Project3 Report

目录

Project3 Report 1

1. Step1 1

1.1 Description 1

1.2 implementation 2

1 step2 4

2.1 Description 4

2.2 Implementation 4

3 Step3 18

3.1 Description 18

3.2 Implement 18

1. Step1

1.1 Description

Implement, as an Internet-domain socket server, a simulation of a physical disk. The simulated disk is organized by cylinder and sector.

BDS will simulate the disk serve.

BDC will simulate the client

Usage:

./BDS <DiskFileName> <#cylinders> <#sector per cylinder> <track-to-track delay>

<port=10356>

./BDC <DiskServerAddress> <port=10356>

1.2 implementation

Cmd_r

```
int arg_num = 2;
    int count = 0;
    char *beh_args[2];
    char* p = strtok(args, " ");
    while (p && count < arg_num)
    {
        beh_args[count] = p;
        count++;
        p = strtok(NULL, " ");
    }
```

```
char *data = diskfile + (c * nsec + s) * BLOCKSIZE;
            char buf[261];
            buf[0] = 'Y';
            buf[1] = 'e';
            buf[2] = 's';
            buf[3] = ' ';
            int j = 4;
            buf[260] = '\0';
            printf("%s", buf);
            for (int i = 0; i < BLOCKSIZE;i++)
                buf[j++] = data[i];
            buf[260] = '\0';
            // buf[j] = '\0';
            // memcpy(buf, &diskfile[BLOCKSIZE * (c * nsec +
s)], BLOCKSIZE);
            // memcpy(buf, &data, BLOCKSIZE);
            send_to_buffer(write_buf,buf , 261);
```

Cmd_w

```
// 写入数据
            for (int i = 0; i < strlen(data);i++)
                printf("%d\n", (int)data[i]);
```

```
            send_to_buffer(write_buf, "Yes", 4);
            char *write_start = diskfile + (c * nsec + s) *
BLOCKSIZE;
            if (l < BLOCKSIZE)
            {
                int i = 0;
                for (i; i < strlen(data);i++)
                    write_start[i] = data[i];
                for (i; i < BLOCKSIZE;i++)
                    write_start[i] = 0;
            }
            else{
                for (int i = 0; i < BLOCKSIZE;i++)
                    write_start[i] = data[i];
            }
```

Open file and mmap

```
// open file
        //  O_RDWR 以可读写方式打开文件
        // O_CREAT 若欲打开的文件不存在则自动建立该文件.
        int fd = open(diskfname, O_RDWR | O_CREAT, 0);
        if(fd<0){
            printf("Error: Could not open file '%s'.\n",
    diskfname);
            exit(-1);
        }

        // stretch the file
        long FILESIZE = BLOCKSIZE * nsec * ncyl;//磁盘文件大小
        int result = lseek(fd, FILESIZE - 1, SEEK_SET);
        if (result == -1)
        {
            perror("Error calling lseek() to 'stretch' the
    file");
            close(fd);
            exit(-1);
        }

        result = write(fd, "", 1);//结尾插入空字符
        if (result != 1)
        {
            perror("Error writing last byte of the file");
```

```
            close(fd);
            exit(-1);
        }
        // mmap
        // char *diskfile;
        diskfile = (char *)mmap(NULL, FILESIZE,
                                PROT_READ | PROT_WRITE,
                                MAP_SHARED, fd, 0);
        if (diskfile == MAP_FAILED)
        {
            close(fd);
            printf("Error: Could not map file.\n");
            exit(-1);
        }
```

1 step2

2.1 Description

Implement an inode file system. The file system should provide operations including:

• Initialize the file system

• Create a file

• Read data from a file

• Write the given data to a file

• Append data to a file • Remove a file

• Create directories

2.2 Implementation

Inode and dinode

```
typedef struct inode
{
    // uint8_t uid;                    //owner id
    uint8_t mode;                      // 0 代表 empty，1 代表文件，2
代表目录
```

```
    uint8_t link_count;              // block 数量
    uint16_t uid;                    // owner id
    uint16_t size;                   // 文件大小
    uint16_t index;                  // inode 编号
    uint32_t time;                   // 记录文件修改时间
    uint16_t parent_inode;           // parent_inode 编号
    uint16_t direct_block[8];        // 8 个直接块,直接连接 block
    uint16_t single_indirect; // 一个间接块,间接块仍然连接 inode

} inode;
typedef struct dinode
{
    uint16_t inode_id;         // 当前目录项表示的文件/目录的对应
inode
    uint8_t valid;             // 当前目录项是否有效
    uint8_t type;              // 当前目录项类型（文件/目录）
    char name[28];             //凑成 32byte 和 inode 一样，方便管理
} dinode;
```

