HW1 solution-sample

main.c (server)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include "transfer.h"

int main(int argc, char *argv[]) {
    struct sockaddr_in client_addr, server_addr;
    socklen_t client_addr_size;
    int server_socket;
    int client_socket;

    FILE *fp;

    char filename[BUFFSIZE] = {0};

    // initialize server info
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(SERVERPORT);

    // create server_socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);

    if (server_socket == -1) {
        printf("Can't allocate server_socket\n");
        exit(1);
    }

    // binding server socket using server info
    if (bind(server_socket, (const struct sockaddr *) &server_addr, sizeof(server_addr)) == -1) {
        printf("Bind Error\n");
        exit(1);
    }

    // listen and queue size initialize 5
    if (listen(server_socket, 5) == -1) {
        printf("Listen Error\n");
        exit(1);
    }
```

```
    while (1) {

        // create client socket and accept
        memset(&client_addr, 0, sizeof(client_addr));
        client_addr_size = sizeof(client_addr);
        client_socket = accept(server_socket, (struct sockaddr *) &client_addr, &client_addr_size);

        if (client_socket == -1) {
            printf("Connect Error\n");
            close(client_socket);
            exit(1);
        }

        // receive file name
        if (recv(client_socket, filename, BUFFSIZE, 0) == -1) {
            printf("Can't receive filename\n");
        }

        // create file
        if ((fp = fopen(filename, "wb")) == NULL) {
            printf("Can't open file\n");
        }

        // receive file using loop
        receive_file(client_socket, fp);

        // check empty file
        if (fp != NULL) {
            fseek(fp, 0, SEEK_END);
            if (ftell(fp) != 0) {
                printf("%s Receive Success\n", filename);
                fclose(fp);
            }
            else {
                remove(filename);
            }
        }
        // flush for stdout buffer
        fflush(stdout);
        close(client_socket);
    }
}

void receive_file(int sockfd, FILE *fp) {
    ssize_t n;
    char buff[BUFFSIZE] = {0};
    while ((n = recv(sockfd, buff, BUFFSIZE, 0)) > 0) {
        fwrite(buff, sizeof(char), n, fp);
    }
```

transfer.h (server)

```
#ifndef TRANSFER_FILE_TRANSFER_H
#define TRANSFER_FILE_TRANSFER_H

#define SERVERPORT 8877
#define BUFFSIZE 4096

void receive_file(int sockfd, FILE *fp);

#endif
```

main.c (client)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <libgen.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include "transfer.h"

int main(int argc, char *argv[]) {
    struct sockaddr_in serveraddr;
    char *filename = basename(argv[1]); // get filename
    char buff[BUFFSIZE] = {0};
    strncpy(buff, filename, strlen(filename));
    FILE *fp;

    int client_socket;

    // initialize server info
    memset(&serveraddr, 0, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_port = htons(SERVERPORT);

    // create client client_socket
    client_socket = socket(AF_INET, SOCK_STREAM, 0);

    if (client_socket < 0) {
        printf("Can't allocate client_socket\n");
        exit(1);
    }

    // initialize server info
    if (inet_pton(AF_INET, "127.0.0.1", &serveraddr.sin_addr) < 0) {
        printf("IP address Convert Error\n");
        exit(1);
    }

    // connect server
    if (connect(client_socket, (const struct sockaddr *) &serveraddr, sizeof(serveraddr)) < 0) {
        printf("Connect Error\n");
        exit(1);
    }
```

```
    if (filename == NULL) {
        printf("Can't get filename¥n");
        exit(1);
    }

    // file open
    if ((fp = fopen(argv[1], "rb")) == NULL) {
        printf("Can't open file¥n");
        exit(1);
    }

    // send filename
    if (send(client_socket, buff, BUFFSIZE, 0) == -1) {
        printf("Can't send filename¥n");
        exit(1);
    }

    // send file
    send_file(fp, client_socket);

    fclose(fp);
    close(client_socket);
    return 0;
}

void send_file(FILE *fp, int sockfd) {
    int n;
    char buff[BUFFSIZE] = {0};
    while ((n = fread(buff, sizeof(char), BUFFSIZE, fp)) > 0) {
        if (n != BUFFSIZE && ferror(fp)) {
            printf("Read File Error");
            exit(1);
        }
        if (send(sockfd, buff, n, 0) == -1) {
            printf("Can't send file");
            exit(1);
        }
    }
}
```

transfer.h (client)

```
#ifndef TRANSFER_FILE_TRANSFER_H
#define TRANSFER_FILE_TRANSFER_H

#define SERVERPORT 8877
#define BUFFSIZE 4096

void send_file(FILE *fp, int sockfd);

#endif
```