ECE 40800 / CSCI 40300 Project 3

Introduction to Operating Systems

Caleb Kirby | John Baker | Dax Patel | Parth Patel

Introduction

The purpose of this project is to improve the media server developed in Project 2. The media server will now read configuration settings from a file, instead of expecting them in the form of a Command Line Argument. Further, the server will log the information of each request to the console. Logging information includes arrival time, the start time of response, complete-time of response, IP address of the client, ephemeral port number of clients, the content of the request. In addition to this, the server will implement random scheduling for processing client requests.

Requirements

Following are the brief requirements of the project:

- Implementing a random job scheduling algorithm.
- Create test scripts that will spawn multiple clients and make them communicate with the server
- Testing job scheduling with multiple clients
- All-time tags while logging must be recorded wall-clock times in nano-second resolution.
- A comparison metrics between FIFO and Random schedule shall be created.

Methodology

A. **Random Scheduling**
In the previous project, a job queue was used to enqueue and de queue tasks for threads and process clients in a FIFO way. To add random scheduling, a new function **random_dequeue** was created. This function randomly generates an integer between **0** and the **number of jobs in the job_queue.** After, this function swaps the job at a randomly generated index with the job present at the front of the queue. Finally, this function will call de queue, which will now return the first job in the queue.

B. **Logging Mechanism**
To implement the logging mechanism, the **timespec struct** and the **get_timespec** function built-in time. h were used. An extra wrapper function was created called **get_time_spec_to_string.** This function gets the current date and time, to nanosecond precision, and puts it in the buffer passed in as a parameter. This function is then used everywhere in the client and server to create a logging message with a time stamp. Note that each logging message will have a time stamp.

C. **Shell Script to Spawn Multiple Client on Multiple Computers**
This was the tricky part of the project. The group did not write a script for ssh. Instead, the testing was done by spinning up a server on the **in-csciprrc01** UNIX server of the Computer Science department at IUPUI. Then each group member opened up 3-5 ssh sessions on IUPUI's **tesla** server, and **in-csci-rrpc02, in-csci-rrpc03**, and so on. Each session represented one client. A client RC script with commands was run on each client-server session. These same steps were done a few times for FIFO and RANDOM scheduling. Results were recorded to create comparison metrics between two scheduling techniques.

Comparison - FIFO vs. Random
For the test, the server ran on 3 threads, with a maximum of 10 requests to connect. At the time of highest traffic, there were 5 requests in the queue, and 3 were being processed by the server.

	FIFO	RANDOM
Throughput	Excellent	Excellent
Turn around Time	It was quick. Wait time was almost 2-5 seconds unless a client performed more than 1 command	Clients waited long sometimes, or sometimes less than 2 seconds.
Waiting Time	~3 .1 seconds on avg	~3.2 seconds
Response Time	Once connected, same as random	ONce connected same as FIFO/

The above metrics were developed from the time stamp difference from the server execution trace and the client execution trace. Waiting and turnaround time are interdependent. If their waiting time is long, turned around time will go up since the client has to wait for the server to assign a thread. Further, FIFO turns out to be more reliable than Random as it is a fair approach. If the situation is changed, for example, a client can have a server as long as many commands, then random will end up being a good choice as it might get chances to the client with fewer jobs to be done. But again it is random and therefore unreliable.

Results
Please look at the end of the appendix for source code, compilation trace, an execution trace.

Conclusion
In this project, we learned the implementation of random scheduling for an already functioning TCP server. Further, the group learned how to use timing functions in C to log in request arrival, completion, and time detail. The group also learned the pros and cons of FIFO and Random schedule.

Appendix

See code and execution traces from the next pages

Makefile

```
CC   = gcc
LDFLAGS = -lm -lnsl
CFLAGS  = -g
TARGET  = media_transfer parser server client

default: $(TARGET)

server: server.o
    gcc $(CFLAGS) -o $@ $? media_transfer.o parser.o $(LDFLAGS) -lpthread

client: client.o
    gcc $(CFLAGS) -o $@ $? media_transfer.o parser.o $(LDFLAGS)

media_transfer: media_transfer.o
    gcc $(CFLAGS) -c media_transfer.c

parser: parser.o
    gcc $(CFLAGS) -c parser.c

clean:
    -rm -f *.o *~

cleanall: clean
    -rm -f $(TARGET)
```

