

加微信：642945106 发送“赠送”领取赠送精品课程



发数字“2”获取众筹列表

下载APP



期中大作业 | 题目以及解答剖析

2019-09-20 盛延敏

网络编程实战

[进入课程 >](#)



讲述：冯永吉

时长 00:22 大小 356.18K



你好，今天是期中大作业讲解课。诚如一位同学所言，这次的大作业不是在考察网络编程的细节，而是在考如何使用系统 API 完成 cd、pwd、ls 等功能。不过呢，网络编程的框架总归还是要掌握的。

我研读了大部分同学的代码，基本上是做得不错的，美中不足的是能动手完成代码编写和调试的同学偏少。我还是秉持一贯的看法，计算机程序设计是一门实战性很强的学科，如果只是单纯地听讲解，没有自己动手这一环，对知识的掌握总归还是差那么点意思。

代码我已经 push 到[这里](#)，你可以点进链接看一下。

客户端程序

废话少说，我贴下我的客户端程序：

 [复制代码](#)

```
1 #include "lib/common.h"
2 #define MAXLINE 1024
3
4 int main(int argc, char **argv) {
5     if (argc != 3) {
6         error(1, 0, "usage: tcp_client <IPaddress> <port>");
7     }
8     int port = atoi(argv[2]);
9     int socket_fd = tcp_client(argv[1], port);
10
11     char recv_line[MAXLINE], send_line[MAXLINE];
12     int n;
13
14     fd_set readmask;
15     fd_set allreads;
16     FD_ZERO(&allreads);
17     FD_SET(0, &allreads);
18     FD_SET(socket_fd, &allreads);
19
20     for (;;) {
21         readmask = allreads;
22         int rc = select(socket_fd + 1, &readmask, NULL, NULL, NULL);
23
24         if (rc <= 0) {
25             error(1, errno, "select failed");
26         }
27
28         if (FD_ISSET(socket_fd, &readmask)) {
29             n = read(socket_fd, recv_line, MAXLINE);
30             if (n < 0) {
31                 error(1, errno, "read error");
32             } else if (n == 0) {
33                 printf("server closed \n");
34                 break;
35             }
36             recv_line[n] = 0;
37             fputs(recv_line, stdout);
38             fputs("\n", stdout);
39         }
40
41         if (FD_ISSET(STDIN_FILENO, &readmask)) {
42             if (fgets(send_line, MAXLINE, stdin) != NULL) {
43                 int i = strlen(send_line);
44                 if (send_line[i - 1] == '\n') {
45                     send_line[i - 1] = 0;
46                 }
47             }
48         }
49     }
50 }
```

```
48     if (strncmp(send_line, "quit", strlen(send_line)) == 0) {
49         if (shutdown(socket_fd, 1)) {
50             error(1, errno, "shutdown failed");
51         }
52     }
53
54     size_t rt = write(socket_fd, send_line, strlen(send_line));
55     if (rt < 0) {
56         error(1, errno, "write failed ");
57     }
58 }
59 }
60 }
61
62 exit(0);
63 }
```

◀ ▶

客户端的代码主要考虑的是使用 select 同时处理标准输入和套接字，我看到有同学使用 fgets 来循环等待用户输入，然后再把输入的命令通过套接字发送出去，当然也是可以正常工作的，只不过不能及时响应来自服务端的命令结果，所以，我还是推荐使用 select 来同时处理标准输入和套接字。

这里 select 如果发现标准输入有事件，读出标准输入的字符，就会通过调用 write 方法发送出去。如果发现输入的是 quit，则调用 shutdown 方法关闭连接的一端。

如果 select 发现套接字流有可读事件，则从套接字中读出数据，并把数据打印到标准输出上；如果读到了 EOF，表示该客户端需要退出，直接退出循环，通过调用 exit 来完成进程的退出。

