Jarred Parr -- October 11 2018

Client.c

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

#define MAXDATASIZE 4096

int get_port();
in_addr_t get_host();
char* input_handler();

int get_port() {
  printf("Enter the port number to run on: ");
  char port[10] = "3000";
  fgets(port, 10, stdin);

  return atoi(port);
}

in_addr_t get_host() {
  printf("Enter the host to connect to: ");
  char ip[100] = "127.0.0.1";
  fgets(ip, 100, stdin);
  return inet_addr(ip);
}

char* input_handler() {
  char* message = (char*) malloc(sizeof(char));
  printf("> ");
  fgets(message, MAXDATASIZE, stdin);

  return message;
}

int main(int argc, char** argv) {
  int sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0) {
    fprintf(stderr, "There was an error creating the socket!\n");
    return EXIT_FAILURE;
  }

  int port = get_port();
```

```
in_addr_t host = get_host();

struct sockaddr_in server;
server.sin_family = AF_INET;
server.sin_port = htons(port);
server.sin_addr.s_addr = host;

int c = connect(sockfd, (struct sockaddr*) &server, sizeof(server));

if (c == -1) {
  fprintf(stderr, "There was an error connecting to the server!\n");
  return EXIT_FAILURE;
}

struct timeval timeout;
timeout.tv_sec = 0;
timeout.tv_usec = 0;

char response[MAXDATASIZE];
char* input;

while(1) {
  fd_set read_fd;
  FD_ZERO(&read_fd);

  int fdmax = sockfd;
  FD_SET(sockfd, &read_fd);
  FD_SET(STDIN_FILENO, &read_fd);

  if(select(fdmax + 1, &read_fd, NULL, NULL, &timeout) == -1) {
    fprintf(stderr, "Unable to modify the file descriptor\n");
  }

  if (FD_ISSET(sockfd, &read_fd)) {
    recv(sockfd, response, MAXDATASIZE, 0);

    if (strcmp("Quit\n", response) == 0) {
      printf("Server ended the connection");
      break;
    } else {
      printf("< %s\n", response);
    }
  }

  if (FD_ISSET(STDIN_FILENO, &read_fd)) {
    input = input_handler();
    int s = send(sockfd, input, strlen(input), 0);
    if (s < 0) {
      fprintf(stderr, "Failed to send message, sorry bruh");
    }

    if (strcmp("Quit\n", input) == 0) {
      printf("You ended the connection");
```

```
        break;
      }
    }
  }

  close(sockfd);

  return EXIT_SUCCESS;
}
```

Server.c

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

#define MAXDATASIZE 4096

int get_port();

int get_port() {
  printf("Enter the port number to run on: ");
  char port[10] = "3000";
  fgets(port, 10, stdin);

  return atoi(port);
}

char* input_handler() {
  char* message = (char*) malloc(sizeof(char));
  printf("> ");
  fgets(message, MAXDATASIZE, stdin);

  return message;
}

int main(int argc, char** argv) {
  int sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0) {
    fprintf(stderr, "There was an error creating the socket!\n");
    return EXIT_FAILURE;
  }

  int port = get_port();
  printf("Server listening on localhost:%d\n", port);
```

```c
struct sockaddr_in server, client;
server.sin_family = AF_INET;
server.sin_port = htons(port);
server.sin_addr.s_addr = INADDR_ANY;

if (bind(sockfd, (struct sockaddr*) &server, sizeof(server)) == -1) {
  fprintf(stderr, "LOUD SCREAMING NOISES AHHHHHHHHHHHHHHHHHHHHHHHHHHHHH");
  return EXIT_FAILURE;
}

listen(sockfd, 10);

struct timeval timeout;
timeout.tv_sec = 0;
timeout.tv_usec = 0;

int clientfd = -1;
char response[MAXDATASIZE];
char* input;

socklen_t sin_size = sizeof client;

while (1) {
  fd_set read_fd;
  FD_ZERO(&read_fd);

  int fdmax = sockfd;
  FD_SET(sockfd, &read_fd);
  FD_SET(STDIN_FILENO, &read_fd);

  if (clientfd > -1) {
    FD_SET(clientfd, &read_fd);
  }

  if (select(fdmax + 2, &read_fd, NULL, NULL, &timeout) == -1) {
    printf("Unable to modify sockfd\n");
  }

  if (FD_ISSET(sockfd, &read_fd)) {
    clientfd = accept(sockfd, (struct sockaddr*) &client, &sin_size);
    printf("New connection\n");
  }

  if (clientfd > -1 && FD_ISSET(clientfd, &read_fd)) {
    recv(clientfd, response, MAXDATASIZE, 0);

    if (strcmp("Quit\n", response) == 0) {
      printf("Client ended the connection");
      break;
    } else {
      printf("< %s\n", response);
    }
```

```
    }

    if (FD_ISSET(STDIN_FILENO, &read_fd)) {
      input = input_handler();
      send(clientfd, input, strlen(input), 0);

      if (strcmp("Quit\n", input) == 0) {
        break;
      }
      free(input);
    }
  }

  close(clientfd);
  close(sockfd);

  return EXIT_SUCCESS;

}
```