Superblock

```
typedef struct super_block
{
    uint32_t used_inodes_count;//使用的
    uint32_t used_blocks_count;
    uint32_t free_inodes_count;//空闲的
    uint32_t free_blocks_count;
    uint32_t root;
    uint32_t inode_map[INODE_MAP]; // 按位记录 inode 是否空闲
    uint32_t block_map[BLOCK_MAP]; // 按位记录 block 是否空闲

} super_block;
```

Cmd

```
typedef struct cmd
{
    char type;
    int block_id;
    char data[256];
} cmd;
```

Utils

```
//alloc and  free
```

```
int alloc_inode();
int free_inode(inode *node);
int alloc_block();
int free_block(uint16_t index);

//
int write_inode_to_disk(inode *node, uint16_t index,int
client);
int write_block(int block_id, char *buf, int client);
int read_block(int block_id, char *buf, int client);

int superblock_init()
int rootinode_init()
//目录中查找文件,返回文件所在的位置
int dir_search(inode *node, char *name, int type)
//向目录中添加文件
int dir_add_inode(inode *node, char *name, uint8_t type)
//向目录中删除文件
int dir_remove_inode(inode* node, char *name,int type)
int read_file(inode *node, char *ret)
int write_file(inode *node, char *data)
```

f

```
// format
int cmd_f(tcp_buffer *write_buf, char *args, int len){
    //init
    superblock_init();
    rootinode_init();
    cur_dir = 0;//0 是 root inode
    strcpy(cur_dir_str, "/");
    for (int i = 1; i < MAX_INODE_COUNT; i++)
    {
        if (init_inode(&inode_table[i], 0,0,0,0,i,114514) < 0)
        {
            printf("Init inode failed.\n");
            return 0;
        }
    }


    format = true;
    send_to_buffer(write_buf, "Format is done", 15);
```

```
    return 0;
}
```

Mk

```
int cmd_mk(tcp_buffer *write_buf, char *args, int len)
{
    if(!format){
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // for (int i = 0; i < strlen(args);i++)
    //     printf("%d\n", args[i]);
    char *name = strtok(args, " \r\n");
    // printf("name:%s\n", name);

    //在文件夹内判断是否重名
    if (dir_search(&inode_table[cur_dir], name,0)>=0)
    {
        printf("The name has existed");
        send_to_buffer(write_buf, "The name has existed", 21);
        return 0;
    }

    //在 dir 中添加文件，每个 dir 有 8 个直接块，每个块又有 8 个 inode，
也就是说一个 dir 可以有 64 个文件（包括子文件夹）
    //在添加文件的时候，要搜索所有块，因为空缺块的位置不确定
    if (dir_add_inode(&inode_table[cur_dir], name, 0) < 0)
    {
        printf("mk failed\n");
        send_to_buffer(write_buf, "mk failed", 10);
        return 0;
    }
    else{
        printf("mk successful\n");
        send_to_buffer(write_buf, "mk successful", 14);
        return 0;
    }
    return 0;
}
```

Mkdir

```
int cmd_mkdir(tcp_buffer *write_buf, char *args, int len)
{
    //format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    //name
    char *name = strtok(args, " \r\n");

    if (dir_search(&inode_table[cur_dir], name, 1) >= 0)
    { // 重名
        printf("Directory name has existed");
        send_to_buffer(write_buf, "Directory name has existed",
27);
        return 0;
    }
    if (dir_add_inode(&inode_table[cur_dir], name, 1) < 0)
    {
        printf("mkdir failed\n");
        send_to_buffer(write_buf, "mkdir failed", 13);
        return 0;
    }
    else
    {
        printf("mkdir successful\n");
        send_to_buffer(write_buf, "mkdir successful", 17);
        return 0;
    }
    return 0;

}
```

