Department of Computer Science & Engineering



Programme Name: B.Tech Course Code: CS17201

Semester: VII Branch: Computer Science & Engg. Course Name: Distributed Systems (Lab)

**Lab Assignment 7**

Name-Sugandh Mishra

Reg-20204211

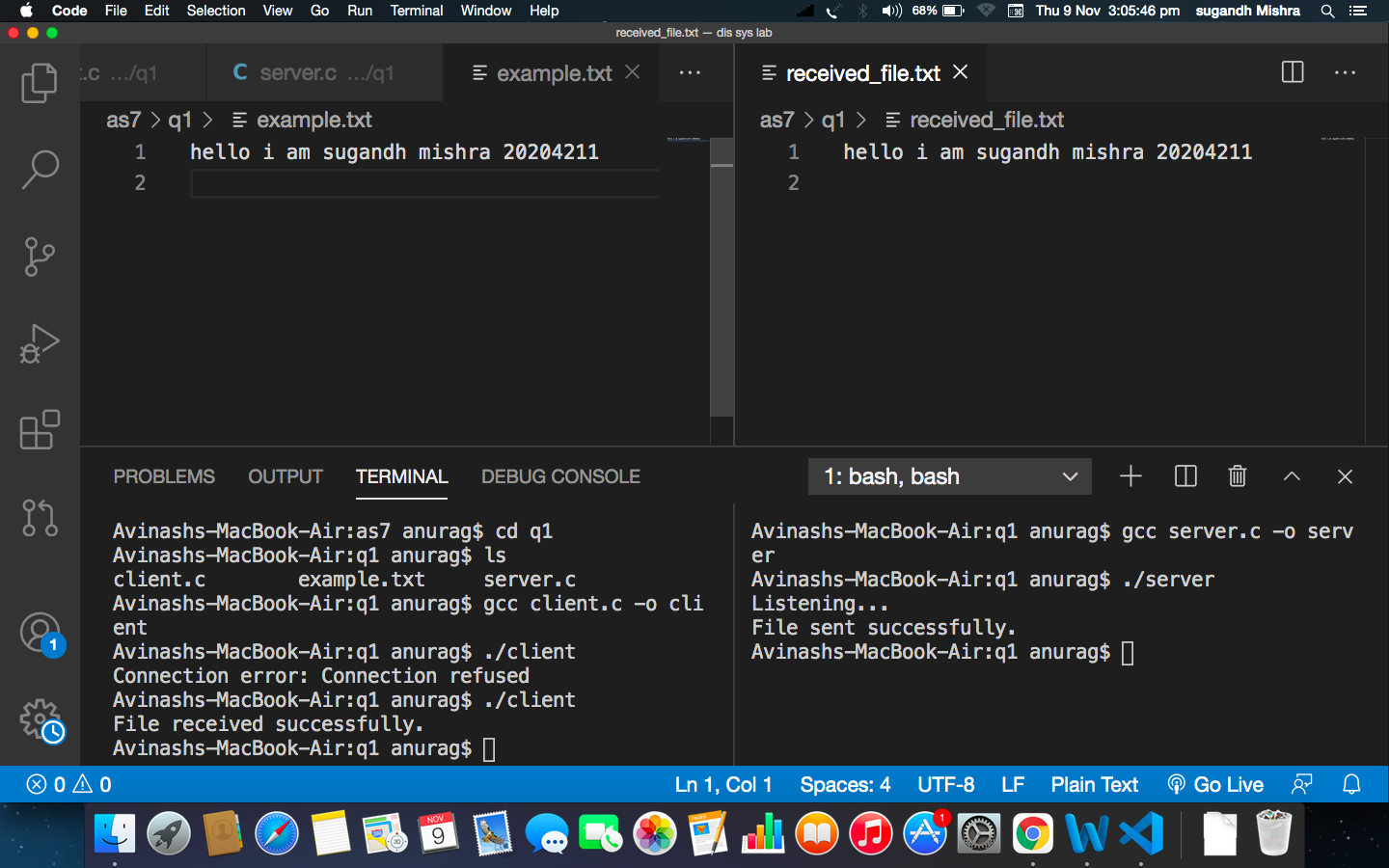
Sec- CSE C



**National**

**211004**

|  |  |
| --- | --- |
| **Lab #** | **Name of Experiments** |
| **7** | 1. Implement RPC mechanism for a file transfer across a network in ‘C’.   Server.c----  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <netinet/in.h>  #include <arpa/inet.h>  #define PORT 12345  void transfer\_file(int client\_socket) {  FILE \*file = fopen("example.txt", "rb");  if (file == NULL) {  perror("File open error");  exit(1);  }  char buffer[1024];  size\_t bytesRead;  while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {  send(client\_socket, buffer, bytesRead, 0);  }  fclose(file);  }  int main() {  int server\_socket, client\_socket;  struct sockaddr\_in server\_addr, client\_addr;  socklen\_t addr\_size;  server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_socket < 0) {  perror("Socket creation error");  exit(1);  }  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_port = htons(PORT);  server\_addr.sin\_addr.s\_addr = INADDR\_ANY;  if (bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {  perror("Binding error");  exit(1);  }  if (listen(server\_socket, 10) == 0) {  printf("Listening...\n");  } else {  perror("Listening error");  exit(1);  }  addr\_size = sizeof(client\_addr);  client\_socket = accept(server\_socket, (struct sockaddr\*)&client\_addr, &addr\_size);  transfer\_file(client\_socket);  printf("File sent successfully.\n");  close(client\_socket);  close(server\_socket);  return 0;  }  client.c----  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <netinet/in.h>  #include <arpa/inet.h>  #define SERVER\_IP "127.0.0.1" // Change this to the server's IP address  #define PORT 12345  void receive\_file(int server\_socket) {  FILE \*file = fopen("received\_file.txt", "wb");  if (file == NULL) {  perror("File create error");  exit(1);  }  char buffer[1024];  int bytesRead;  while ((bytesRead = recv(server\_socket, buffer, sizeof(buffer), 0)) > 0) {  fwrite(buffer, 1, bytesRead, file);  }  fclose(file);  }  int main() {  int client\_socket;  struct sockaddr\_in server\_addr;  