服务器端程序

下面是我写的服务器端程序：

 复制代码

```
1 #include "lib/common.h"
2 static int count;
3
4 static void sig_int(int signo) {
5     printf("\nreceived %d datagrams\n", count);
6     exit(0);
7 }
```

```
8
9 char *run_cmd(char *cmd) {
10     char *data = malloc(16384);
11     bzero(data, sizeof(data));
12     FILE *fdp;
13     const int max_buffer = 256;
14     char buffer[max_buffer];
15     fdp = popen(cmd, "r");
16     char *data_index = data;
17     if (fdp) {
18         while (!feof(fdp)) {
19             if (fgets(buffer, max_buffer, fdp) != NULL) {
20                 int len = strlen(buffer);
21                 memcpy(data_index, buffer, len);
22                 data_index += len;
23             }
24         }
25         pclose(fdp);
26     }
27     return data;
28 }
29
30 int main(int argc, char **argv) {
31     int listenfd;
32     listenfd = socket(AF_INET, SOCK_STREAM, 0);
33
34     struct sockaddr_in server_addr;
35     bzero(&server_addr, sizeof(server_addr));
36     server_addr.sin_family = AF_INET;
37     server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
38     server_addr.sin_port = htons(SERV_PORT);
39
40     int on = 1;
41     setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
42
43     int rt1 = bind(listenfd, (struct sockaddr *) &server_addr, sizeof(server_addr));
44     if (rt1 < 0) {
45         error(1, errno, "bind failed ");
46     }
47
48     int rt2 = listen(listenfd, LISTENQ);
49     if (rt2 < 0) {
50         error(1, errno, "listen failed ");
51     }
52
53     signal(SIGPIPE, SIG_IGN);
54
55     int connfd;
56     struct sockaddr_in client_addr;
57     socklen_t client_len = sizeof(client_addr);
58
59
```

```
60     char buf[256];
61     count = 0;
62
63     while (1) {
64         if ((connfd = accept(listenfd, (struct sockaddr *) &client_addr, &client_len)) < 0)
65             error(1, errno, "bind failed ");
66     }
67
68     while (1) {
69         bzero(buf, sizeof(buf));
70         int n = read(connfd, buf, sizeof(buf));
71         if (n < 0) {
72             error(1, errno, "error read message");
73         } else if (n == 0) {
74             printf("client closed \n");
75             close(connfd);
76             break;
77         }
78         count++;
79         buf[n] = 0;
80         if (strncmp(buf, "ls", n) == 0) {
81             char *result = run_cmd("ls");
82             if (send(connfd, result, strlen(result), 0) < 0)
83                 return 1;
84         } else if (strncmp(buf, "pwd", n) == 0) {
85             char buf[256];
86             char *result = getcwd(buf, 256);
87             if (send(connfd, result, strlen(result), 0) < 0){
88                 return 1;
89             }
90             free(result);
91         } else if (strncmp(buf, "cd ", 3) == 0) {
92             char target[256];
93             bzero(target, sizeof(target));
94             memcpy(target, buf + 3, strlen(buf) - 3);
95             if (chdir(target) == -1) {
96                 printf("change dir failed, %s\n", target);
97             }
98         } else {
99             char *error = "error: unknown input type";
100             if (send(connfd, error, strlen(error), 0) < 0)
101                 return 1;
102         }
103     }
104 }
105 exit(0);
106
107 }
```

服务器端程序需要两层循环，第一层循环控制多个客户端连接，当然咱们这里没有考虑使用并发，这在第三个模块中会讲到。严格来说，现在的服务器端程序每次只能服务一个客户连接。

第二层循环控制和单个连接的数据交互，因为我们不止完成一次命令交互的过程，所以这一层循环也是必须的。

大部分同学都完成了这个两层循环的设计，我觉得非常棒。

在第一层循环里通过 `accept` 完成了连接的建立，获得连接套接字。

在第二层循环里，先通过调用 `read` 函数从套接字获取字节流。我这里处理的方式是反复使用了 `buf` 缓冲，每次使用之前记得都要调用 `bzero` 完成初始化，以便重复利用。

如果读取数据为 0，则说明客户端尝试关闭连接，这种情况下，需要跳出第二层循环，进入 `accept` 阻塞调用，等待新的客户连接到来。我看到有同学使用了 `goto` 来完成跳转，其实使用 `break` 跳出就可以了，也有同学忘记跳转了，这里需要再仔细看一下。

在读出客户端的命令之后，就进入处理环节。通过字符串比较命令，进入不同的处理分支。C 语言的 `strcmp` 或者 `strncpy` 可以帮助我们进行字符串比较，这个比较类似于 Java 语言的 `String equalsIgnoreCase` 方法。当然，如果命令的格式有错，需要我们把错误信息通过套接字传给客户端。