Server.c

```c
/* A simple echo server using TCP */
#include <arpa/inet.h>
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <netdb.h>
#include <netinet/in.h>
#include <pthread.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <poll.h>
#include <unistd.h>
#include <time.h>

#include "parser.h"
#include "media_transfer.h"
#include "queue.h"

#define SERVER_TCP_PORT     3000    /* well-known port */
#define HEADERLEN           256     /* header packet length */
#define CONFIG_BUFFER       256     /* length of buffer for config line */

typedef enum sched_type {
    FIFO,
    RANDOM
} sched_type_e;

typedef struct hanlder_arg {
    int port;                       /* port number server is running on */
    int client_socket;              /* port number of client to deal with */
} handler_arg_t;

typedef struct {
    int sd;                         /* server socket descriptor */
    int port;                       /* port number server will listen on */
    int num_threads;                /* no.of threads - can be modified using CL Arg #3 */
    int max_requests;               /* max no.of client requests server can have at any
    time - can be modified using CL Arg number #4 */
    char * directory;               /* place to look for media files */
    pthread_t *handlers;            /* array of threads server can handle client
    requests */
    pthread_mutex_t lock;           /* mutex lock to do some thread safe
    functionanlities */
    pthread_cond_t cond;            /* condition variable */
    Queue *job_queue;               /* process client request queue */
    sched_type_e scheduling_type;   /* FIFO or RANDOM processing */
} server_config_t;

/*
 *  @param config: struct to store config value from rc script
 *  @param confgirc: file to read config information from
 *  @returns: success - 1 or failure - 0
 *  reads config file in to config struct
 */
int parse_configuration(server_config_t *config, char *configrc);

/*
 *  @param config: server configurtion struct
 *  @returns number of chars printed
 *  prints server configuration summary
 */
int print_configuration(server_config_t *config);

/*
```

```c
 * @param filepath - name of the file for which extension is needed
 * @returns
 *      point to first char in extension
 */
const char *get_file_ext(const char *filename);

/*
 * @param config: servert config struct
 * initializes threads, and locks for config struct
 */
int initialize_thread_pool(server_config_t *config);

/*
 * @param arg - handler args passing
 * @returns
 *      1 if client wants to disconnect
 *      0 if client wants to continue
 * Fulfils client requests
 *
 */
void handle_request(void*arg);

/*
 * @param arg - server config arg will be passed
 * takes a job from job queue.
 */
void *watch_requests(void *arg);

int main(int argc, char * argv[]) {
    char * config_file = NULL;

    switch(argc) {
    case 1:
        config_file = "mserver.config";
        break;
    case 3:
        if (strcmp(argv[1], "-c") == 0) config_file = argv[2];
        break;
    }


    /* seed for random generator */
    srand(time(0));

    /* init server configuration */
    char pwd[BUFLEN];
    getcwd(pwd, BUFLEN);

    /* fill in default config */
    server_config_t config;
    config.port = SERVER_TCP_PORT;
    config.directory = pwd;
    config.num_threads = 4;
    config.max_requests = 10;
    config.job_queue = createQueue(config.max_requests);
    config.scheduling_type = RANDOM;

    /* override default config if file provided */
    if (argc == 3) {
        switch(parse_configuration(&config, config_file)) {
            case -1:
                fprintf(stderr, "Unable to open configuration file!\n");
                break;
            case -2:
                fprintf(stderr, "Configuration error!\n");
                break;
        }
    } else {
        parse_configuration(&config, config_file);
    }
    config.handlers  = (pthread_t*)malloc(sizeof(pthread_t)*(config.num_threads));

    /* switch current working dir to media dir */
    int ret = chdir(config.directory);
    if(ret != 0) {
        printf("cannot change to dir %s.\n", config.directory);
        exit(1);
    }

    struct sockaddr_in server;

    /* Create a stream socket */
    if ((config.sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        fprintf(stderr, "Can't create a socket\n");
        exit(1);
    }

    /* Bind an address to the socket */
    bzero((char *)&server, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_port = htons(config.port);
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(config.sd, (struct sockaddr *)&server, sizeof(server)) == -1) {
        fprintf(stderr, "Can't bind name to socket\n");
        exit(1);
    }

    /* initialize threads and mutex locks */
    initialize_thread_pool(&config);

    /* print sever configuration */
    print_configuration(&config);

    /* queue up to config.max_requests connect requests */
    listen(config.sd, config.max_requests);

    /* loop forever and add requests to queue */
    while(1) {

        /* get a new connection req on server socket*/
        int new_client_sd = accept(config.sd, NULL, NULL);

        /* if found a request, enqueue it for processing */
        if(new_client_sd > 0) {
            char enqueue_time[TIME_BUFFER_LEN];
            get_time_spec_to_string(enqueue_time, TIME_BUFFER_LEN);
            printf("\n%s: Main: Accepting New Connection: %d\n", enqueue_time,
            new_client_sd);

            handler_arg_t *arg = (handler_arg_t*)malloc(sizeof(handler_arg_t));
            arg->port = config.port;
            arg->client_socket = new_client_sd;

            printf("%s: Main: Adding New Client to the Job queue...\n", enqueue_time);
            /* Locks the queue to add job */
            pthread_mutex_lock(&(config.lock));

            /* add connectiong to queue */
            enqueue(config.job_queue, (void*) arg);

            /* give up the lock on the queue */
            pthread_mutex_unlock(&(config.lock));

            get_time_spec_to_string(enqueue_time, TIME_BUFFER_LEN);
            printf("%s: Main: Added New Client to the Job queue\n", enqueue_time);

        }
    }
```

```c
}

int parse_configuration(server_config_t *config, char *configrc) {
    if (configrc == NULL) return -1;
    if (config == NULL) config = malloc(sizeof(server_config_t));

    FILE * c_file = fopen(configrc, "r");
    if (c_file == NULL) return -1;

    char buf[CONFIG_BUFFER];

    while (fgets(buf, CONFIG_BUFFER, c_file) != NULL) {
        if (strstr(buf, "#")) *(strstr(buf, "#")) = '\0';
        if (strstr(buf, "\n")) *(strstr(buf, "\n")) = '\0';
        if (buf[0] == '\0') continue;
        if (strstr(buf, ":") == NULL) return -2;

        char *split = strstr(buf, ": ");
        *split = '\0';

        char *key = buf;
        char *value = split + 2;

        // Fast and loose config parsing, only checking to see if config
        // line contains the key value, thus, if you have something like:
        // PortNumThreads: 5, it will match to PortNum and nothing else.
        if (strstr(key, "PortNum")) config->port = atoi(value);
        else if (strstr(key, "Threads")) config->num_threads = atoi(value);
        else if (strstr(key, "Sched")) {
            if (strcmp(value, "FIFO") == 0) config->scheduling_type = FIFO;
            else if (strcmp(value, "Random") == 0) config->scheduling_type = RANDOM;
        } else if (strstr(key, "Directory")) {
            config->directory = malloc(strlen(value));
            strcpy(config->directory, value);
        }
    }
}

int print_configuration(server_config_t *config) {
    printf("*******Server configuration*******\n");
    printf("Port Number: %d\n", config->port);
    printf("Num Threads: %d\n", config->num_threads);
    printf("Max Reqs: %d\n", config->max_requests);
    printf("Media Path: %s \n", config->directory);
    printf("*********************************\n");
    return 0;
}

const char *get_file_ext(const char *filename) {
    const char *dot_loc = strrchr(filename, '.');
    if(!dot_loc || dot_loc == filename) {
        return "Unknown";
    }
    return dot_loc + 1;
}

int initialize_thread_pool(server_config_t *config) {
    if (pthread_mutex_init(&(config->lock), NULL) != 0) {
        printf("\n mutex init has failed\n");
        return -1;
    }

    for (int i = 0; i < config->num_threads; ++i) {
        if(pthread_create(&(config->handlers[i]), NULL, watch_requests, (void*)config)
        != 0) {
            printf("Failed to create a thread");
            exit(1);
        }
    }
    return 0;
}

void *watch_requests(void *arg) {

    server_config_t *config = (server_config_t*)arg;

    void *job = NULL;

    while(1) {

        pthread_mutex_lock(&(config->lock));

        if(!isEmpty(config->job_queue)) {
            if(config->scheduling_type == FIFO) {
                job = dequeue(config->job_queue);
            }
            else {
                job = random_dequeue(config->job_queue);
            }
        }

        pthread_mutex_unlock(&(config->lock));

        if(job) {
            char time_processing_start[TIME_BUFFER_LEN];
            get_time_spec_to_string(time_processing_start, TIME_BUFFER_LEN);
            printf("%s: Watch Request: Thead %lu: Handling client %d\n",
            time_processing_start, pthread_self(), ((handler_arg_t*)job)->client_socket);
            handle_request(job);
        }

        job = NULL;
    }
}

void handle_request(void *client_sd)
{
    /* Some vairable declaration */
    char time_buf[TIME_BUFFER_LEN];
    handler_arg_t* info = ((handler_arg_t*)client_sd);

    /* Print out client information */
    struct sockaddr_in client_socket_addr;
    socklen_t len;
    len = sizeof(client_socket_addr);
    char client_ip[32];
    unsigned int ephemeral_port;

    bzero(&client_socket_addr, len);

    if (getsockname(info->client_socket, (struct sockaddr *)&client_socket_addr, &len)
    == 0) {
        /* get ip and the temp port*/
        inet_ntop(AF_INET, &client_socket_addr.sin_addr, client_ip, sizeof(client_ip));
        ephemeral_port = ntohs(client_socket_addr.sin_port);

        /* print contents of ss*/
        get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
        printf("%s: Handle Request: Client IP: %s Ephemeral Port: %ld\n", time_buf,
        client_ip, ephemeral_port);
        fflush(stdout);
    }

    while(1) {
        char buf[BUFLEN] = {0};
        char *bp = buf;
        int  bytes_to_read = BUFLEN;
        int  n = 0;
        while ((n = read(info->client_socket, bp, bytes_to_read)) > 0) {
            bp += n;
            bytes_to_read -= n;
        }

        if (bp <= 0) {
            // client probably disconnected
            close(info->client_socket);
        }
        get_time_spec_to_string(time_buf, BUFLEN);
        printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Command Recevied string: %s", time_buf, client_ip, ephemeral_port, buf);

        /* put a null character at the end */
        int size = strlen(buf);
        buf[strcspn(buf, "\n")] = 0;

        switch(get_command_from_request(buf)) {
            case LIST: {
                char listing[1024];
                get_media_list(".", listing, 1024);
                // send the header packet
                send_header(info->client_socket, info->port, strlen(listing), "Text", 100);
                if(send(info->client_socket, listing, strlen(listing), 0) == -1) {
                    get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
                    printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Error sending list\n", time_buf, client_ip, ephemeral_port);
                }
                break;
            }
            case GET: {
                // get the length of the file needed to be read.
                FILE *fp = fopen(&(buf[4]), "rb");

                if (fp == NULL) {
                    send_header(info->client_socket, info->port, 0, "", 404);
                    break;
                }

                fseek(fp, 0L, SEEK_END);
                size_t len = ftell(fp);
                fseek(fp, 0L, SEEK_SET);
                fclose(fp);

                // get file extension
                const char *extension = get_file_ext(buf + 4);

                // send header information
                send_header(info->client_socket, info->port, len, extension, 100);

                get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
                printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Sent Header Information\n", time_buf, client_ip, ephemeral_port);

                // send requested media
                send_media(info->client_socket, buf + 4, len);

                get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
                printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Sent: %s\n", time_buf, client_ip, ephemeral_port, buf);
                break;
            }
            case EXIT:
                close(info->client_socket);
                get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
                printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Closed connection with client: %d\n", time_buf, client_ip, ephemeral_port, info->client_socket);
                return ;
            default:
                // invalid request header
                send_header(info->client_socket, info->port, 0, "", 301);
                get_time_spec_to_string(time_buf, TIME_BUFFER_LEN);
                printf("%s: Handle_Request: Client IP: %s Ephemeral Port: %ld : Invalid
                request\n", time_buf, client_ip, ephemeral_port);
            break;
        }
    }
}
```

Client.c

```c
/* A simple TCP client */
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h> //Added string library
#include <strings.h>//For bzero function
#include <stdlib.h> //Added standard library
#include <unistd.h>
#include <signal.h>

#include "media_transfer.h"
#include "parser.h"

#define SERVER_TCP_PORT     (3000)

int main(int argc, char **argv)
{
    sigaction(SIGPIPE, &(struct sigaction){SIG_IGN}, NULL);

    int     n, bytes_to_read;
    int     batch_mode = 0;
    int     sd, port;
    struct  hostent     *hp;
    struct  sockaddr_in     server;
    char    *host, *bp, rbuf[BUFLEN], sbuf[BUFLEN];

    switch(argc) {
    case 2:
        host = argv[1];
        if (strrchr(host, ':')) {
            port = atoi(strrchr(host, ':') + 1);
            char *ope = strrchr(host, ':');
            *ope = 0;
        } else port = SERVER_TCP_PORT;
        break;
    case 3:
        host = argv[1];
        if (strrchr(host, ':')) {
            port = atoi(strrchr(host, ':') + 1);
            char *ope = strrchr(host, ':');
            *ope = 0;
        } else port = SERVER_TCP_PORT;
        batch_mode = 1;
        break;
    default:
        fprintf(stderr, "Usage: %s <host>[:port] [script]\n", argv[0]);
        exit(1);
    }

    /* Create a stream socket */
    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        fprintf(stderr, "Can't create a socket\n");
        exit(1);
    }

    /* Find the server to connect to */
    bzero((char *)&server, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_port = htons(port);
    if ((hp = gethostbyname(host)) == NULL) {
        fprintf(stderr, "Can't get server's address\n");
        exit(1);
    }

    printf("h_length = %d\n", hp->h_length);

    bcopy(hp->h_addr_list[0], (char *)&server.sin_addr, hp->h_length);

    /* Connecting to the server */
    if (connect(sd, (struct sockaddr *)&server, sizeof(server)) == -1) {
        fprintf(stderr, "Can't connect\n");
        exit(1);
    }
    printf("Connected: server's address is %s\n", hp->h_name);

    if(batch_mode) {
        process_batch(sd, argv[2]);
    }
    else {
        char time_stamp[TIME_BUFFER_LEN];
        get_time_spec_to_string(time_stamp, TIME_BUFFER_LEN);
        while (1) {
            printf("%s: TX: ", time_stamp);
            fgets(sbuf, BUFLEN, stdin);         /* get user's text */
            if(strcmp(sbuf, "exit\n") == 0) {
                write(sd, sbuf, BUFLEN);
                close(sd);
                break;
            }
            else {
                printf("%s: Sent Command: %s\n", time_stamp, sbuf);
                handle_command(sd, sbuf, BUFLEN);
            }
        }
    }
    return 0;
}
```

