Team_4

TCP Echo Server and Client

Team members

- Luming Xu: `client.c`, `test.sh`
- Akhilesh Rawat: `server.c`, `Makefile`

Usage

`SERVER`: `./echos PORT`

`CLIENT`: `./echo IPAddr PORT`

Architecture

The server runs on `localhost` and can connect to `BACKLOG` number of clients at a time. It echoes the messages sent to it by the clients respectively. The server serves every new connection request that comes to it on `PORT` number mentioned when started, in a new process.

The server responds with `TCP FIN received` when a client a diconnects itself from the server.

The function `sigchild_handler` has been taken verbatim from Beej's guide. For other system calls, which are not automatially restarted when interrupted by a signal handler, we are manually restarting the call (Reference).

Source Code

Client Code

```
#include "headers.h"

#define MAXDATASIZE 100

void *get_in_addr(struct sockaddr *sa) {
    if (sa->sa_family == AF_INET) {
        return &(((struct sockaddr_in *) sa)->sin_addr);
    }
    return &(((struct sockaddr_in6 *) sa)->sin6_addr);
}

int writen(int sockfd, char *buf) {
    int numbytes;
    while ((numbytes = send(sockfd, buf, MAXDATASIZE - 1, 0)) == -1 && errno ==
EINTR) {
        // manually restarting
        continue;
    }
```

```
    return numbytes;
}

int readline(int sockfd, char *recvbuf) {
    int numbytes;
    while ((numbytes = recv(sockfd, recvbuf, MAXDATASIZE - 1, 0)) == -1 && errno
== EINTR) {
        // manually restarting
    }
    return numbytes;
}

int server_lookup_connect(char *host, char *server_port) {
    struct addrinfo hints, *server_info, *p;
    int status;
    int sock_fd;

    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    // hints.ai_flags = AI_PASSIVE;     // fill in my IP

    // argv[1]: IPAdr
    // argv[2]: Port
    if ((status = getaddrinfo(host, server_port, &hints, &server_info)) != 0) {
        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
        return 2;
    }

    for (p = server_info; p != NULL; p = p->ai_next) { //loop through link list
        sock_fd = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
        if (sock_fd == -1) { //socket creation failed
            perror("client: socket");
            continue;
        }

        if (connect(sock_fd, p->ai_addr, p->ai_addrlen) == -1) { //connection
failed
            close(sock_fd);
            perror("client: connect");
            continue;
        }
        break;
    }
    if (p == NULL) {
        fprintf(stderr, "client: failed to connect\n");
        return 2;
    }

    printf("client: connected to %s:%s\n", host, server_port);

    freeaddrinfo(server_info);
    return sock_fd;
```

```
}

int main(int argc, char *argv[]) {

    int sock_fd;
    int numbytes_send, numbytes_recv;
    char buf[MAXDATASIZE], recvbuf[MAXDATASIZE];
    char *host, *server_port;

    if (argc == 3) {
        host = argv[1];
        server_port = argv[2];
        printf("Akhilesh Rawat and Luming Xu Team 4 Echo Service\n");
        printf("client: client started\n");
    } else {
        fprintf(stderr, "usage: echo IPAdr Port\n");
        exit(1);
    }

    sock_fd = server_lookup_connect(host, server_port);
    // reading off the stdin
    while (fgets(buf, MAXDATASIZE, stdin)) {
        numbytes_send = writen(sock_fd, buf);
        if (numbytes_send == -1) { //writen and error handling
            perror("send");
            exit(1);
        }
        numbytes_recv = readline(sock_fd, recvbuf); //numbytes received
        if (numbytes_recv == -1) {//readline and error handling
            perror("recv");
            exit(1);
        }

        //successful write and receive echo
        printf("%s\n", recvbuf);
        int comparison;

        comparison = strcmp(buf, recvbuf);
        if (strcmp(buf, recvbuf) != 0) { //buffer not matching
            if (strncmp(buf, recvbuf, numbytes_recv) == 0) { // recv string is a
substring of send string
                printf("    !recv buffer out of space.\n");
            } else { //send string does not match recv string
                perror("    !client: recv string did not match");
                exit(1);
            }
        }
    }

    printf("Closing the socket \n"); // buf[numbytes] = '\0';
    close(sock_fd);
    return 0;
}
```