Rm

```
int cmd_rm(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
```

```
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char *name = strtok(args, " \r\n");

    if (dir_remove_inode(&inode_table[cur_dir],name,0) < 0){
            printf("rm failed\n");
            send_to_buffer(write_buf, "rm failed", 10);
            return 0;
    }
    else{
        printf("rm successful\n");
        send_to_buffer(write_buf, "rm successful", 14);
        return 0;
    }
    return 0;
}
```

Cd

```
int cmd_cd(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char *name = strtok(args, " \r\n");
    // 先保存当前 dir
    uint16_t pre_dir = cur_dir;
    char pre_dir_str[1000];
    strcpy(pre_dir_str, cur_dir_str);

    char *path = strtok(name, "/");
    while (path != NULL)
    {
        if (strcmp(path, "..") == 0)
        { // 回到上一级
```

```
            uint16_t parent =
inode_table[cur_dir].parent_inode;
            if (parent != 114514)
            {
                cur_dir = parent;

                if (strcmp(cur_dir_str, "/") != 0)
                {
                    char *temp = strrchr(cur_dir_str, '/');
                    *temp = '\0';
                }
                if (parent == 0)
                    strcpy(cur_dir_str, "/");
            }
        }
        else
        {
            int ret = dir_search(&inode_table[cur_dir], path,
1);
            if (ret == -1)
            {
                printf("NO directory\n");
                send_to_buffer(write_buf, "NO directory", 13);
                cur_dir = pre_dir; // 没有对应文件夹
                strcpy(cur_dir_str, pre_dir_str);
                return 0;
            }
            else{
                if (cur_dir != 0)
                    strcat(cur_dir_str, "/");
                cur_dir = ret;
                strcat(cur_dir_str, path);
            }

        }
        path = strtok(NULL, "/");
    }
    char tmp[100];
    sprintf(tmp,"cur_dir:%d  cur_dir_str:%s\n", cur_dir,
cur_dir_str);
    printf("%s\n", tmp);
```

```
    send_to_buffer(write_buf,tmp,strlen(tmp+1));//一定要 send 出
去，不然会报错
    return 0;
}
```

Rmdir

```
int cmd_rmdir(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char *name = strtok(args, " \r\n");

    if (dir_remove_inode(&inode_table[cur_dir], name,1) < 0)
    {
        printf("rmdir failed\n");
        send_to_buffer(write_buf, "rmdir failed", 13);
        return 0;
    }
    else
    {
        printf("rmdir successful\n");
        send_to_buffer(write_buf, "rmdir successful", 17);
        return 0;
    }
    return 0;
}
```

Ls

```
int cmd_ls(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
```

```
        return 0;
    }


    _Bool flag = false; // 文件夹是否为空

    char file_name[100][100];
    uint16_t inode_list[100];//存放文件的 inode_id
    int file_count = 0, dir_count = 0, count=0;

    dinode dir_items[8];
    inode *node = &inode_table[cur_dir];
    char buf[BLOCK_SIZE];
    char list[10000] = "";
    for (int i = 0; i < 8; i++)
    {
        if (node->direct_block[i] == 0)
            continue;
        if (read_block(node->direct_block[i], buf,client) < 0)
            continue;
        memcpy(&dir_items, buf, BLOCK_SIZE);
        for (int j = 0; j < 8; j++)
        {
            if (dir_items[j].valid)
            {
                printf("%d\n", dir_items[j].inode_id);
                 inode_list[count] = dir_items[j].inode_id;
                strcpy(file_name[count++], dir_items[j].name);
                if (dir_items[j].type == 0){
                    file_count++;
                }

                else{
                    dir_count++;
                }
                 flag = true;
            }
        }
    }
    // 排序
    qsort(file_name, count, sizeof(file_name[0]), cmp);
```

```
    for (int i = 0; i < count; i++)
    {
        strcat(list, file_name[i]);
        strcat(list, "      ");
        inode *tmp = &inode_table[inode_list[i]];
        char ctmp[100];
        sprintf(ctmp, "type:%d size:%d time:%d\n",tmp->mode,
tmp->size, tmp->time);
        strcat(list, ctmp);
    }
    if(flag==false){
        send_to_buffer(write_buf, "dir_num:0\nfile_num:0",21);
    }

    else{
        char ctmp[100];
        sprintf(ctmp, "dir_num:%d\nfile_num:%d", dir_count,
file_count);
        strcat(list, ctmp);
        send_to_buffer(write_buf, list, strlen(list) + 1);
    }

    return 0;
}
```

