概述

耗时一个星期，终于草草结束了这个网盘项目，虽然还有几个功能基于时间和能力未能实现，但也保证了自己已完成的功能的简洁性和较高的鲁棒性。

- 本项目完成的功能有：

> 一期：
>
> 1. cd
> 2. ls
> 3. puts
> 4. gets
> 5. remove
> 6. pwd
> 7. mkdir
>
> 二期：
>
> 1. 密码验证
> 2. 日志记录
> 3. 断点续传
> 4. mmap传输和recvCycle循环接收
>
> 三期：
>
> 1. 数据库记录日志
> 2. 数据库密码验证（包含首次登陆注册和密码错误重新输入）
>
> 四期：
>
> 1. 超时断开连接

因为本项目已完成的功能大多是对之前Linux阶段学习的熟悉和运用，所以以下会列出一些本人在完成项目过程中遇到的比较困难的点和一些本项目的亮点。

技术难点

1. 本项目完成的功能基本具有较高的鲁棒性

```
yysongzy@linux:~/test/database/client/src$ ./client 192.168.5.35 2000
Please enter your username:123
Please enter password:
This user is already registered
Please enter the command:
cd 123
No such file or directory
Please enter the command:
puts 123
sendfd = -1
open: No such file or directory
No such file, please re-enter the command:
gets 123
recv: Resource temporarily unavailable
No such file or directory
Please enter the command:
123
No such command, please re-enter:
```

- 代码示例节选：

```
    //recvFile.c
    //沉睡1秒，因为下一次recv为非阻塞，所以需要给服务器时间去准备
    //否则很容易是客户端的recv先执行，因为设置为了非阻塞
    //所以由于服务器还未来得及发送数据而返回-1
    sleep(1);
    int dataLen = 0;
    //设置为非阻塞，若若对端无数据发送则直接返回-1
    int ret = recv(sfd, &dataLen, 4, MSG_DONTWAIT);
    ERROR_CHECK(ret, -1, "recv");
```

可以看到每一个错误输入都有错误提示并且可以继续执行接下来的指令而不会导致程序错误。

2. 本项目中的cd功能能做到和shell基本相同的功能（包括cd ~, cd -, cd /, cd ../../..等功能）

```
Please enter the command:
cd ..
/home/yysongzy/test/database/server

include         .               ..              conf            log
src
Please enter the command:
cd /
/

cdrom           etc             dev             var             opt
vmlinuz.old     lost+found      initrd.img      proc            bin
root            run             snap            .               lib
..              tmp             boot            lib64           media
usr             vmlinuz         sbin            initrd.img.old  srv
mnt             swapfile        home            sys
Please enter the command:
cd -
/home/yysongzy/test/database/server

include         .               ..              conf            log
src
Please enter the command:
cd ../client
/home/yysongzy/test/database/client

include         .               ..              src
Please enter the command:
```

- 代码示例节选：

```c
//cmd_CD.c
int cmd_CD(char *curPath, char *data, char *lastPath){
    char newPath[128] = {0};
    if('~' == data[0]){
        memcpy(lastPath, curPath, 128);
        memcpy(curPath, data, 128);
        return 0;
    }
    else if('/' == data[0]){
        memcpy(lastPath, curPath, 128);
        memcpy(curPath, data, 128);
        return 0;
    }
    else if('-' == data[0]){
        memcpy(curPath, lastPath, 128);
        return 0;
    }
    else{
        sprintf(newPath, "%s%c%s", curPath, '/', data);
        if(NULL == opendir(newPath)){
            printf("No such file or directory\n");
            return -1;
        }
        memcpy(lastPath, curPath, 128);
        memcpy(curPath, newPath, 128);
    }
    return 0;
}
```

```c
//threadpool.c
case CD:
    ret = cmd_CD(begPath, msg.data, lastPath);
    if(0 == ret){
        socketpair(AF_UNIX, SOCK_STREAM, 0, pQue->cwdPipe);
        write(pQue->cwdPipe[1], begPath, 128);
        if(0 == fork()){
            //注意：因为创建的进程继承了之前的进程环境，
            //所以当前目录就是家目录
            char parentPath[128] = {0};
            read(pQue->cwdPipe[0], parentPath, 128);
            chdir(parentPath);
            //更新工作目录
            getcwd(parentPath, 128);
            write(pQue->cwdPipe[0], parentPath, 128);
            exit(0);
        }
        read(pQue->cwdPipe[1], begPath, 128);
        send(pCur->clientFd, begPath, sizeof(begPath), 0);
    }
    if(-1 == ret){
        send(pCur->clientFd, noFile, sizeof(noFile), 0);
    }
    break;
```

