TCP Client Program in C

1

TCP

2

2.1

```
#include <stdio.h>
#include <unistd.h>
#include <strings.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <errno.h>
#include <signal.h>

#define BUFFER_SIZE BUFSIZ
```

`sys/socket.h`, `netdb.h` `BUFFER_SIZE`

2.2 write_all

```
ssize_t write_all(int sockfd, const void *buf, size_t len) {
    size_t total_written = 0;
    while (total_written < len) {
        ssize_t written = write(sockfd, buf + total_written, len - total_writ
        if (written <= 0) {
            if (errno == EINTR) continue; //
            perror(”write␣error”);
            return -1;
        }
        total_written += written;
    }
    return total_written;
```

```
}
```

`write_all`

2.3 main

```
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage:␣%s␣<hostname>␣<port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    const char *hostname = argv[1];
    const char *port = argv[2];

    int sockfd;
    struct addrinfo hints, *res, *p;
    signal(SIGPIPE, SIG_IGN);

    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET;          // IPv4
    hints.ai_socktype = SOCK_STREAM;  // TCP

    if (getaddrinfo(hostname, port, &hints, &res) != 0) {
        perror("getaddrinfo");
        exit(EXIT_FAILURE);
    }

    for (p = res; p != NULL; p = p->ai_next) {
        sockfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
        if (sockfd < 0) continue;

        if (connect(sockfd, p->ai_addr, p->ai_addrlen) == 0) {
            break; //
        }
        close(sockfd);
    }

    if (p == NULL) {
        fprintf(stderr, "Failed␣to␣connect␣to␣server\n");
        freeaddrinfo(res);
        exit(EXIT_FAILURE);
    }

    freeaddrinfo(res);
    printf("Connected␣to␣%s␣on␣port␣%s\n", hostname, port);
```

`getaddrinfo` `sockfd`

2.4

```
pid_t pid = fork();
if (pid < 0) {
    perror("fork␣failed");
    close(sockfd);
    exit(EXIT_FAILURE);
}

if (pid == 0) { //    :
    char buf[BUFFER_SIZE];
    while (1) {
        memset(buf, 0, BUFFER_SIZE);
        ssize_t nbytes = read(sockfd, buf, BUFFER_SIZE - 1);

        if (nbytes < 0) {
            perror("read␣error");
            break;
        } else if (nbytes == 0) {
            printf("Server␣disconnected.\n");
            break;
        }

        buf[nbytes] = '\0'; //  NULL
        fputs(buf, stdout);
    }
    close(sockfd);
    exit(0);
} else { //    :
    char mesg[BUFFER_SIZE];
    while (1) {
        memset(mesg, 0, BUFFER_SIZE);
        if (fgets(mesg, BUFFER_SIZE, stdin) == NULL) {
            printf("EOF␣detected.␣Closing␣connection...\n");
            break;
        }

        mesg[strcspn(mesg, "\n")] = '\0';

        if (write_all(sockfd, mesg, strlen(mesg)) < 0) {
            break;
        }
    }

    kill(pid, SIGKILL);
    close(sockfd);
```

```
    }

    return 0;
}
```

3

C TCP