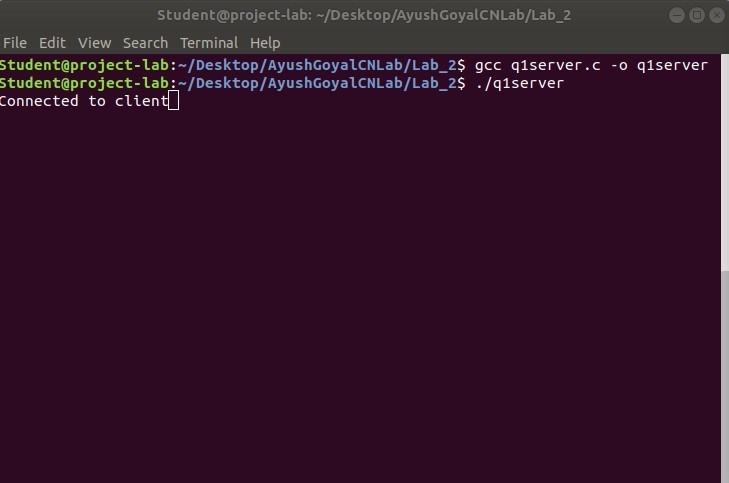
getchar();

}

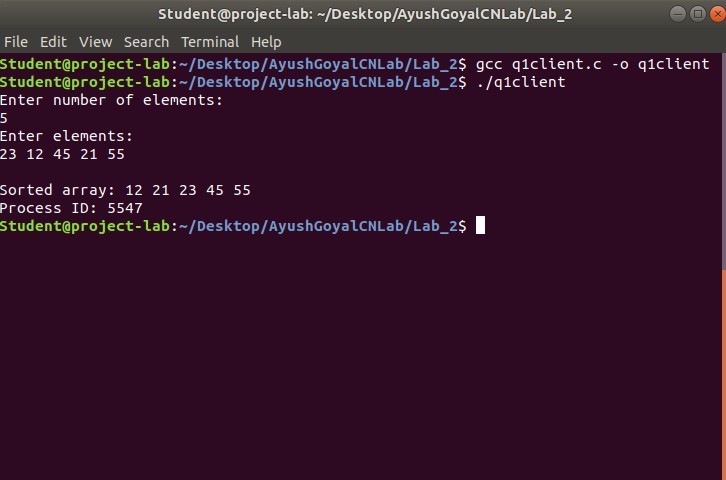
**We first compile and execute the server side program in a terminal, after which we compile and run the client side program. The desired output is as shown below:**

**Output:**

**Server side terminal:**



**Client side terminal:**



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

**Server Side program file called “q2server.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)

{

//in child process

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); // parent process

}

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 Side program file called “q2client.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)

{

printf("I am a client and my name is Ayush Goyal.\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 = htonl(INADDR\_ANY);

len = sizeof(server\_address);

result = connect(sockfd, (struct sockaddr \*)&server\_address, len); if(result == -1)

{

printf("Connection Error Occured!\n"); exit(0);

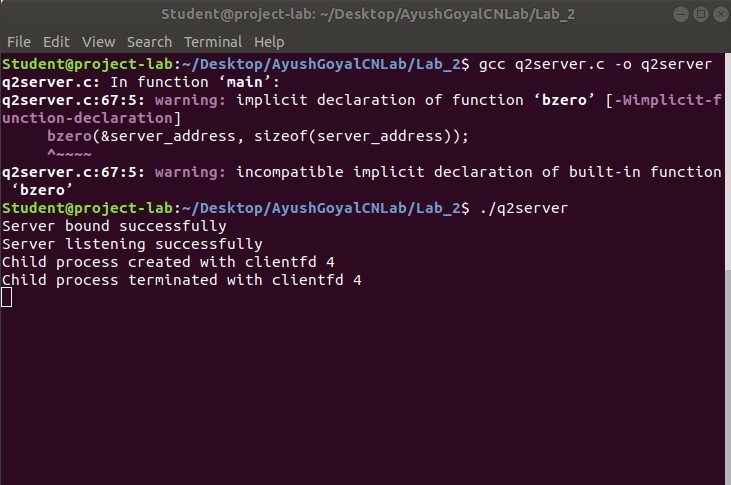
}

clifunc(sockfd);

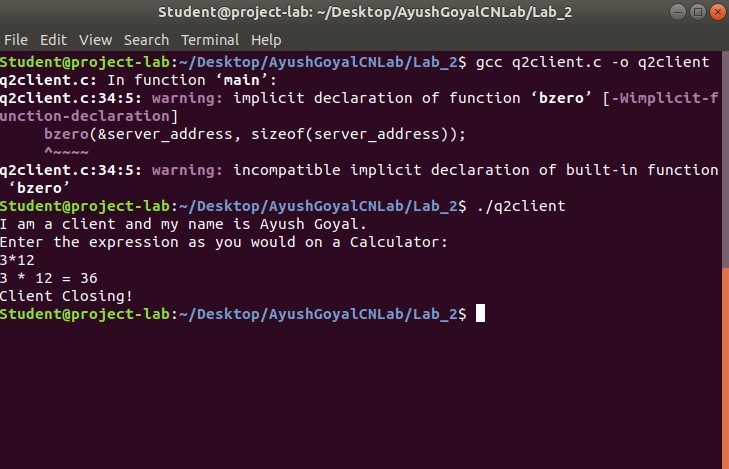
close(sockfd); }

**We first compile and execute the server side program in a terminal, after which we compile and run the client side program. The desired output is as shown below: Output:**

**Server side terminal:**



**Client side terminal:**



**3. Implement simple TCP daytime server using fork.**

**Server Side program file called “q3server.c”:**

#include <arpa/inet.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <time.h>