Server Code

```
#include "headers.h"

#define BACKLOG 10
#define MAXDATASIZE 100

// function taken from beej's guide
void sigchild_handler(int s) {
    int saved_errno = errno;

    while(waitpid(-1, NULL, WNOHANG) > 0);

    errno = saved_errno;
}

void *get_in_addr(struct sockaddr *sa) {
    if (sa->sa_family == AF_INET) {
        return &(((struct sockaddr_in*) sa)->sin_addr);
    }
    return &(((struct sockaddr_in6*)sa)->sin6_addr);
}

int server_read(int new_fd, char * buf) {
    // read the received buffer from the socket
    return recv(new_fd, buf, MAXDATASIZE-1, 0);
}

int server_write(int new_fd, char * buf) {
    // send the buffer to the socket
    return send(new_fd, buf, MAXDATASIZE-1, 0);
}

int main(int argc, char *argv[]) {

    int sockfd, new_fd;
    struct addrinfo hints, *res, *p;
    struct sockaddr_storage their_addr;           // connector's address
information
    socklen_t addr_size;
    int yes = 1;
    struct sigaction sa;
    int status, numbytes;
    char buf[MAXDATASIZE];

    if (argc != 2) {
        // check for correct usage
        fprintf(stderr, "usage: echos Port\n");
        exit(1);
    }
```

```
    memset(&hints, 0, sizeof(hints));

    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;     // fill in my IP

    if ((status = getaddrinfo(NULL, argv[1] ,&hints, &res)) != 0) {
        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
        return 1;
    }

    // loop through all the results and bind to the first correct
    for (p = res; p != NULL; p = p->ai_next) {
        if ((sockfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol)) == -1)
{
           perror("server: socket");
           continue;
        }

        // allow other sockets to bind to this port
        if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes))) {
           perror("setsocketopt");
           exit(1);
        }

        if (bind(sockfd, p->ai_addr, p->ai_addrlen) == -1)  {
            close(sockfd);
            perror("server: bind");
            continue;
        }

        break;
    }

    // we don't need it now
    freeaddrinfo(res);

    if (p == NULL) {
        fprintf (stderr, "server: failed to bind\n");
        exit(1);
    }

    if (listen(sockfd, BACKLOG) == -1) {
        perror("listen");
        exit(1);
    }

    // reap all dead process - function taken from beej's guide
    sa.sa_handler = sigchild_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_RESTART;
    if (sigaction(SIGCHLD, &sa, NULL) == -1) {
        perror("sigaction");
        exit(1);
```

```
    }

    printf("server: waiting for connections....\n");

    int sin_size = sizeof(their_addr);
    char str[sin_size];
    while(1) {
        while ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr,
&sin_size)) == -1 && errno == EINTR) {
            // manually restart
            continue;
        }

        if (new_fd == -1) {
            perror("accept");
            continue;
        }

        inet_ntop(their_addr.ss_family, get_in_addr((struct sockaddr
*)&their_addr), str, sin_size);
        printf("server: got conection from %s\n", str);

        if (!fork()) {
            while (numbytes =  server_read(new_fd, buf)) {
                if  (numbytes == -1 && errno == EINTR) {
                    continue;
                }

                if (numbytes == -1) {
                    perror("recv");
                    exit(1);
                }

                if (numbytes == 0) {
                    printf("TCP FIN received");
                }

                printf("recv: %s", buf);

                // echo back
                while ((numbytes = server_write(new_fd, buf) == -1 && errno ==
EINTR)) {
                    continue;
                }

                if (numbytes == -1) {
                    perror("send");
                    exit(1);
                }

                printf("send: %s\n", buf);
            }
            // client disconnected
            printf("TCP FIN received");
```

```
            fflush(stdout);
            // let it go
            close(new_fd);
        }
    }
    return 0;
}
```

Header

```
#ifndef HEADERS_H_
#define HEADERS_H_

#include<stdio.h>
#include<errno.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<sys/wait.h>
#include<stdlib.h>
#include<string.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<netdb.h>
#include<sys/types.h>
#include<signal.h>

#endif
```

Makefile

```
C = gcc
DEPS = headers.h

TARGETS = echos echo

all: $(TARGETS)

echos : server.o $(DEPS)
        $(C) -o echos server.o
echo : client.o $(DEPS)
        $(C) -o echo client.o

clean :
        rm -f *.o *~ echos echo
```

Test Cases

Case 1: line of text terminated by a newline

```
Terminal
File Edit View Search Terminal Help
Akhilesh Rawat and Luming Xu Team 4 Echo Service
server: waiting for connections....
server: got conection from 127.0.0.1
recv: line of text terminated by a newline
send: line of text terminated by a newline
```

```
Terminal
File Edit View Search Terminal Help
Akhilesh Rawat and Luming Xu Team 4 Echo Service
client: client started
client: connected to localhost:12346
line of text terminated by a newline
line of text terminated by a newline
```

Case 2: line of text exceeding the maximum line length

Input exceeding maximum buffer length are segmented into several packets.

Server Side Capture

```
Terminal
File Edit View Search Terminal Help
Akhilesh Rawat and Luming Xu Team 4 Echo Service
server: waiting for connections....
server: got conection from 127.0.0.1
recv: Lorem Ipsum is simply dummy text of the printing and typesetting industry.
 Lorem Ipsum has been thesend: Lorem Ipsum is simply dummy text of the printing
and typesetting industry. Lorem Ipsum has been the
recv:  industry's standard dummy text ever since the 1500s, when an unknown prin
ter took a galley of typesend:  industry's standard dummy text ever since the 15
00s, when an unknown printer took a galley of type
recv:  and scrambled it to make a type specimen book. It has survived not only f
ive centuries, but also tsend:  and scrambled it to make a type specimen book. I
t has survived not only five centuries, but also t
recv: he leap into electronic typesetting, remaining essentially unchanged. It w
as popularised in the 196send: he leap into electronic typesetting, remaining es
sentially unchanged. It was popularised in the 196
recv: 0s with the release of Letraset sheets containing Lorem Ipsum passages, an
d more recently with desksend: 0s with the release of Letraset sheets containing
 Lorem Ipsum passages, and more recently with desk
recv: top publishing software like Aldus PageMaker including versions of Lorem I
psum.
send: top publishing software like Aldus PageMaker including versions of Lorem I
psum.
```