对于 “`pwd`” 命令，我是通过调用 `getcwd` 来完成的，`getcwd` 是一个 C 语言的 API，可以获得当前的路径。

对于 “`cd`” 命令，我是通过调用 `chdir` 来完成的，`cd` 是一个 C 语言的 API，可以将当前目录切换到指定的路径。有的同学在这里还判断支持了 “`cd ~`”，回到了当前用户的 `HOME` 路径，这个非常棒，我就没有考虑这种情况了。

对于 “`ls`” 命令，我看到有同学是调用了 `scandir` 方法，获得当前路径下的所有文件列表，再根据每个文件类型，进行了格式化的输出。这个方法非常的棒，是一个标准实现。我这里呢，为了显得稍微不一样，通过了 `popen` 的方法，执行了 `ls` 的 `bash` 命令，把 `bash` 命令的结果通过文件字节流的方式读出，再将该字节流通过套接字传给客户端。我看到有的同学在自己的程序里也是这么做的。

这次的期中大作业，主要考察了客户端 - 服务器编程的基础知识。

客户端程序考察使用 `select` 多路复用，一方面从标准输入接收字节流，另一方面通过套接字读写，以及使用 `shutdown` 关闭半连接的能力。

服务器端程序则考察套接字读写的能力，以及对端连接关闭情况下的异常处理等能力。

不过，服务器端程序目前只能一次服务一个客户端连接，不具备并发服务的能力。如何编写一个具备高并发服务能力的服务器端程序，将是我们接下来课程的重点。我们将会重点讲述基于 I/O 多路复用的事件驱动模型，并以此为基础设计一个高并发网络编程框架，通过这个框架，实现一个 HTTP 服务器。挑战和难度越来越高，你准备好了吗？



极客时间

网络编程实战

从底层到实战，深度解析网络编程

盛延敏

前大众点评云平台首席架构师

新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 期中大作业 | 动手编写一个自己的程序吧！

下一篇 20 | 大名鼎鼎的select：看我如何同时感知多个I/O事件

精选留言 (9)

 写留言



LDxy

2019-09-21

服务器端程序里面的count++是做何用的?

展开 ▼



沉淀的梦想

2019-09-21

测了一些strcmp, 好像写大小写敏感的, 更接近Java里的equals, 而不是equalsIgnoreCase吧?



Steiner

2019-09-20

为什么select要用两个fd_set来操作,只有一个fd_set会出现什么问题?



(一一)

2019-09-20

我用的把dup2把标准输入输出重定向到套接字, 用system调用命令



CountingStars

2019-09-20

可能老师的程序只是为了给我们演示。我发现老师程序主体都在一个main函数写的, 没有分开组织成多个小函数, 有时if的嵌套有点深, 其实可以把异常情况直接return回去, 这样嵌套就会少很多。这样代码也会容易理解一些。

展开 ▼

作者回复: 嗯, 只是一个演示, 你们可以自行优化。拆分成多个函数当然是可以的。



骏Jero

2019-09-20

老师, 有个问题想问下 UDP报文最大长度? 之前你的专栏udp那篇的提问, 我编写代码在局域网试了下可以达到65507个字节。然后参照往上一些资料有些根据mtu来进行计算, 但是为什么实际种事65507而不是mtu 1500字节计算出来的1472字节

展开 ▼

作者回复: 你是怎么测试的? 贴上代码来看看, 很感兴趣的说。

1



刘丹

2019-09-20

malloc的内存没有被释放?

展开 ▼

作者回复: 是的, 已修正。一会更新下。

1



MoonGod

2019-09-20

老师有个问题没有想清楚, 就是服务端代码中, 在读取到客户端发送的EOF后, 会打印printf("server closed \n");。但我在实践的过程中, 发现这行日志总是在客户端重新连接后, 并发送第一条指令后, 才在服务端的控制台打印出来, 为什么不是在客户端发送quit之后立马打印出来的呢?

展开 ▼

作者回复: 我这里的现象是客户端quit之后会打印。你是什么系统?

1



传说中的成大大

2019-09-20

我觉得网络不难 难的是你让我去程序里面 ls pwd cd .../o(╥﹏╥)o

作者回复: 哈哈, 不是都搞定了么

1