完成此功能使用了fork()函数创建了一个子进程，并利用socketpair()函数全双工的特性将拼接后的目录（并不符合Linux标准格式）传入，使子进程chdir()到目标目录，再通过getcwd()获得标准的目录格式，最后将结果传输给当前进程。

cd - 功能则通过建立了lastPath字符数组存储。

3. 数据库密码验证（包含首次登陆注册和密码错误重新输入）

```
yysongzy@linux:~/test/database/client/src$ ./client 192.168.5.35 2000
Please enter your username:123
Please enter password:
This is your first login and an account has been registered for you
Please enter the command:
^C
yysongzy@linux:~/test/database/client/src$ ./client 192.168.5.35 2000
Please enter your username:123
Please enter password:
Wrong password, please re-enter
This user is already registered
Please enter the command:
```

- 代码示例节选：

```c
//login.c
int login(int sfd, char *username){
    char salt[64] = {0};
    char ciphertext[72] = {0};
    //先查询此用户名是否已注册
    //queryLogin返回1表示此用户已注册，返回0表示此用户未注册
    //若已注册则取出salt值
    int ret = queryLogin(username, salt);
    //ifRegistered判断是否已注册，若已注册则发送消息给客户端
    //取消之后的salt和密文传送
    //0表示未注册，1表示已注册
    char isRegistered[2] = "0";
    //isPasswordCorrect判断密码是否正确，0表示密码错误，1表示密码正确
    char isPasswordCorrect[2] = "1";
    if(1 == ret){
        //若已注册，则验证客户端发来的密文正确性
        strcpy(isRegistered, "1");
        send(sfd, isRegistered, 2, 0);
        //发送salt
        send(sfd, salt, 64, 0);
label:
        //重置为1
        strcpy(isPasswordCorrect, "1");
        //接收密文
        recv(sfd, ciphertext, 72, 0);
        //queryCiphertext返回1表示密码正确，返回0表示密码错误
        ret = queryCiphertext(ciphertext);
        if(0 == ret){
            strcpy(isPasswordCorrect, "0");
            send(sfd, isPasswordCorrect, 2, 0);
            printf("Wrong password\n");
            goto label;
        }
        else{
            send(sfd, isPasswordCorrect, 2, 0);
```

```
            printf("This user is already registered\n");
        }
        return 0;
    }
    send(sfd, isRegistered, 2, 0);
    //生成salt
    generateSalt(salt);
    //发送salt
    send(sfd, salt, 64, 0);
    //接收密文
    recv(sfd, ciphertext, 72, 0);
    //将用户名，salt，密文写入login数据库
    insertLogin(username, salt, ciphertext);
    return 0;
}
```

```
//main.c
printf("Please enter your username:");
    scanf("%s", username);

    myConnect(sfd, argv[1], argv[2]);

    //发送pid
    send(sfd, &pid, 4, 0);
    //发送用户名
    send(sfd, username, 64, 0);
    char isRegistered[2] = {0};
    char isPasswordCorrect[2] = {0};
    //输入密码
    passwd = getpass("Please enter password:");
    recv(sfd, isRegistered, 2, 0);
    if(0 == strcmp(isRegistered, "0")){
        //接收服务器随机生成的salt
        recv(sfd, salt, 64, 0);
        //根据salt和密码生成密文
        ciphertext = crypt(passwd, salt);
        //发送密文
        send(sfd, ciphertext, 72, 0);
        printf("This is your first login and an account has been
registered for you\n");
    }
    else{
        //接收服务器随机生成的salt
        recv(sfd, salt, 64, 0);
label:
        //根据salt和密码生成密文
        ciphertext = crypt(passwd, salt);
        //发送密文
        send(sfd, ciphertext, 72, 0);
        //接收密码是否正确的消息
        recv(sfd, isPasswordCorrect, 2, 0);
        if(0 == strcmp(isPasswordCorrect, "0")){
            passwd = getpass("Wrong password, please re-enter");
            goto label;
        }
        else{
            printf("This user is already registered\n");
```

```
        }
    }
```

由于用户名和密码由数据库保存和验证，所以灵活性大大提高。

4. 断点续传

```
yysongzy@linux:~/test/database/client/src$ ./client 192.168.5.35 2000
Please enter your username:123
Please enter password:
This user is already registered
Please enter the command:
gets video.avi
fileSize = 56633312
downSize = 0
^C.81%
yysongzy@linux:~/test/database/client/src$ ./client 192.168.5.35 2000
Please enter your username:123
Please enter password:
This user is already registered
Please enter the command:
gets video.avi
fileSize = 56633312
downSize = 6123000
100.00%
Please enter the command:
```