Client Side Capture

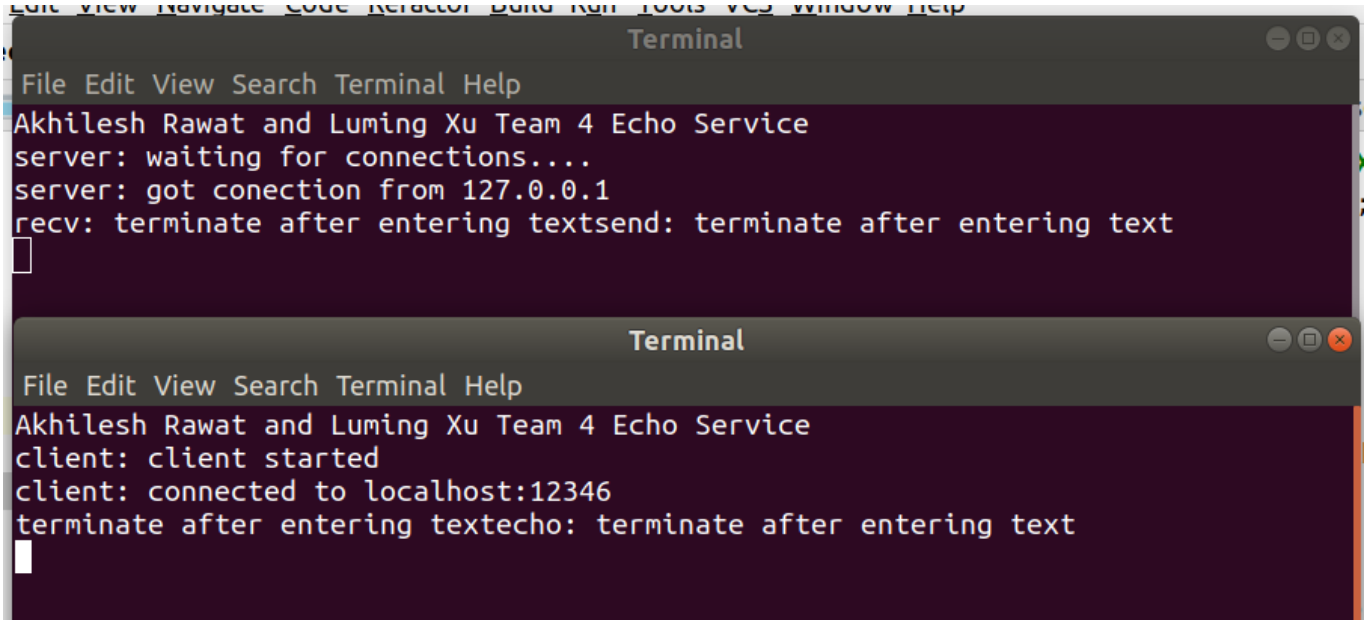
```
Terminal
File Edit View Search Terminal Help
 unknown printer took a galley of type and scrambled it to make a type specimen
book. It has survived not only five centuries, but also the leap into electronic
 typesetting, remaining essentially unchanged. It was popularised in the 1960s w
ith the release of Letraset sheets containing Lorem Ipsum passages, and more rec
ently with desktop publishing software like Aldus PageMaker including versions o
f Lorem Ipsum.
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
 Ipsum has been the)
    !recv buffer out of space.
 industry's standard dummy text ever since the 1500s, when an unknown printer to
ok a galley of type)
    !recv buffer out of space.
 and scrambled it to make a type specimen book. It has survived not only five ce
nturies, but also t)
    !recv buffer out of space.
he leap into electronic typesetting, remaining essentially unchanged. It was pop
ularised in the 196)
    !recv buffer out of space.
0s with the release of Letraset sheets containing Lorem Ipsum passages, and more
 recently with desk)
    !recv buffer out of space.
top publishing software like Aldus PageMaker including versions of Lorem Ipsum.
```

Case 3: line with no characters and EOF

```
Terminal
File Edit View Search Terminal Help
Akhilesh Rawat and Luming Xu Team 4 Echo Service
server: waiting for connections....
server: got conection from 127.0.0.1
TCP FIN received

andre@andre-VirtualBox: ~/CLionProjects/ECEN602/echo-server/bash
File Edit View Search Terminal Help
Akhilesh Rawat and Luming Xu Team 4 Echo Service
client: client started
client: connected to localhost:12346
Closing the socket
andre@andre-VirtualBox:~/CLionProjects/ECEN602/echo-server/bash$
```

Case 4: client terminated after entering text



Case 5: three clients conected to the server