HW3

Server.c

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

struct file_params{
    char *ip_address;
    int transfer_port;
    int start_index;
    int offset;
    char filename[1024];
};


void send_chunk(int server_socket, struct file_params* ptr){

    FILE *fp = fopen(ptr->filename, "r");
    printf("Sending data. start_index: %d, offset: %d\n", ptr->start_index, ptr-
>offset);
    int n;
    char data[1] = {0};
    int count = 0;

    fseek(fp, ptr->start_index, SEEK_SET);
    while (((n = fread(data, sizeof(char), 1, fp)) > 0) && (count <= ptr-
>offset)){

        if (send(server_socket, data, n, 0) < 0){
            perror("Error: Error in sending file.");
            exit(1);
        }

        bzero(data, 1);
        count++;
    }
}

void *thread(void* p){
```

```
    struct file_params* ptr = (struct file_params*) p;
    int server_socket, client_socket, n;
    struct sockaddr_in server_addr, client_address;
    socklen_t address_size;

    printf("[/] Socket port: %d\n", ptr->transfer_port);
    server_socket = socket(AF_INET, SOCK_STREAM, 0);

    if (server_socket < 0){
        perror("Error: Error in socket.\n");
        exit(1);
    }

    printf("Server socket created successfully.\n");
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = ptr->transfer_port;
    server_addr.sin_addr.s_addr = inet_addr(ptr->ip_address);
    n = bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr));

    if (n < 0){
        exit(1);
    }

    printf("Binding successfull.\n");
    if (listen(server_socket, 10) == 0){
        printf("Listening....\n");
    }

    else{
        exit(1);
    }

    address_size = sizeof(client_address);
    client_socket = accept(server_socket, (struct sockaddr*)&client_address,
&address_size);
    printf("Client connected!\n");
    send_chunk(client_socket, ptr);
    printf("Data sent to the Client successfully.\n");
    close(client_socket);
    return p;
}

int main(int argc){
    char *ip = "127.0.0.1";
    int port=5566;
```

```
    int server_socket, client_socket;
    struct sockaddr_in server_address, client_address;
    socklen_t address_size;
    char buffer[1024];
    int n;
    FILE *fp;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);

    if (server_socket < 0){
        perror("[-]Socket error");
        exit(1);
    }

    printf("[+]Server socket created successfully.\n");
    memset(&server_address,'\0',sizeof(server_address));
    server_address.sin_family = AF_INET;
    server_address.sin_port = port;
    server_address.sin_addr.s_addr = inet_addr(ip);

    n = bind(server_socket, (struct sockaddr*)&server_address,
sizeof(server_address));

    if (n < 0){
        perror("[-]Binding Error");
        exit(1);
    }
    printf("[+]Binding successfull.\n");
    listen(server_socket, 5);
    printf(" Listening....\n");

    while (1){
        address_size = sizeof(client_address);
        client_socket = accept(server_socket, (struct sockaddr*)&client_address,
&address_size);
        printf("[+]Client connected.\n");

        bzero(buffer, 1024);
        recv(client_socket, buffer, 1024, 0);
        printf("[+]Filename received: %s \n", buffer);

        strcat(buffer, '\0');
        char filename[1024];
        strcat(filename, buffer);
        fp = fopen(buffer, "rb");
```

```
        int size = ftell(fp);
        fseek(fp, 0L, SEEK_SET);
        printf("File Size: %d", size);

        int n_sessions = 1;
        bzero(buffer, 1024);

        n_sessions = ntohl(n_sessions);
        printf("The number of threads requested is: %d\n", n_sessions);

        int new_size = htonl(size);

        printf("File size Sent: %d\n", size);

        pthread_t tid[n_sessions];
        struct file_params params[n_sessions];
        int offset = (size / n_sessions);

        for (int i = 0; i < n_sessions; i++){
            params[i].start_index = i*offset;
            params[i].ip_address = ip;
            params[i].transfer_port = port + i*5 + 1;
            strcpy(params[i].filename,"");
            strcat(params[i].filename, filename);
            printf("Calling thread %d with port %d.\n", i,
params[i].transfer_port);

            if (i == n_sessions - 1){
                params[i].offset = size - offset*(n_sessions);
            }

            else{
                params[i].offset = offset;
            }

            pthread_create(&tid[i], NULL, thread, &params[i]);
        }

        for (int i = 0; i < n_sessions; i++){
            pthread_join(tid[i], NULL);
        }

