COMPUTER NETWORKS

LAB ASSIGNMENT-06

REDDIPALLI SAI CHARISH

CS22B1095

SERVER CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 8080
#define BUFFER_SIZE 1024
#define ACK_MSG "ACK"

void send_file(FILE *fp, int sockfd);
void receive_file(int sockfd, const char *filename);
int file_exists(const char *filename);
void create_file(const char *filename);

int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   int addrlen = sizeof(address);
   char buffer[BUFFER_SIZE] = {0};
   char filename[BUFFER_SIZE] = {0};

   // Create socket file descriptor
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("Socket failed");
      exit(EXIT_FAILURE);
   }
```

```
// Bind the socket to the network address and port
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(PORT);

if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    perror("Bind failed");
    exit(EXIT_FAILURE);
}

// Listen for connections
if (listen(server_fd, 3) < 0) {
    perror("Listen failed");
    exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);

if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen)) < 0) {
    perror("Accept failed");
    exit(EXIT_FAILURE);
}
//printf("Connection accepted from %s:%d\n", inet_ntoa(address.sin_addr), ntohs(address.sin_port));

// Step 1: Server creates a file
printf("Enter the name of the file to create on the server: ");
scanf("%s", filename);
create_file(filename);

// Step 2: Server requests a file from the client
printf("Enter the name of the file to request from the client: ");
scanf("%s", filename);
send(new_socket, filename, strlen(filename), 0);

// Step 3: Receive the file or ACK message from the client
recv(new_socket, buffer, BUFFER_SIZE, 0);
if (strcmp(buffer, ACK_MSG) == 0) {
    printf("Client responded that the file '%s' does not exist.\n", filename);
```

```
    } else {
        // If we receive data, it's the file content
        FILE *fp = fopen(filename, "w");
        if (fp == NULL) {
            perror("Could not open file to write");
            exit(EXIT_FAILURE);
        }
        fprintf(fp, "%s", buffer);
        fclose(fp);
        printf("File '%s' received successfully from client.\n", filename);
        // Display contents only after successful transfer
        receive_file(new_socket, filename);
    }

    // Step 4: Handle the client's request for a file
    memset(buffer, 0, BUFFER_SIZE);
    read(new_socket, buffer, BUFFER_SIZE);
    printf("Client requested file: %s\n", buffer);

    if (file_exists(buffer)) {
        FILE *fp = fopen(buffer, "r");
        send_file(fp, new_socket);
        fclose(fp);
    } else {
        send(new_socket, ACK_MSG, strlen(ACK_MSG), 0);
        printf("File '%s' does not exist. Sent acknowledgment to client.\n", buffer);
    }

    close(new_socket);
    close(server_fd);
    return 0;
}

void send_file(FILE *fp, int sockfd) {
    char data[BUFFER_SIZE] = {0};

    while (fgets(data, BUFFER_SIZE, fp) != NULL) {
        if (send(sockfd, data, sizeof(data), 0) == -1) {
```

```
            perror("Failed to send file.");
            exit(EXIT_FAILURE);
        }
        memset(data, 0, BUFFER_SIZE);
    }
}

int file_exists(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file) {
        fclose(file);
        return 1;
    }
    return 0;
}

void create_file(const char *filename) {
    FILE *fp = fopen(filename, "w");
    char buffer[BUFFER_SIZE];

    if (fp == NULL) {
        perror("File creation failed");
        return;
    }

    printf("Enter contents for the file '%s':\n", filename);
    getchar();  // Clear newline from previous input
    fgets(buffer, BUFFER_SIZE, stdin);
    fprintf(fp, "%s", buffer);

    fclose(fp);
    printf("File '%s' created successfully.\n", filename);
}

void receive_file(int sockfd, const char *filename) {
    char buffer[BUFFER_SIZE] = {0};
    FILE *fp = fopen(filename, "r");
```