Catch

```
int cmd_cat(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char *name = strtok(args, " \r\n");


    char buf[60000]="";
    int id;
    //没有该文件
```

```
    if ((id = dir_search(&inode_table[cur_dir], name, 0)) < 0)
    {
        printf("File does not exist\n");
        send_to_buffer(write_buf, "File does not exist",20);
        return 0;
    }

    if (read_file(&inode_table[id], buf) == 0)
    {
        printf("%s\n", buf);
        send_to_buffer(write_buf, buf, strlen(buf) + 1);
    }
    else
        send_to_buffer(write_buf, "Can't catch the file",21);
    return 0;
}
```

w

```
int cmd_w(tcp_buffer *write_buf, char *args, int length)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char* name = strtok(args, " \r\n");//name
    int len = atoi(strtok(NULL, " \r\n"));//len
    char *data = strtok(NULL, " \r\n");//data
    printf("name:%s len:%d data:%s\n", name, len, data);

    int id;
    // 没有该文件
    if ((id = dir_search(&inode_table[cur_dir], name, 0)) < 0)
    {
        printf("File does not exist\n");
        send_to_buffer(write_buf, "File does not exist", 20);
        return 0;
    }
```

```
    if (write_file(&inode_table[id], data) == 0){
        printf("Write data successful\n");
        send_to_buffer(write_buf, "Write data successful", 22);
    }

    else
    {
        printf("Write data failed\n");
        send_to_buffer(write_buf, "Write data failed", 18);
    }
    return 0;
}
```

I

```
int cmd_i(tcp_buffer *write_buf, char *args, int length)
{
    if (!format)
    {
        printf("Don't  format");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
    char *name = strtok(args, " \r\n"); // name
    int pos = atoi(strtok(NULL, " \r\n"));    //pos
    int len = atoi(strtok(NULL, " \r\n"));    // len
    char *data = strtok(NULL, " \r\n");       // data
    printf("name:%s pos:%d len:%d data:%s\n", name, pos, len,
data);

    char buf[10000];

    int id = 0;
    if(len<=0)
        return 0;

    // 文件不存在
    if ((id = dir_search(&inode_table[cur_dir], name, 0)) < 0)
    {
        printf("File does not exist\n");
        send_to_buffer(write_buf, "File does not exist", 20);
        return 0;
```

```
    }


    int file_size = inode_table[id].size; // 获取文件当前的大小
    if (pos > file_size)
        pos = file_size;//insert 到末尾

    int new_size = file_size + len;

    memset(buf, 0, 10000);
    int ret = read_file(&inode_table[id], buf);

    // 在指定位置插入数据
    //memove 将 src 开始的 N 位移动到 dst 的位置
    memmove(buf + pos + len, buf + pos, file_size - pos);
    memcpy(buf + pos, data, len);
    buf[new_size] = '\0';

    if (write_file(&inode_table[id], buf) == 0){
        printf("Insert data successful\n");
        send_to_buffer(write_buf, "Insert data successful",
23);
    }

    else
    {
        printf("Insert data failed\n");
        send_to_buffer(write_buf, "Insert data failed", 19);
    }
    return 0;
}
```