        bzero(buffer, 1024);
        close(client_socket);
```

```
        close(server_socket);
        fclose(fp);
        break;
    }
    return 0;
}
```

Client.c

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

struct file_params{
    int offset;
    int start_index;
    char file_id[3];
    FILE *file_to_receive;
    char *fname;
    char *ip_address;
    char *file_data;
    int transfer_port;

};

void *file_write(int sock, char *duplicate, struct file_params* ptr){

    int n = 0;
    FILE *fp;
    char filename[1024] = "";
    strcat(filename, ptr->file_id);
    strcat(filename, duplicate);
    char buffer[1];
    bzero(buffer, 1);

    fp = fopen(filename, "w");
    while (1){
        n = recv(sock, buffer, 1, 0);

        if (n <= 0){
            break;
        }
```

```
        int write_size = fwrite(buffer, sizeof(char), n, fp);

        if(write_size < n){
            perror("[-] File write failed on your pc.\n");
            exit(1);
        }
        bzero(buffer, 1);
    }
    fclose(fp);
    return ptr;
}

void *thread(void* p){

    struct file_params* ptr = (struct file_params*) p;
    int sock, n;
    struct sockaddr_in address;
    sock = socket(AF_INET, SOCK_STREAM, 0);

    if(sock < 0){
        perror("[-] Error in socket");
        exit(1);
    }

    printf("*** Server socket created successfully.\n");
    address.sin_family = AF_INET;
    address.sin_port = ptr->transfer_port;
    address.sin_addr.s_addr = inet_addr(ptr->ip_address);

    printf("[/] Establishing connection with socket port: %d\n", ptr-
>transfer_port);
    sleep(2);
    n = connect(sock, (struct sockaddr*)&address, sizeof(address));

    if(n == -1){
        perror("[-] Error in socket");
        exit(1);
    }

    file_write(sock, ptr->fname, ptr);
    printf("File data sent successfully on port: %d.\n", ptr->transfer_port);
    printf("Closing the connection.\n");
    close(sock);
    return p;
}
```

```c
int main(int argc, char **argv){
    char *ip = "127.0.0.1";
    int port=5566;
    int sock;
    struct sockaddr_in address;
    socklen_t address_size;
    char buffer[1024];
    int n;
    FILE *fp;
    char filename[64];
    printf("Enter the filename: ");
    scanf("%s", filename);
    int total_threads;
    printf("Enter the number of threads: ");
    scanf("%d", &total_threads);

    pthread_t tid[total_threads];
    struct file_params params[total_threads];
    sock = socket(AF_INET, SOCK_STREAM, 0);

    if(sock < 0){
        perror("Error: Error in socket");
        exit(1);
    }
    printf("[+]Server socket created successfully.\n");
    memset(&address,'\0',sizeof(address));
    address.sin_family = AF_INET;
    address.sin_port = port;
    address.sin_addr.s_addr = inet_addr(ip);

    connect(sock, (struct sockaddr*)&address, sizeof(address));
    printf("[+]Connected to the server.\n");

    int sessions = htonl(total_threads);

    int filesize = 0;

    filesize = ntohl(filesize);
    printf("File size received: %d\n", filesize);

    int offset = (filesize / total_threads);

    for (int i = 0; i < total_threads; i++){
        params[i].file_to_receive = fp;
```

```
        params[i].fname = filename;

        sprintf(params[i].file_id, "%d", i);
        params[i].ip_address = ip;
        params[i].transfer_port = port + i*5 + 1;


        if (i == total_threads - 1){
            params[i].offset = filesize - offset*(total_threads);
        }

        else{
            params[i].offset = offset;
        }

        params[i].start_index = offset*i;
        params[i].file_data = (char *) malloc(sizeof(char)*offset);
        printf("Calling thread %d with port %d.\n", i, params[i].transfer_port);
        pthread_create(&tid[i], NULL, thread, &params[i]);
    }

    for (int i = 0; i < total_threads; i++){
        pthread_join(tid[i], NULL);
    }

    int last_position = 0;
    char final[1024] = "received_";
    strcat(final, filename);
    FILE *file_received = fopen(final, "a");

    for (int i = 0; i < total_threads; i++){
        char name[1024];
        sprintf(name, "%d", i);
        strcat(name, filename);

        FILE *handle = fopen(name, "r");
        char arr[offset];
        fread(arr, 1, offset, handle);
        fseek(file_received, last_position, SEEK_SET);
        fwrite(arr, 1, offset, file_received);
        last_position = i * offset;
        fclose(handle);
        remove(name);
    }
```

```
    fclose(file_received);
    printf("Closing the connection.\n");
    close(sock);

    for (int i = 0; i < total_threads; i++){
        free(params[i].file_data);
    }

    return 0;
}
```