Parser.h

```c
#ifndef _PARSER_H_
#define _PARSER_H_

#define BUFLEN              (256)     /* buffer length */
#define TIME_BUFFER_LEN     128     /* lenght of time buffer to print time stamp*/

typedef struct {
    int status;
    size_t length;
    char *type;
    char *host;
} header;

enum commands {
    INVALID,
    LIST,
    GET,
    COMMENT,
    EXIT
};

typedef enum commands command_t;

/*
 * A contructor function for header struct
 * @returns an empty header struct
 */
header create_header();

/*
 * @param request - string line to validate
 * @returns
 *      1 if valid, -1 if not
 */
command_t get_command_from_request(const char *request);

/*
 * @param header - buffer containing header information
 * @param line_number - spefic line of header buffer to return
 * @returns
 *      a particular line from header buffer
 */
char * get_line(char * header_text, unsigned int line_number);

/*
 * @param string - buffer to find occurence of chracter from
 * @param c      - value of char whose occurence to be found
 * @param n      - number of occurences to be found
 * @returns      - position index of the nth occurence.
 */
int get_occurrence_n(char * string, char c, int n);

/*
 *  @param buf - buffer to store the time spec in
 *  @param buflen - size of the buffer
 */
void get_time_spec_to_string(char *buf, size_t buflen);

/*
 * @param str - string to find the number of lines it contains
 * @returns   - number of lines in a string
 */
int count_lines(char const *str);

/*
 * @param socket - socket id to receive header text from
 * @returns      - prints and then returns a buffer containing header text
 */
char * read_header_text(int socket);
```

```c

/*
 * @param header_text - buffer to read from
 * @param header_ptr  - storage location to store information
 * @returns           - success or failure
 */
int buffer_to_header(char * header_text, header *ptr);

/* Handle command from a string value
 * @param socker        - socket to use for server communication
 * @param command       - command string read from usr or file
 * @param len           - len of incoming command
 * @returns             - success or failure
 */
int handle_command(int socket, char *command, int len);

/*
 * Handle any command request from client
 * @param server_socket - socket to communicate to server
 * @param command       - string containing the full get <filename>
 * @param               - strlen of command
 * @returns - success or failure
 */
int process_command(int server_socket, char *command, int len);

/*
 * Runs commands from batch script
 * @param clientrc_path - path to read client commands from
 */
int process_batch(int socket, char * clienrc_path);

#endif
```

Parser.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include "media_transfer.h"
#include "parser.h"

/*
 * This functions checks if a request contains
 * "list" or "get" as the first few bytes. Function,
 * then returns a command type based on request.
 */
command_t get_command_from_request(const char *request) {
    if(request == NULL) {
        return INVALID;
    }
    else if(request[0] == '#') {
        return COMMENT;
    }
    else if(strncmp(request, "list", 4) == 0) {
        return LIST;
    }
    else if(strncmp(request, "get", 3) == 0) {
        int len = strlen(request);
        if(len <= 4) { // no file name specified
            printf("No file name specified for get command\n");
            return INVALID;
        }
        return GET;
    }
    else if(strncmp(request, "exit", 4) == 0) {
        return EXIT;
    }
    else {
        return INVALID;
    }
}

/*
 * A contructor function for header struct
 * @returns an empty header struct
 */
header create_header() {
    header h;
    h.status = 0;
    h.length = 0;
    h.type = 0;
    h.host = 0;

    return h;
}

/*
 * @param string - buffer to find occurence of chracter from
 * @param c      - value of char whose occurence to be found
 * @param n      - number of occurences to be found
 * @returns      - position index of the nth occurence.
 */
int get_occurrence_n(char * string, char c, int n) {
    if (string != NULL) {
        int occ = 0;
        for (int i = 0; i < strlen(string); i++) {
            if (string[i] == c) {
                if ((++occ) == n) return i;
            }
        }
    }

```

```c
        return -1;
    }

    void get_time_spec_to_string(char *buf, size_t buflen) {
        struct timespec ts;
        timespec_get(&ts, TIME_UTC);
        char temp[buflen];
        strftime(temp, buflen, "%D %T", gmtime(&ts.tv_sec));
        sprintf(buf, "%s.%09ld UTC", temp, ts.tv_nsec);
    }

    /*
     * @param str - string to find the number of lines it contains
     * @returns   - number of lines in a string
     */
    int count_lines(char const *str)
    {
        char const *p = str;
        int count;
        for (count = 0; ; ++count) {
            p = strstr(p, "\r\n");
            if (!p)
                break;
            p = p + 2;
        }
        return count - 1;
    }

    /*
     * @param header - buffer containing header information
     * @param line_number - spefic line of header buffer to return
     * @returns
     *       a particular line from header buffer
     */
    char * get_line(char * header_text, unsigned int line_number) {
        char * ret = 0;
        int line_count = 1;
        int start = -2;
        int cur = 0;
        for (int i = 0; i < line_number; ++i) {
            start = cur;
            cur = start + 2;
            while (header_text[cur] && header_text[cur] != '\r') {
                if (header_text[cur + 1] && header_text[cur + 1] == '\n') break;
                cur++;
            }

            if (header_text[cur + 2] && header_text[cur + 2] == '\r') {
                if (header_text[cur + 3] && header_text[cur + 3] == '\n') {
                    break;
                }
            }

            line_count++;
        }
        if (line_number > line_count) return NULL;

        if (line_number == 1) {
            ret = calloc(cur + 1, sizeof(char));
            strncpy(ret, header_text, cur);
        }
        else {
            ret = calloc(cur - start - 1, sizeof(char));
            strncpy(ret, header_text + start + 2, cur - start - 2);
        }

        return ret;
    }
```

```c
/*
 * @param socket - socket id to receive header text from
 * @returns      - prints and then returns a buffer containing header text
 */
char * read_header_text(int socket) {
    char buffer[BUFLEN] = {0};
    int buf_ind = 0;
    int ret_size = 0;
    int cont = 1;
    char *header_text = NULL;
    while (cont) {
        while (buf_ind < BUFLEN && 1 == read(socket, &buffer[buf_ind], 1)) {
            if (buf_ind > 2                 &&
                '\n' == buffer[buf_ind]     &&
                '\r' == buffer[buf_ind - 1] &&
                '\n' == buffer[buf_ind - 2] &&
                '\r' == buffer[buf_ind - 3])
            {
                cont = 0;
                break;
            }
            buf_ind++;
        }

        buf_ind++;


        if (header_text == NULL) {
            header_text = (char*)malloc(buf_ind * sizeof(char) + 1);
            memset(header_text, 0, buf_ind + 1);
            strncpy(header_text, buffer, buf_ind);

            ret_size = buf_ind + 1;
        } else {
            header_text = (char*) realloc(header_text, (ret_size += buf_ind));
            memset(header_text + ret_size - 1, 0, 1);
            strncat(header_text, buffer, buf_ind);
        }

        memset(buffer, 0, BUFLEN);
        buf_ind = 0;
    }

    //printf("%s\n", header_text);
    return header_text;
}

/*
 * @param header_text - buffer to read from
 * @param h           - storage location to store information
 * @returns           - success or failure
 */
int buffer_to_header(char * header_text, header *h) {

    if(!header_text) {
        return -1;
    }

    char * line = NULL;
    int current = 1;
    int additional_count = 0;
    while ((line = get_line(header_text, current)) != NULL) {
        int token_loc = get_occurrence_n(line, ':', 1);
        if (token_loc > 0) {
            char key[token_loc + 1];
            char value[strlen(line) - token_loc];

            memset(key, 0, sizeof(key));
            memset(value, 0, sizeof(value));

            for (int i = 0; i < sizeof(key) - 1; i++) key[i] = line[i];
            for (int i = 0; i < sizeof(value) - 1; i++) value[i] = line[token_loc + i +
            2];

            if (strcmp(key, "Status") == 0) h->status = atoi(value);
            else if (strcmp(key, "Host") == 0) {
                h->host = malloc(sizeof(value));
                strcpy(h->host, value);
            } else if (strcmp(key, "Type") == 0) {
                h->type = malloc(sizeof(value));
                strcpy(h->type, value);
            } else if (strcmp(key, "Length") == 0) h->length = atoi(value);
        }

        free(line);
        line = NULL;
        if (++current > count_lines(header_text)) break;
    }
    return 1;
}