client\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  if (client\_socket < 0) {  perror("Socket creation error");  exit(1);  }  server\_addr.sin\_family = AF\_INET;  server\_addr.sin\_port = htons(PORT);  server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);  if (connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {  perror("Connection error");  exit(1);  }  receive\_file(client\_socket);  printf("File received successfully.\n");  close(client\_socket);  return 0;  } |



ii) Implement ‘Java RMI’ mechanism for accessing methods of remote systems.

Server.c---

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

int add(int a, int b) {

return a + b;

}

int main() {

int server\_socket, client\_socket;

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

socklen\_t addr\_size;

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

if (server\_socket < 0) {

perror("Socket creation error");

exit(1);

}

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

server\_addr.sin\_port = htons(12345);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

if (bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Binding error");

exit(1);

}

if (listen(server\_socket, 10) == 0) {

printf("Listening...\n");

} else {

perror("Listening error");

exit(1);

}

addr\_size = sizeof(client\_addr);

client\_socket = accept(server\_socket, (struct sockaddr\*)&client\_addr, &addr\_size);

int a, b, result;

recv(client\_socket, &a, sizeof(a), 0);

recv(client\_socket, &b, sizeof(b), 0);

result = add(a, b);

send(client\_socket, &result, sizeof(result), 0);

close(client\_socket);

close(server\_socket);

return 0;

}

client.c-----

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

int main() {

int client\_socket;

struct sockaddr\_in server\_addr;

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

if (client\_socket < 0) {

perror("Socket creation error");

exit(1);

}

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

server\_addr.sin\_port = htons(12345);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

if (connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Connection error");

exit(1);

}

int a = 5, b = 3, result;

send(client\_socket, &a, sizeof(a), 0);

send(client\_socket, &b, sizeof(b), 0);

recv(client\_socket, &result, sizeof(result), 0);

printf("Result: %d\n", result);

close(client\_socket);

return 0;

}