- 代码示例节选：

```c
//recvFile.c
    while(NULL != (ptr = readdir(dir))){
        if(0 == strcmp(fileName, ptr->d_name)){
            sprintf(filePath, "./file_dir/%s", fileName);
            stat(filePath, &st);
            downSize = st.st_size;
            lseek(fd, downSize, SEEK_SET);
            printf("downSize = %ld\n", downSize);
            send(sfd, &downSize, 4, 0);
        }
    }
```

此功能通过记录已接收的文件大小，通过lseek()函数对断点续传的文件进行偏移，再进行文件续传，需要注意的是偏移需在服务器和客户端两端都进行。

5. 超时断开连接

```
exitFlag = 5    pid = 36555     exitFd = 8
Count down: 7
Count down: 8
Count down: 9
Count down: 10
Count down: 11
Count down: 12
Count down: 13
Count down: 14
Count down: 15
Count down: 16
Count down: 17
Count down: 18
Count down: 19
Count down: 20
Count down: 21
Count down: 22
Count down: 23
Count down: 24
Count down: 25
Count down: 26
Count down: 27
Count down: 28
Count down: 29
Count down: 0
Count down: 1
Count down: 2
Count down: 3
Count down: 4
Count down: 5
pid = 36555 is down, bye~
wait
```

- 代码示例节选：

```c
//threadpool.c
                pQue->exitNode[pCur->clientFd].exitFlag = slot++;
                slot %= 30;
                pQue->exitNode[pCur->clientFd].pid = pid;
                pQue->exitNode[pCur->clientFd].exitFd = pCur->clientFd;
                printf("exitFlag = %d\t", pQue->exitNode[pCur->clientFd].exitFlag);
                printf("pid = %d\t", pQue->exitNode[pCur->clientFd].pid);
                printf("exitFd = %d\n", pQue->exitNode[pCur->clientFd].exitFd);
                write(pQue->exitPipe[1], pQue->exitNode, 20 * sizeof(ExitNode_t));
```

```c
//main.c
            else if(timeFd == evs[i].data.fd){
                read(evs[i].data.fd, &exp, sizeof(uint64_t));
                ++slot;
                slot %= 30;
                printf("Count down: %d\n", slot);
                for(int j = 0; j != 20; ++j){
                    if(exitNode[j].exitFlag == slot){
```

```
                        printf("pid = %d is down, bye~\n",
exitNode[j].pid);
                        close(exitNode[j].exitFd);
                        kill(exitNode[j].pid, SIGUSR1);
                        //复原pid、exitFd和exitFd的值，以防重复
                        exitNode[j].pid = -1;
                        exitNode[j].exitFlag = -1;
                        exitNode[j].exitFd = -1;
                    }
                }
            }
            else if(threadPool.que.exitPipe[0] == evs[i].data.fd){
                //接收子线程发送的要退出的子线程信息
                read(threadPool.que.exitPipe[0], exitNode, 20 *
sizeof(ExitNode_t));
```

本功能定义一个全局变量slot通过timerfd系函数建立一个定时器，实现每隔1秒触发一次epoll_wait，并建立一个数组存储即将要退出的进程的pid，数组下标为每个子线程的fd。在触发epoll_wait时遍历数组检测是否有要退出的pid。此功能借鉴了环形队列的思想。

收获

通过完成此项目我学习和收获了以下知识：

1. 函数传参

   完成此项目使我对函数传参的使用更加熟练，在何时需要传值，何时需要传指针或者二级指针，现在都比较清楚。

   做项目时比较记忆犹新的一个点是要写了一个很复杂的函数，本以为要传很多参数进去，没想到最终写完只传了两个参数，使整个函数更加简洁。

2. 字符串处理

   此项目的许多功能都需要对字符串进行处理，比如传递命令，cd功能，通过完成这些功能使我对字符串的处理更加得心应手。

3. 完成小型项目

   通过完成此项目我也清楚了如何去完成一个小型项目，需要在一开始就把基础框架代码写好，尽量没有差错，之后再逐步增加内容。同时要注意增量编写，每写完一个功能之后都及时检查是否有错误，且要保证鲁棒性，可以说在完成这个项目的过程中，我花在debug上的时间比写各个功能的时间要长的多。