/* Handle command from a string value
 * @param socker        - socket to use for server communication
 * @param command       - command string read from usr or file
 * @param len           - len of incoming command
 * @returns             - success or failure
 */
int handle_command(int socket, char *command, int len) {
    switch (get_command_from_request(command)) {
        case GET:
            process_command(socket, command, BUFLEN);
            break;
        case LIST:
            process_command(socket, command, BUFLEN);
            break;
        case EXIT:
            printf("Good bye\n");
            return 1;
            break;
        case INVALID:
            printf("Invalid Command: %s\n", command);
        default:
            break;
    }
    return 1;
}

/*
 * Handle get request from client
 * @param server_socket - socket to communicate to server
 * @returns - success or failure
 */
int process_command(int server_socket, char *command, int len) {

    /* send out user command */
    write(server_socket, command, len);

    // read header response
    char *header_text = read_header_text(server_socket);
    char time_stamp[TIME_BUFFER_LEN];
    get_time_spec_to_string(time_stamp, BUFLEN);
    printf("%s: Header Response Received\n", time_stamp);
    if(!header_text) {
        perror("fatal error\n");
    }

    // store buffer information to header struc
    header h = create_header();
    buffer_to_header(header_text, &h);

    free(header_text);
    header_text = NULL;

    get_time_spec_to_string(time_stamp, TIME_BUFFER_LEN);
    printf("%s: Status:%d  Host:%s  Length:%ld  Type:%s  \n", time_stamp,h.status,
    h.host, h.length, h.type);

    switch (h.status) {
        case 100:
            if (strcmp(h.type, "Text") == 0) {
                char list[h.length + 1];
                list[h.length];
                memset(list, 0, h.length + 1);

                size_t received = 0;

                while (received < h.length) {
                    if (read(server_socket, list + received, 1)) ++received;
                }

                printf("%s\n", list);
                get_time_spec_to_string(time_stamp, TIME_BUFFER_LEN);
                printf("%s: File Listing Received\n", time_stamp);
            }
            else {
                command[strcspn(command, "\n")] = 0;
                // get output name of the file from user
                char output_name[BUFLEN];
                printf("%s: Name of the file to put data received from server to: ",
                time_stamp);
                fgets(output_name, BUFLEN, stdin);
                output_name[strcspn(output_name, "\n")] = 0;

                // store to the output file
                receive_media(server_socket, output_name, h.length);
                get_time_spec_to_string(time_stamp, TIME_BUFFER_LEN);
                printf("%s: Media Received and Downloaded\n", time_stamp);
            }
            break;
        case 301:
            fprintf(stderr, "Unknown command!\n");
            break;
        case 404:
            fprintf(stderr, "File not found!\n");
            break;
        default:
            fprintf(stderr, "Undefined error!\n");
            break;
    }
}


/*
 * Runs commands from batch script
 * @param clientrc_path - path to read client commands from
 */
int process_batch(int socket, char * clienrc_path) {
    if(!clienrc_path) {
        perror("Could not find script path\n");
        return - 1;
    }

    FILE* fp = fopen(clienrc_path, "r");
    if(!fp) {
        perror("Could not find script path\n");
        return -1;
    }
    char buffer[BUFLEN];
    while(fgets(buffer, BUFLEN, fp)){
        switch(get_command_from_request(buffer)) {
            case GET:
                handle_command(socket, buffer, BUFLEN);         /* send it out */
                break;
            case LIST:
                handle_command(socket, buffer, BUFLEN);         /* send it out */
                break;
            case EXIT:
                return 1;
            default:
                break;
        }
    }
}
```

media_transfer.h

```c
#ifndef _MEDIA_TRASNFER_H_
#define _MEDIA_TRASNFER_H_

#include <stdio.h>

/*
 * @param fp       - pointer to the media to be sent
 * @param sockfd   - client socke to send the media to
 */
int send_media(int sockfd, const char *media_path, size_t length);

/*
 * @param sockfd   - client socket to receive the media on
 * @param filename - filename to write received data to
 */
int receive_media(int sockfd, const char *media_path, size_t length);

/*
 * @param path        - sends lists all the media under this path
 * @param buffer      - place to store the listing to
 * @param buffer_size - size of the buffer passed
 * @returns
 *      1 if success, -1 if failure
 */
int get_media_list(const char *path, char *buffer, size_t buffer_size);

/*
 * @param client_socket - client socket to send header to
 * @param port          - port socket is hosted on
 * @param media_size    - size of media to be sent
 * @param media_type    - type of the media to be sent
 * @returns
 *      1 if sucess, -1 if fail
 */
int send_header(int client_socket, int port, size_t media_size, const char *media_type,
int status);

#endif
```

media_transfer.c

```c
#include <arpa/inet.h>
#include <dirent.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "media_transfer.h"

#define LEN 1024

int send_media(int sockfd, const char *media_path, size_t length) {
    int n;
    char *data = malloc(length);

    FILE *fp = fopen(media_path, "rb");
    if(fp == NULL){
        printf("File: %s, not Found", media_path);
        return -1;
    }

    size_t sent = 0;
    fread(data, length, 1, fp);
    while(sent < length) {
        size_t t = send(sockfd, data, length, 0);
        if (t != -1) {
            sent += t;
        } else {
            perror("send_media");
            exit(1);
        }
    }

    fclose(fp);
    free(data);
    return 1;
}

int receive_media(int sockfd, const char *filename, size_t length) {
    unsigned int n = 0;
    size_t pos = 0;
    FILE *fp;
    char buffer[LEN];
    char *media = malloc(length);

    while (1) {
        n = read(sockfd, buffer, LEN);
        if (n < 0) continue;
        memcpy(media + pos, buffer, n);
        pos += n;
        if (pos >= length) break;
    }

    fp = fopen(filename, "w");
    fwrite(media, length, 1, fp);
    fclose(fp);
    free(media);

    return 1;
}

int get_media_list(const char *path, char *buffer, size_t buffer_size) {
    DIR *dh = opendir(path);
    struct dirent *d;
    struct stat fstat;

    int n = 0;
    n += sprintf(buffer, "\tSize\t\tName\n");
    while((d = readdir(dh)) != NULL) {
        stat(d->d_name, &fstat);
        n += sprintf(buffer + n, "\t%ld\t\t%s\n", fstat.st_size, d->d_name);
    }
    closedir(dh);
    return 1;
}

int send_header(int client_socket, int port, size_t media_size, const char *media_type,
int status) {
    char host[256];
    char *IP;
    struct hostent *host_entry;
    int hostname;

    //find the host name
    hostname = gethostname(host, sizeof(host));
    if(hostname == -1) {
        printf("Cannot find host information");
    }

    //find host information
    host_entry = gethostbyname(host);
    if(host_entry == NULL) {
        printf("Cannot find the host from id\n");
    }

    //Convert into IP string
    IP = inet_ntoa(*((struct in_addr*) host_entry->h_addr_list[0]));

    // create the header
    char header[LEN];
    int n = 0;
    n += sprintf(header, "Status: %d\r\n", status);                // req is valid
    n += sprintf(header + n, "Host: %s:%d\r\n", IP, port);          // append host
    information
    n += sprintf(header + n, "Type: %s\r\n", media_type);           // append file type
    n += sprintf(header + n, "Length: %ld\r\n\r\n", media_size);        // append file
    length

    // finally send the header packet
    if(send(client_socket, header, n, 0) == -1) {
        return -1;
    }
    else{
        return 0;
    }
}
```

queue.h

```c
#ifndef __QUEUE_H__
#define __QUEUE_H__

#include <limits.h>
#include <stdlib.h>

typedef struct {
    int front, rear, size;
    unsigned capacity;
    void** job;
} Queue;

Queue* createQueue(unsigned capacity)
{
    Queue* queue = (Queue*)malloc(
        sizeof(Queue));
    queue->capacity = capacity;
    queue->front = queue->size = 0;