```
    if (fp == NULL) {
        perror("Could not open file to display");
        return;
    }

    printf("\n--- File Contents of '%s' ---\n", filename);
    while (fgets(buffer, BUFFER_SIZE, fp) != NULL) {
        printf("%s", buffer);
    }
    printf("--- End of File ---\n");

    fclose(fp);
}
```

Client Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 8080
#define BUFFER_SIZE 1024
#define ACK_MSG "ACK"

void send_file(FILE *fp, int sockfd);
void receive_file(int sockfd, const char *filename);
int file_exists(const char *filename);
void create_file(const char *filename);

int main() {
    int sockfd;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};
    char filename[BUFFER_SIZE] = {0};

    // Create socket
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        return -1;
    }

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

    // Convert IPv4 and IPv6 addresses from text to binary form
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        printf("Invalid address or address not supported\n");
        return -1;
    }
```

```
// Connect to the server
if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    perror("Connection failed");
    return -1;
}
printf("Connected to server at %s:%d\n","127.0.0.1", PORT);
// Step 1: Client creates a file
printf("Enter the name of the file to create on the client: ");
scanf("%s", filename);
create_file(filename);

// Step 2: Handle the server's request for a file
recv(sockfd, buffer, BUFFER_SIZE, 0);
printf("Server requested file: %s\n", buffer);

if (file_exists(buffer)) {
    FILE *fp = fopen(buffer, "r");
    send_file(fp, sockfd);
    fclose(fp);
} else {
    // Send acknowledgment if the file doesn't exist
    send(sockfd, ACK_MSG, strlen(ACK_MSG), 0);
    printf("File '%s' does not exist. Sent acknowledgment to server.\n", buffer);
}

// Step 3: Client requests a file from the server
printf("Enter the name of the file to request from the server: ");
scanf("%s", filename);
send(sockfd, filename, strlen(filename), 0);

// Step 4: Receive the file or ACK message
recv(sockfd, buffer, BUFFER_SIZE, 0);
if (strcmp(buffer, ACK_MSG) == 0) {
    printf("Server responded that the file '%s' does not exist.\n", filename);
} else {
    FILE *fp = fopen(filename, "w");
    fprintf(fp, "%s", buffer);
```

```
        fclose(fp);
        printf("File '%s' received successfully.\n", filename);
        receive_file(sockfd, filename);
    }

    close(sockfd);
    return 0;
}

void send_file(FILE *fp, int sockfd) {
    char data[BUFFER_SIZE] = {0};

    while (fgets(data, BUFFER_SIZE, fp) != NULL) {
        if (send(sockfd, data, sizeof(data), 0) == -1) {
            perror("Failed to send file.");
            exit(EXIT_FAILURE);
        }
        memset(data, 0, BUFFER_SIZE);
    }
}

int file_exists(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file) {
        fclose(file);
        return 1;
    }
    return 0;
}

void create_file(const char *filename) {
    FILE *fp = fopen(filename, "w");
    char buffer[BUFFER_SIZE];

    if (fp == NULL) {
        perror("File creation failed");
        return;
    }
```

```
    printf("Enter contents for the file '%s':\n", filename);
    getchar();  // Clear newline from previous input
    fgets(buffer, BUFFER_SIZE, stdin);
    fprintf(fp, "%s", buffer);

    fclose(fp);
    printf("File '%s' created successfully.\n", filename);
}

void receive_file(int sockfd, const char *filename) {
    char buffer[BUFFER_SIZE] = {0};
    FILE *fp = fopen(filename, "r");

    if (fp == NULL) {
        perror("Could not open file to display");
        return;
    }

    printf("\n--- File Contents of '%s' ---\n", filename);
    while (fgets(buffer, BUFFER_SIZE, fp) != NULL) {
        printf("%s", buffer);
    }
    printf("--- End of File ---\n");

    fclose(fp);
}
```

```
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 06$ gcc server_file_tcp.c
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 06$ ./a.out
Server listening on port 8080...
Connection accepted from 127.0.0.1:8080
Enter the name of the file to create on the server: m.txt
Enter contents for the file 'm.txt':
hello pranay!!
File 'm.txt' created successfully.
Enter the name of the file to request from the client: c.txt
File 'c.txt' received successfully from client.

--- File Contents of 'c.txt' ---
hello charish!!
--- End of File ---
Client requested file: m.txt
```

```
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 06$ gcc client_file_tcp.c
charish@LAPTOP-GFCS9LJ9:~/cn/LAB 06$ ./a.out
Connected to server at 127.0.0.1:8080
Enter the name of the file to create on the client: c.txt
Enter contents for the file 'c.txt':
hello charish!!
File 'c.txt' created successfully.
Server requested file: c.txt
Enter the name of the file to request from the server: m.txt
File 'm.txt' received successfully.

--- File Contents of 'm.txt' ---
hello pranay!!
--- End of File ---
```