#define MAX 80

#define PORT 5000

#define SA struct sockaddr

void servfunc(int sockfd, struct sockaddr\_in \*cli){ char buff[MAX];

int n; for(;;){

bzero(buff, sizeof(buff)); n = recv(sockfd, buff, sizeof(buff), 0); buff[n] = '\0'; if (strcmp(buff, "QUIT") == 0){ printf("Server Exit...\n");

break;

} else if (strcmp(buff, "time") == 0){ time\_t rawtime; struct tm \*info; time(&rawtime); info = localtime(&rawtime); char \*str = asctime(info); ssize\_t size\_str = strlen(str); n = send(sockfd, str, size\_str, 0);

if (n == -1){

printf("Error in sending message. Try Again!\n"); continue;

} } else{ char str[] = "ERROR"; if(send(sockfd, str, sizeof(str), 0) == -1){ printf("Error in sending message. Try Again!\n"); continue;

}

}

}

}

int main(){ int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli; sockfd = socket(AF\_INET, SOCK\_STREAM, 0); if(sockfd == -1){

printf("Socket creation failed...\n");

exit(0); } else

printf("Socket 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("socket bind failed...\n");

exit(0); } else

printf("Socket successfully binded..\n"); if ((listen(sockfd, 5)) != 0)

{

printf("Listen failed...\n");

exit(0); } else

printf("Server listening..\n"); len = sizeof(cli);

connfd = accept(sockfd, (SA \*)&cli, &len);

if (connfd < 0)

{

printf("server acccept failed...\n");

exit(0); } else printf("server acccept the client...\n"); servfunc(connfd, (struct sockaddr\_in \*)&cli); close(sockfd);

}

**Client Side program file called “q3client.c”:**

#include <arpa/inet.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 5000

#define SA struct sockaddr

void clifunc(int sockfd, struct sockaddr\_in \*cli)

{

char buff[MAX];

int n;

char \*client\_ip = inet\_ntoa(cli->sin\_addr);

int client\_port = (int)ntohs(cli->sin\_port); for (;;)

{

printf("I am a client. My name is Ayush Goyal.\n"); bzero(buff, sizeof(buff)); printf("Enter the String : "); n = 0;

scanf("%[^\n]%\*c", buff);

if (send(sockfd, buff, sizeof(buff), 0) == -1)

{

printf("Error in sending message. Try Again!\n"); continue;

}

if (strcmp(buff, "QUIT") == 0)

{

printf("Server Closed, client exiting!\n"); break;

}

bzero(buff, sizeof(buff));

n = recv(sockfd, buff, sizeof(buff), 0); buff[n] = '\0';

if (strcmp(buff, "ERROR") == 0)

{

printf("Wrong time command. Enter time\n"); continue;

} else

printf("Recieved time from Server IP:%s and Port:%d is %s\n", client\_ip, client\_port, buff); }

}

int main(){ int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

sockfd = socket(AF\_INET, SOCK\_STREAM, 0); if (sockfd == -1){

printf("socket creation failed...\n"); exit(0);

} else

printf("Socket 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(connect(sockfd, (SA \*)&servaddr, sizeof(servaddr)) != 0){ printf("connection with the server failed...\n"); exit(0);

} else

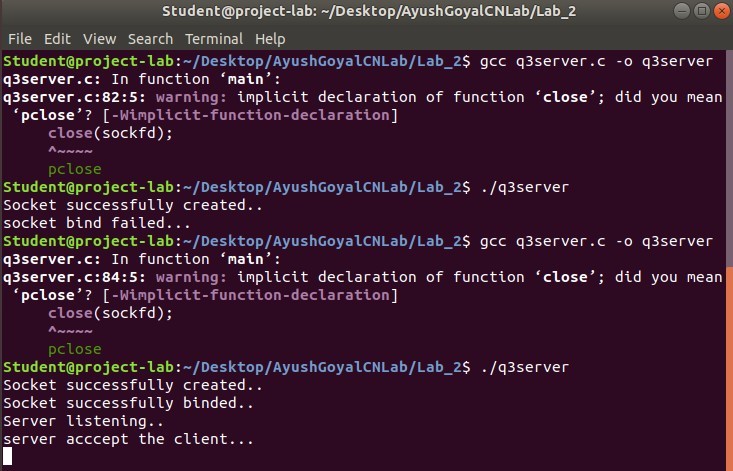
printf("connected to the server..\n");

clifunc(sockfd, &servaddr); close(sockfd); }

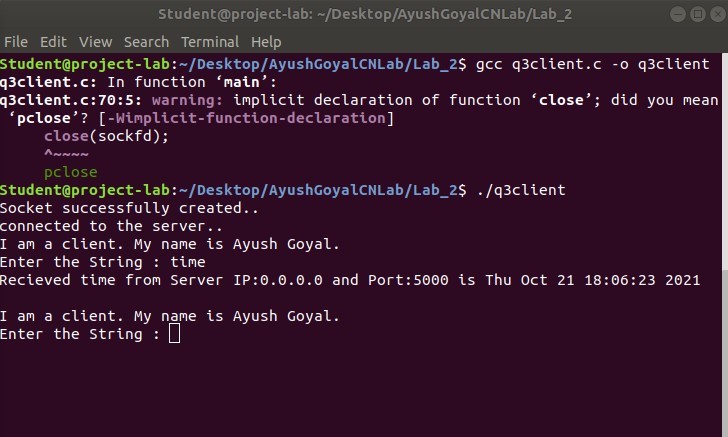
**We first compile and execute the server side program in a terminal, after which we compile and run the client side program. The desired output is as shown below:**

**Output:**

**Server side terminal:**



**Client side terminal:**



**THE END**