    // This is important, see the enqueue
    queue->rear = capacity - 1;
    queue->job = (void*)malloc(
        queue->capacity * sizeof(int));
    return queue;
}

int isFull(Queue* queue)
{
    return (queue->size == queue->capacity);
}

// Queue is empty when size is 0
int isEmpty(Queue* queue)
{
    return (queue->size == 0);
}

void enqueue(Queue* queue, void* item)
{
    if (isFull(queue))
        return;
    queue->rear = (queue->rear + 1)
                  % queue->capacity;
    queue->job[queue->rear] = item;
    queue->size = queue->size + 1;
}

void* dequeue(Queue* queue)
{
    if (isEmpty(queue))
        return NULL;
    void* item = queue->job[queue->front];
    queue->front = (queue->front + 1)
                   % queue->capacity;
    queue->size = queue->size - 1;
    return item;
}

void* random_dequeue(Queue *queue)
{
    if (isEmpty(queue))
        return NULL;
    else if (queue->size == 1)
        return dequeue(queue);

    int lower_limit = 0;
    int upper_limit = queue->size - 1;

    int random_index = (rand() % (upper_limit - lower_limit) + 1) + lower_limit;

    /* swap the random index with the one at front and then call dequeue */

    /* get the pointer at random index, and make a copy of it*/
    void *temp = queue->job[random_index];

    /* the pointer at random index points to same place as front pointer*/
    queue->job[random_index] = queue->job[0];

    /* front pointer now points where the old random index pointed to */
    queue->job[0] = temp;

    /* return normal dequeue - random pointer will be returned */
    return dequeue(queue);
}

#endif
```

```
# mserver configuration file
# remove the pond sign to activate a configuration
PortNum: 1234
# Block: 2048
Threads: 4
# Buffers:  3
Sched: RANDOM
# Directory: /media/
```

dummy config file for server

```
[daxpate@in-csci-rrpc01 Project-3]$ script server-load.script
Script started, file is server-load.script
[daxpate@in-csci-rrpc01 Project-3]$
[daxpate@in-csci-rrpc01 Project-3]$ ./server
*******Server configuration*******
Port Number: 1234
Num Threads: 3
Max Reqs: 10
Media Path: /home/daxpate/ece40800/Project-3
**********************************

04/08/21 22:58:59.110122973 UTC: Main: Accepting New Connection: 5
04/08/21 22:58:59.110122973 UTC: Main: Adding New Client to the Job queue...
04/08/21 22:58:59.110227938 UTC: Main: Added New Client to the Job queue
04/08/21 22:58:59.110228632 UTC: Watch Request: Thead 139803194681088: Handling client 5
04/08/21 22:58:59.110255459 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 1234

04/08/21 22:59:01.695403789 UTC: Main: Accepting New Connection: 6
04/08/21 22:59:01.695403789 UTC: Main: Adding New Client to the Job queue...
04/08/21 22:59:01.695424819 UTC: Main: Added New Client to the Job queue
04/08/21 22:59:01.695423971 UTC: Watch Request: Thead 139803211466496: Handling client 6
04/08/21 22:59:01.695450680 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 1234

04/08/21 22:59:04.613565699 UTC: Main: Accepting New Connection: 7
04/08/21 22:59:04.613565699 UTC: Main: Adding New Client to the Job queue...
04/08/21 22:59:04.613588709 UTC: Main: Added New Client to the Job queue
04/08/21 22:59:04.613589976 UTC: Watch Request: Thead 139803203073792: Handling client 7
04/08/21 22:59:04.613623407 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 1234
04/08/21 22:59:05.552166743 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 22:59:08.546578521 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list

04/08/21 22:59:11.306005918 UTC: Main: Accepting New Connection: 8
04/08/21 22:59:11.306005918 UTC: Main: Adding New Client to the Job queue...
04/08/21 22:59:11.306043784 UTC: Main: Added New Client to the Job queue

04/08/21 22:59:13.402649233 UTC: Main: Accepting New Connection: 9
04/08/21 22:59:13.402649233 UTC: Main: Adding New Client to the Job queue...
04/08/21 22:59:13.402685486 UTC: Main: Added New Client to the Job queue
04/08/21 22:59:14.873625625 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get test.mp3
04/08/21 22:59:14.876993890 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 22:59:19.427418989 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get test.mp3
04/08/21 22:59:26.053774100 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get mserver.config
04/08/21 22:59:26.055197809 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 22:59:26.055366579 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get mserver.config
04/08/21 22:59:29.736352327 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 22:59:39.421926201 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get server.c
04/08/21 22:59:39.425904323 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 22:59:39.427146236 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get server.c
04/08/21 22:59:41.607423644 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get server
04/08/21 22:59:41.608999194 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 22:59:41.609245782 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get server
04/08/21 22:59:46.570303654 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 22:59:49.725497193 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: exit
04/08/21 22:59:49.725558550 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Closed connection with client: 7
04/08/21 22:59:49.725577533 UTC: Watch Request: Thead 139803203073792: Handling client 8
04/08/21 22:59:49.725593795 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 1234
04/08/21 22:59:49.725621605 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 22:59:54.867078018 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get client.html
04/08/21 22:59:54.870120226 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
```

```
04/08/21 22:59:54.870384944 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get client.html
04/08/21 23:00:02.102061001 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get send.txt
04/08/21 23:00:02.105633550 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 23:00:02.106337243 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get send.txt
04/08/21 23:00:07.115006958 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 23:00:12.831200944 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: exit
04/08/21 23:00:12.831258081 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Closed connection with client: 8
04/08/21 23:00:12.831274178 UTC: Watch Request: Thead 139803203073792: Handling client 9
04/08/21 23:00:12.831321460 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 1234
04/08/21 23:00:12.831340957 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 23:00:13.370999540 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 23:00:14.203767607 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 23:00:17.461355339 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: list
04/08/21 23:00:26.618958765 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get test.mp3
04/08/21 23:00:26.620562047 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 23:00:31.196703986 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get song.mp3
04/08/21 23:00:31.199712802 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 23:00:34.469675185 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get song.mp3
04/08/21 23:00:34.471211582 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 23:00:40.016987833 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get test.mp3
04/08/21 23:01:00.954741987 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: get song.mp3
04/08/21 23:01:00.956951114 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent Header Information
04/08/21 23:01:11.119970980 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get song.mp3
04/08/21 23:01:42.662720004 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get song.mp3
04/08/21 23:01:43.022612346 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Sent: get song.mp3
04/08/21 23:01:48.257015878 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: exit
04/08/21 23:01:48.257076075 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Closed connection with client: 9
04/08/21 23:02:09.298075087 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: exit
04/08/21 23:02:09.298114272 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Closed connection with client: 5
04/08/21 23:02:12.003400642 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Command Recevied string: exit
04/08/21 23:02:12.003425265 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 1234 :
Closed connection with client: 6
^C
[daxpate@in-csci-rrpc01 Project-3]$ exit
exit
Script done, file is server-load.script
[daxpate@in-csci-rrpc01 Project-3]$ git status
```

```
Script started on Thu 08 Apr 2021 07:20:17 PM EDT
[daxpate@in-csci-rrpc01 Project-3]$ ./server
*******Server configuration*******
Port Number: 3000
Num Threads: 3
Max Reqs: 10
Media Path: /home/daxpate/ece40800/Project-3
**********************************

04/08/21 23:20:31.640086852 UTC: Main: Accepting New Connection: 5
04/08/21 23:20:31.640086852 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:31.640221024 UTC: Main: Added New Client to the Job queue
04/08/21 23:20:31.640221126 UTC: Watch Request: Thead 139875339474688: Handling client 5
04/08/21 23:20:31.640265601 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 3000

04/08/21 23:20:34.612238389 UTC: Main: Accepting New Connection: 6
04/08/21 23:20:34.612238389 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:34.612259037 UTC: Main: Added New Client to the Job queue
04/08/21 23:20:34.612260051 UTC: Watch Request: Thead 139875331081984: Handling client 6
04/08/21 23:20:34.612290091 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 3000
04/08/21 23:20:38.175981635 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 3000 :
Command Recevied string: list

04/08/21 23:20:39.776127279 UTC: Main: Accepting New Connection: 7
04/08/21 23:20:39.776127279 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:39.776150170 UTC: Main: Added New Client to the Job queue
04/08/21 23:20:39.776149538 UTC: Watch Request: Thead 139875347867392: Handling client 7
04/08/21 23:20:39.776178395 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 3000

04/08/21 23:20:40.640938779 UTC: Main: Accepting New Connection: 8
04/08/21 23:20:40.640938779 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:40.640976789 UTC: Main: Added New Client to the Job queue

04/08/21 23:20:41.719012898 UTC: Main: Accepting New Connection: 9
04/08/21 23:20:41.719012898 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:41.719051702 UTC: Main: Added New Client to the Job queue
04/08/21 23:20:47.852451302 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:20:47.852451302 UTC Ephemeral Port: 3000 : Command Recevied string: get
r.mp3
04/08/21 23:20:47.854130804 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:20:47.854130804 UTC Ephemeral Port: 3000 : Sent Header Information
04/08/21 23:20:57.884379411 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:20:57.884379411 UTC Ephemeral Port: 3000 : Sent: get r.mp3

04/08/21 23:20:58.213331980 UTC: Main: Accepting New Connection: 10
04/08/21 23:20:58.213331980 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:20:58.213376001 UTC: Main: Added New Client to the Job queue
04/08/21 23:21:04.339736340 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 3000 :
Command Recevied string: list

04/08/21 23:21:13.756194070 UTC: Main: Accepting New Connection: 11
04/08/21 23:21:13.756194070 UTC: Main: Adding New Client to the Job queue...
04/08/21 23:21:13.756236799 UTC: Main: Added New Client to the Job queue
04/08/21 23:21:36.766463243 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:21:36.766463243 UTC Ephemeral Port: 3000 : Command Recevied string: get
song.mp3
04/08/21 23:21:36.768571274 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:21:36.768571274 UTC Ephemeral Port: 3000 : Sent Header Information
04/08/21 23:21:46.929214149 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:21:46.929214149 UTC Ephemeral Port: 3000 : Sent: get song.mp3
04/08/21 23:21:49.283459847 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 3000 :
Command Recevied string: list
04/08/21 23:21:52.454196108 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:21:52.454196108 UTC Ephemeral Port: 3000 : Command Recevied string: exit
04/08/21 23:21:52.454321366 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
        4321604/08/21 23:21:52.454321366 UTC Ephemeral Port: 3000 : Closed connection with client:
6
04/08/21 23:21:52.454361655 UTC: Watch Request: Thead 139875331081984: Handling client 8
04/08/21 23:21:52.454377907 UTC: Handle Request: Client IP: 10.234.136.55 Ephemeral Port: 3000
04/08/21 23:21:52.454396883 UTC: Handle_Request: Client IP: 10.234.136.55 Ephemeral Port: 3000 :
Command Recevied string: list
04/08/21 23:21:52.456726217 UTC: Handle_Request: Client IP:     13521           server-fifo
        166             mserver.config
        25928           server.o
[daxpate@in-csci-rrpc01 Project-3]$ exit
exit

Script done on Thu 08 Apr 2021 07:22:22 PM EDT
Script started on 2021-04-08 17:50:51-04:00 [TERM="xterm-256color" TTY="/dev/tty1" COLUMNS="148" LINES="32"]
%


  Project-3 git:(exe-traces)   ./client 192.168.0.124:1234

h_length = 4
Connected: server's address is 192.168.0.124
04/08/21 21:51:05.322832700 UTC: TX: list
04/08/21 21:51:05.322832700 UTC: Sent Command: list

04/08/21 21:51:09.347689600 UTC: Header Response Received
04/08/21 21:51:09.347758100 UTC: Status:100  Host:127.0.1.1:1234  Length:494  Type:Text
        Size            Name
        473             .gitignore
        0               typescript
        8196            a.out
        9733309         received.mp3
        4096            .git
        10716           client.o
        113             send.txt
        2706            media_transfer.c
        4096            .
        1125            media_transfer.h
        2353            client.c
        39              clientrc
        4096            ..
        21128           server.o
        17355           ansi2html.sh
        47200           server
        12479           server.c
        2615            parser.h
        8911            old.c
        34432           client
        9114            parser.c
        167             mserver.config
        10332           media_transfer.o
        9733309         r.mp3
        1961            queue.h
        9733309         song.mp3
        4096            .vscode
        15996           parser.o
        445             Makefile

04/08/21 21:51:09.392414800 UTC: File Listing Received
04/08/21 21:51:05.322832700 UTC: TX: get song.mp3
04/08/21 21:51:05.322832700 UTC: Sent Command: get song.mp3

04/08/21 21:52:17.021055600 UTC: Header Response Received
04/08/21 21:52:17.021092900 UTC: Status:100  Host:127.0.1.1:1234  Length:9733309  Type:mp3
04/08/21 21:52:17.021092900 UTC: Name of the file to put data received from server to: test.mp3
04/08/21 21:52:27.074399900 UTC: Media Received and Downloaded
04/08/21 21:51:05.322832700 UTC: TX: exit
%


  Project-3 git:(exe-traces)   mv typescript client.script

%
```

```
Script started on Thu 08 Apr 2021 06:47:33 PM EDT
[kirbycm@in-csci-rrpc03 Project-3]$ ./client in-csci-rrpc01:3000
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:50:30.389929154 UTC: TX: list
04/08/21 22:50:30.389929154 UTC: Sent Command: list

^C
[kirbycm@in-csci-rrpc03 Project-3]$ exit
exit

Script done on Thu 08 Apr 2021 06:52:18 PM EDT
Script started on 2021-04-08 17:50:35-04:00 [TERM="xterm-256color" TTY="/dev/pts/3" COLUMNS="195" LINES="47"]
pi@raspberrypi:~/repos/ece40800/Project-3 $ make
gcc -g   -c -o media_transfer.o media_transfer.c
gcc -g -c media_transfer.c
gcc -g   -c -o parser.o parser.c
gcc -g -c parser.c
gcc -g   -c -o server.o server.c
gcc -g -o server server.o media_transfer.o parser.o -lm -lnsl -lpthread
gcc -g   -c -o client.o client.c
gcc -g -o client client.o media_transfer.o parser.o -lm -lnsl
pi@raspberrypi:~/repos/ece40800/Project-3 $ ./server
*******Server configuration*******
Port Number: 1234
Num Threads: 4
Max Reqs: 10
Media Path: /home/pi/repos/ece40800/Project-3
**********************************

04/08/21 21:51:05.224235906 UTC: Main: Accepting New Connection: 5
04/08/21 21:51:05.224235906 UTC: Main: Adding New Client to the Job queue...
04/08/21 21:51:05.228580789 UTC: Watch Request: Thead 3066995808: Handling client 5
04/08/21 21:51:05.228665732 UTC: Handle Request: Client IP: 192.168.0.124 Ephemeral Port: 1234
04/08/21 21:51:05.228577159 UTC: Main: Added New Client to the Job queue
04/08/21 21:51:09.244611698 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: list

04/08/21 21:52:11.181261627 UTC: Main: Accepting New Connection: 6
04/08/21 21:52:11.181261627 UTC: Main: Adding New Client to the Job queue...
04/08/21 21:52:11.181328792 UTC: Main: Added New Client to the Job queue
04/08/21 21:52:11.181336551 UTC: Watch Request: Thead 3058603104: Handling client 6
04/08/21 21:52:11.181363477 UTC: Handle Request: Client IP: 192.168.0.124 Ephemeral Port: 1234
04/08/21 21:52:12.687719411 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: list
04/08/21 21:52:16.919490219 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: get song.mp3
04/08/21 21:52:16.919685789 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent Header Information
04/08/21 21:52:26.827175881 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent: get song.mp3
04/08/21 21:52:31.600172260 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: get r.mp3
04/08/21 21:52:31.600406108 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent Header Information
04/08/21 21:52:36.766682435 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent: get r.mp3
04/08/21 21:52:38.351151265 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: exit
04/08/21 21:52:38.351313521 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Closed connection with client: 6
04/08/21 21:52:40.313385603 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: exit
04/08/21 21:52:40.313656449 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Closed connection with client: 5
^C
pi@raspberrypi:~/repos/ece40800/Project-3 $ ls
 ansi2html.sh   client     client.o  'get receive.mp3'   media_transfer.c   media_transfer.o   old.c      parser.h   queue.h       r.mp3      server     server.o   typescript
 a.out          client.c   clientrc   Makefile           media_transfer.h   mserver.config     parser.c   parser.o   received.mp3  send.txt   server.c   song.mp3
pi@raspberrypi:~/repos/ece40800/Project-3 $ ans2
```

```
Script started on Thu 08 Apr 2021 06:17:06 PM EDT
[pakpatel@in-csci-rrpc04 Project-3]$ ./client in-csci-rrpc01:1234
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:17:41.439841515 UTC: TX: list
04/08/21 22:17:41.439841515 UTC: Sent Command: list

04/08/21 22:22:10.130516495 UTC: Header Response Received
04/08/21 22:22:10.130577486 UTC: Status:100  Host:10.234.136.55:1234  Length:602  Type:Text
        Size            Name
        4096            .
        50              ..
        4096            .git
        473             .gitignore
        39              clientrc
        1125            media_transfer.h
        113             send.txt
        9733309         song.mp3
        445             Makefile
        2353            client.c
        2706            media_transfer.c
        8911            old.c
        2615            parser.h
        1961            queue.h
        9733309         r.mp3
        9733309         received.mp3
        9114            parser.c
        17355           ansi2html.sh
        20596           client.html
        34304           client.script
        9733309         get receive.mp3
        22157           server-exe.html
        8192            server.script
        9733309         test.mp3
        12520           media_transfer.o
        18848           parser.o
        25928           server.o
        43216           server
        13312           client.o
        35728           client
        168             mserver.config
        12479           server.c
        0               dax-tesla.script

04/08/21 22:22:10.132477897 UTC: File Listing Received
04/08/21 22:17:41.439841515 UTC: TX: get r.mp3
04/08/21 22:17:41.439841515 UTC: Sent Command: get r.mp3

04/08/21 22:22:56.504876181 UTC: Header Response Received
04/08/21 22:22:56.504908514 UTC: Status:100  Host:10.234.136.55:1234  Length:9733309  Type:mp3
04/08/21 22:22:56.504908514 UTC: Name of the file to put data received from server to: receive-
parth.mp3
04/08/21 22:23:22.304170322 UTC: Media Received and Downloaded
04/08/21 22:17:41.439841515 UTC: TX: exit
[pakpatel@in-csci-rrpc04 Project-3]$ ./client in-csci-rrpc01:1234
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:25:17.481820634 UTC: TX: exit
[pakpatel@in-csci-rrpc04 Project-3]$ stop
bash: stop: command not found...
Similar command is: 'top'
[pakpatel@in-csci-rrpc04 Project-3]$ exit
exit

Script done on Thu 08 Apr 2021 06:27:32 PM EDT
Script started on Thu 08 Apr 2021 06:47:27 PM EDT
[pakpatel@in-csci-rrpc04 Project-3]$ ./client in-csci-rrpc01:3000
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:50:23.290318008 UTC: TX: list
04/08/21 22:50:23.290318008 UTC: Sent Command: list

04/08/21 22:50:53.022564103 UTC: Header Response Received
04/08/21 22:50:53.022638146 UTC: Status:100  Host:10.234.136.55:3000  Length:1196  Type:Text
        Size            Name
        4096            .
        50              ..
        4096            .git
        473             .gitignore
        39              clientrc
        1125            media_transfer.h
        113             send.txt
        9733309         song.mp3
        445             Makefile
        2353            client.c
        2706            media_transfer.c
        8911            old.c
        2615            parser.h
        1961            queue.h
        9733309         r.mp3
        9733309         received.mp3
        9114            parser.c
        17355           ansi2html.sh
        20596           client.html
        34304           client.script
        9733309         get receive.mp3
        22157           server-exe.html
        8192            server.script
        9733309         test.mp3
        12479           server.c
        5352            dax-tesla.script
        9733309         receive-dax.mp3
        194             typescript
        9097            server-load.script
        0               dax-tesla-2.script
        12479           server-copy.c
        20596           caleb-client.html
        168             caleb-mserver.config
        3956            caleb-rrpc03.script
        43216           caleb-server
        9733309         caleb-song.mp3
        20596           caleb-tesla-client.html
        9733309         caleb-tesla.mp3
        6991            caleb-tesla.script
        9733309         caleb-test.mp3
        8911            old-parth.c
        2178            parth-rrpc
        1927            parth-rrpc2
        113             parth-send.txt
        3494            parth-tesla
        2550            parth-tesla2
        9733309         parth-test.mp3
        9733309         receive-parth.mp3
        9733309         song-parth123.mp3
        13521           server-fifo
        166             mserver.config
        25928           server.o
        43216           server
        13312           client.o
        35728           client
        12520           media_transfer.o
        18848           parser.o
        0               server-fifo-2.script

04/08/21 22:50:53.030003127 UTC: File Listing Received
04/08/21 22:50:23.290318008 UTC: TX: get song.mp3
04/08/21 22:50:23.290318008 UTC: Sent Command: get song.mp3

04/08/21 22:51:25.446816265 UTC: Header Response Received
04/08/21 22:51:25.446851033 UTC: Status:100  Host:10.234.136.55:3000  Length:9733309  Type:mp3
04/08/21 22:51:25.446851033 UTC: Name of the file to put data received from server to: parth-song-
fifo.mp3
04/08/21 22:51:36.112922662 UTC: Media Received and Downloaded
04/08/21 22:50:23.290318008 UTC: TX: exit
[pakpatel@in-csci-rrpc04 Project-3]$ exit
```

```
exit

Script done on Thu 08 Apr 2021 06:52:03 PM EDT
Script started on Thu 08 Apr 2021 06:43:23 PM EDT
[johjbake@tesla Project-3]$ ./client in-csci-rrpc01:1234
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:43:40.788921475 UTC: TX: list
04/08/21 22:43:40.788921475 UTC: Sent Command: list

04/08/21 22:43:44.089239066 UTC: Header Response Received
04/08/21 22:43:44.089264102 UTC: Status:100  Host:10.234.136.55:1234  Length:1167  Type:Text
        Size            Name
        4096            .
        50              ..
        4096            .git
        473             .gitignore
        39              clientrc
        1125            media_transfer.h
        113             send.txt
        9733309         song.mp3
        445             Makefile
        2353            client.c
        2706            media_transfer.c
        8911            old.c
        2615            parser.h
        1961            queue.h
        9733309         r.mp3
        9733309         received.mp3
        9114            parser.c
        17355           ansi2html.sh
        20596           client.html
        34304           client.script
        9733309         get receive.mp3
        22157           server-exe.html
        8192            server.script
        9733309         test.mp3
        12520           media_transfer.o
        18848           parser.o
        25928           server.o
        43216           server
        13312           client.o
        35728           client
        12479           server.c
        5352            dax-tesla.script
        9733309         receive-dax.mp3
        194             typescript
        9097            server-load.script
        0               dax-tesla-2.script
        12479           server-copy.c
        20596           caleb-client.html
        168             caleb-mserver.config
        3956            caleb-rrpc03.script
        43216           caleb-server
        9733309         caleb-song.mp3
        20596           caleb-tesla-client.html
        9733309         caleb-tesla.mp3
        6991            caleb-tesla.script
        9733309         caleb-test.mp3
        8911            old-parth.c
        2178            parth-rrpc
        1927            parth-rrpc2
        113             parth-send.txt
        3494            parth-tesla
        2550            parth-tesla2
        9733309         parth-test.mp3
        9733309         receive-parth.mp3
        9733309         song-parth123.mp3
        166             mserver.config
        0               server-fifo

04/08/21 22:43:44.090787648 UTC: File Listing Received
04/08/21 22:43:40.788921475 UTC: TX: get server-load.script
04/08/21 22:43:40.788921475 UTC: Sent Command: get server-load.script

04/08/21 22:44:05.961310569 UTC: Header Response Received
04/08/21 22:44:05.961326739 UTC: Status:100  Host:10.234.136.55:1234  Length:9097  Type:script
04/08/21 22:44:05.961326739 UTC: Name of the file to put data received from server to: johntestfifo
04/08/21 22:44:12.339113544 UTC: Media Received and Downloaded
04/08/21 22:43:40.788921475 UTC: TX: list
04/08/21 22:43:40.788921475 UTC: Sent Command: list

04/08/21 22:44:15.856823413 UTC: Header Response Received
```

```
04/08/21 22:44:15.856839241 UTC: Status:100  Host:10.234.136.55:1234  Length:1167  Type:Text
        Size            Name
        4096            .
        50              ..
        4096            .git
        473             .gitignore
        39              clientrc
        1125            media_transfer.h
        113             send.txt
        9733309         song.mp3
        445             Makefile
        2353            client.c
        2706            media_transfer.c
        8911            old.c
        2615            parser.h
        1961            queue.h
        9733309         r.mp3
        9733309         received.mp3
        9114            parser.c
        17355           ansi2html.sh
        20596           client.html
        34304           client.script
        9733309         get receive.mp3
        22157           server-exe.html
        8192            server.script
        9733309         test.mp3
        12520           media_transfer.o
        18848           parser.o
        25928           server.o
        43216           server
        13312           client.o
        35728           client
        12479           server.c
        5352            dax-tesla.script
        9733309         receive-dax.mp3
        194             typescript
        9097            server-load.script
        0               dax-tesla-2.script
        12479           server-copy.c
        20596           caleb-client.html
        168             caleb-mserver.config
        3956            caleb-rrpc03.script
        43216           caleb-server
        9733309         caleb-song.mp3
        20596           caleb-tesla-client.html
        9733309         caleb-tesla.mp3
        6991            caleb-tesla.script
        9733309         caleb-test.mp3
        8911            old-parth.c
        2178            parth-rrpc
        1927            parth-rrpc2
        113             parth-send.txt
        3494            parth-tesla
        2550            parth-tesla2
        9733309         parth-test.mp3
        9733309         receive-parth.mp3
        9733309         song-parth123.mp3
        166             mserver.config
        0               server-fifo

04/08/21 22:44:15.858343979 UTC: File Listing Received
04/08/21 22:43:40.788921475 UTC: TX: get parser.o
04/08/21 22:43:40.788921475 UTC: Sent Command: get parser.o

04/08/21 22:44:22.586144329 UTC: Header Response Received
04/08/21 22:44:22.586161544 UTC: Status:100  Host:10.234.136.55:1234  Length:18848  Type:o
04/08/21 22:44:22.586161544 UTC: Name of the file to put data received from server to: johnfifo2
04/08/21 22:44:26.312384308 UTC: Media Received and Downloaded
04/08/21 22:43:40.788921475 UTC: TX: get old.c
04/08/21 22:43:40.788921475 UTC: Sent Command: get old.c

04/08/21 22:44:34.939147851 UTC: Header Response Received
04/08/21 22:44:34.939164577 UTC: Status:100  Host:10.234.136.55:1234  Length:8911  Type:c
04/08/21 22:44:34.939164577 UTC: Name of the file to put data received from server to: johnfifo3
04/08/21 22:44:42.089232929 UTC: Media Received and Downloaded
04/08/21 22:43:40.788921475 UTC: TX: get song.mp3
04/08/21 22:43:40.788921475 UTC: Sent Command: get song.mp3

04/08/21 22:45:01.008260960 UTC: Header Response Received
04/08/21 22:45:01.008277515 UTC: Status:100  Host:10.234.136.55:1234  Length:9733309  Type:mp3
04/08/21 22:45:01.008277515 UTC: Name of the file to put data received from server to: johnfifo4
```

```
04/08/21 22:45:05.406703533 UTC: Media Received and Downloaded
04/08/21 22:43:40.788921475 UTC: TX: list
04/08/21 22:43:40.788921475 UTC: Sent Command: list

04/08/21 22:45:23.359560304 UTC: Header Response Received
04/08/21 22:45:23.359578192 UTC: Status:100  Host:10.234.136.55:1234  Length:1170  Type:Text
        Size            Name
        4096            .
        50              ..
        4096            .git
        473             .gitignore
        39              clientrc
        1125            media_transfer.h
        113             send.txt
        9733309         song.mp3
        445             Makefile
        2353            client.c
        2706            media_transfer.c
        8911            old.c
        2615            parser.h
        1961            queue.h
        9733309         r.mp3
        9733309         received.mp3
        9114            parser.c
        17355           ansi2html.sh
        20596           client.html
        34304           client.script
        9733309         get receive.mp3
        22157           server-exe.html
        8192            server.script
        9733309         test.mp3
        12520           media_transfer.o
        18848           parser.o
        25928           server.o
        43216           server
        13312           client.o
        35728           client
        12479           server.c
        5352            dax-tesla.script
        9733309         receive-dax.mp3
        194             typescript
        9097            server-load.script
        0               dax-tesla-2.script
        12479           server-copy.c
        20596           caleb-client.html
        168             caleb-mserver.config
        3956            caleb-rrpc03.script
        43216           caleb-server
        9733309         caleb-song.mp3
        20596           caleb-tesla-client.html
        9733309         caleb-tesla.mp3
        6991            caleb-tesla.script
        9733309         caleb-test.mp3
        8911            old-parth.c
        2178            parth-rrpc
        1927            parth-rrpc2
        113             parth-send.txt
        3494            parth-tesla
        2550            parth-tesla2
        9733309         parth-test.mp3
        9733309         receive-parth.mp3
        9733309         song-parth123.mp3
        166             mserver.config
        8192            server-fifo

04/08/21 22:45:23.361245489 UTC: File Listing Received
04/08/21 22:43:40.788921475 UTC: TX: get ver-copy.c
04/08/21 22:43:40.788921475 UTC: Sent Command: get server-copy.c

04/08/21 22:45:30.207880119 UTC: Header Response Received
04/08/21 22:45:30.207895154 UTC: Status:100  Host:10.234.136.55:1234  Length:12479  Type:c
04/08/21 22:45:30.207895154 UTC: Name of the file to put data received from server to: johnfifo4
04/08/21 22:45:32.967094810 UTC: Media Received and Downloaded
04/08/21 22:43:40.788921475 UTC: TX: exit
[johjbake@tesla Project-3]$ ./client in-csci-rrpc01:1234
h_length = 4
Can't connect
[johjbake@tesla Project-3]$ ./client in-csci-rrpc01:1234
h_length = 4
Can't connect
[johjbake@tesla Project-3]$ ./client in-csci-rrpc01:1234
```

```
h_length = 4
Can't connect
[johjbake@tesla Project-3]$ script johnnew
Script started, file is johnnew
[johjbake@tesla Project-3]$ ./client in-csci-rrpc01:3000
h_length = 4
Connected: server's address is in-csci-rrpc01
04/08/21 22:50
```

```
Script started on 2021-04-08 17:50:51-04:00 [TERM="xterm-256color" TTY="/dev/tty1" COLUMNS="148" LINES="32"]
%


  Project-3 git:(exe-traces)    ./client 192.168.0.124:1234

h_length = 4
Connected: server's address is 192.168.0.124
04/08/21 21:51:05.322832700 UTC: TX: list
04/08/21 21:51:05.322832700 UTC: Sent Command: list

04/08/21 21:51:09.347689600 UTC: Header Response Received
04/08/21 21:51:09.347758100 UTC: Status:100  Host:127.0.1.1:1234  Length:494  Type:Text
        Size            Name
        473             .gitignore
        0               typescript
        8196            a.out
        9733309         received.mp3
        4096            .git
        10716           client.o
        113             send.txt
        2706            media_transfer.c
        4096            .
        1125            media_transfer.h
        2353            client.c
        39              clientrc
        4096            ..
        21128           server.o
        17355           ansi2html.sh
        47200           server
        12479           server.c
        2615            parser.h
        8911            old.c
        34432           client
        9114            parser.c
        167             mserver.config
        10332           media_transfer.o
        9733309         r.mp3
        1961            queue.h
        9733309         song.mp3
        4096            .vscode
        15996           parser.o
        445             Makefile

04/08/21 21:51:09.392414800 UTC: File Listing Received
04/08/21 21:51:05.322832700 UTC: TX: get song.mp3
04/08/21 21:51:05.322832700 UTC: Sent Command: get song.mp3

04/08/21 21:52:17.021055600 UTC: Header Response Received
04/08/21 21:52:17.021092900 UTC: Status:100  Host:127.0.1.1:1234  Length:9733309  Type:mp3
04/08/21 21:52:17.021092900 UTC: Name of the file to put data received from server to: test.mp3
04/08/21 21:52:27.074399900 UTC: Media Received and Downloaded
04/08/21 21:51:05.322832700 UTC: TX: exit
%


  Project-3 git:(exe-traces)    mv typescript client.script

%
```

```
Script started on 2021-04-08 17:50:35-04:00 [TERM="xterm-256color" TTY="/dev/pts/3" COLUMNS="195" LINES="47"]
pi@raspberrypi:~/repos/ece40800/Project-3 $ make
gcc -g   -c -o media_transfer.o media_transfer.c
gcc -g -c media_transfer.c
gcc -g   -c -o parser.o parser.c
gcc -g -c parser.c
gcc -g   -c -o server.o server.c
gcc -g -o server server.o media_transfer.o parser.o -lm -lnsl -lpthread
gcc -g   -c -o client.o client.c
gcc -g -o client client.o media_transfer.o parser.o -lm -lnsl
pi@raspberrypi:~/repos/ece40800/Project-3 $ ./server
*******Server configuration*******
Port Number: 1234
Num Threads: 4
Max Reqs: 10
Media Path: /home/pi/repos/ece40800/Project-3
**********************************

04/08/21 21:51:05.224235906 UTC: Main: Accepting New Connection: 5
04/08/21 21:51:05.224235906 UTC: Main: Adding New Client to the Job queue...
04/08/21 21:51:05.228580789 UTC: Watch Request: Thead 3066995808: Handling client 5
04/08/21 21:51:05.228665732 UTC: Handle Request: Client IP: 192.168.0.124 Ephemeral Port: 1234
04/08/21 21:51:05.228577159 UTC: Main: Added New Client to the Job queue
04/08/21 21:51:09.244611698 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: list

04/08/21 21:52:11.181261627 UTC: Main: Accepting New Connection: 6
04/08/21 21:52:11.181261627 UTC: Main: Adding New Client to the Job queue...
04/08/21 21:52:11.181328792 UTC: Main: Added New Client to the Job queue
04/08/21 21:52:11.181336551 UTC: Watch Request: Thead 3058603104: Handling client 6
04/08/21 21:52:11.181363477 UTC: Handle Request: Client IP: 192.168.0.124 Ephemeral Port: 1234
04/08/21 21:52:12.687719411 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: list
04/08/21 21:52:16.919490219 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: get song.mp3
04/08/21 21:52:16.919685789 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent Header Information
04/08/21 21:52:26.827175881 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent: get song.mp3
04/08/21 21:52:31.600172260 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: get r.mp3
04/08/21 21:52:31.600406108 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent Header Information
04/08/21 21:52:36.766682435 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Sent: get r.mp3
04/08/21 21:52:38.351151265 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: exit
04/08/21 21:52:38.351313521 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Closed connection with client: 6
04/08/21 21:52:40.313385603 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Command Recevied string: exit
04/08/21 21:52:40.313656449 UTC: Handle_Request: Client IP: 192.168.0.124 Ephemeral Port: 1234 : Closed connection with client: 5
^C
pi@raspberrypi:~/repos/ece40800/Project-3 $ ls
 ansi2html.sh   client     client.o  'get receive.mp3'   media_transfer.c   media_transfer.o   old.c      parser.h   queue.h        r.mp3      server     server.o   typescript
 a.out          client.c   clientrc   Makefile           media_transfer.h   mserver.config     parser.c   parser.o   received.mp3   send.txt   server.c   song.mp3
pi@raspberrypi:~/repos/ece40800/Project-3 $ ans2
```