d

```
int cmd_d(tcp_buffer *write_buf, char *args, int length)
{
    if (!format)
    {
        printf("Don't  format\n");
        send_to_buffer(write_buf, "Please format first", 20);
        return 0;
    }
    // name
```

```
    char *name = strtok(args, " \r\n");    // name
    int pos = atoi(strtok(NULL, " \r\n")); // pos
    int len = atoi(strtok(NULL, " \r\n")); // len
    printf("name:%s pos:%d len:%d\n", name, pos, len);

    int id;
    if ((id = dir_search(&inode_table[cur_dir], name, 0)) < 0)
    { // 文件不存在
        printf("File does not exist\n");
        send_to_buffer(write_buf, "File does not exist", 20);
        return 0;
    }

    int file_size = inode_table[id].size;
    if (pos > file_size)
    {
        printf("Pos can't larger than file_size\n");
        send_to_buffer(write_buf, "Pos can't larger than
file_size", 32);
        return 0;
    }

    //当删除长度过长
    if (file_size < pos + len)
        len = file_size - pos;

    char buf[10000];
    memset(buf, 0, sizeof(buf));
    int ret = read_file(&inode_table[id], buf);

    // 在指定位置删除数据
    //memmove(buf+pos, buf+pos+len, file_size-pos-len);
    char tmp[10000];
    memcpy(tmp, buf, pos);
    memcpy(tmp + pos, buf + pos + len, file_size - pos - len);
    tmp[file_size - len] = '\0';

    if (write_file(&inode_table[id], tmp) == 0){
        printf("Delete data successful\n");
        send_to_buffer(write_buf, "Delete data successful",
23);
    }
```

```
    else
    {
        printf("Delete data failed\n");
        send_to_buffer(write_buf, "Delete data failed", 19);
    }
    return 0;
}
```

Dir

```
//返回 cur_dir
int cmd_dir(tcp_buffer *write_buf, char *args, int len)
{
    send_to_buffer(write_buf,cur_dir_str,strlen(cur_dir_str)+1)
;
    return 0;
}
```

3 Step3

3.1 Description

Support multiple users in the file system

3.2 Implement

FC

```
 // 每次都获取一下当前所在目录
    // 我新定义了一个命令 dir
    char *command = "dir";
    char dir[100] = "";
    char username[100];
    static char buf[4096];
    int n;
    // 处理用户名
    printf("请输入你的用户名: ");
    scanf("%s", username);
    char tmp = getchar();//吃掉回车，scanf 会把回车留在缓存区
    sprintf(buf, "user %s", username);

    client_send(client, buf, strlen(buf) + 1);
    n=client_recv(client, buf, sizeof(buf));
    buf[n] = 0;
```

```
    printf("%s\n", buf);

    while (1)
    {
        client_send(client, command, strlen(command) + 1);
        n = client_recv(client, dir, sizeof(dir));
        dir[n] = 0;
        printf("%s$ ", dir);

        //读取
        fgets(buf, sizeof(buf), stdin);
        if (feof(stdin))
            break;
        client_send(client, buf, strlen(buf) + 1);
        int n = client_recv(client, buf, sizeof(buf));
        buf[n] = 0;
        printf("%s\n", buf);
        if (strcmp(buf, "Bye!") == 0)
            break;
    }
    client_destroy(client);
```

FS

User

```
//step3 中保存用户
int cmd_user(tcp_buffer *write_buf, char *args, int len)
{
    // format check
    if (!format)
    {
        printf("Don't  format\n");
        strcpy(response, "Please format first");
        // send_to_buffer(write_buf, "Please format first",
20);
        return 0;
    }
    printf("dir:%d dir_str:%s\n", cur_dir, cur_dir_str);
    char *name = strtok(args, " \r\n"); // name
    // 调用一下 f 指令初始化一下
    // cmd_f(write_buf, args, len);
    // 保存用户信息
```

```
    // 查找是否有该用户
    for (int i = 0; i < user_count;i++){
        if(strcmp(user_table[i],name)==0){
            //如果已经有该用户，则已经有该用户的文件夹了
            //回到该文件夹
            cmd_cd(write_buf, name, len);
            char tmp[100];
            sprintf(tmp,"Welcome to back! %s", name);
            strcpy(response, tmp);
            // send_to_buffer(write_buf, tmp, strlen(tmp) + 1);
            return 0;
        }
    }
    // 如果没有该用户,添加到用户表
    strcpy(user_table[user_count], name);
    user_count++;
    // 创建用户文件夹
    cmd_mkdir(write_buf, name, len);
    cmd_cd(write_buf, name, len);
    char tmp[100];
    sprintf(tmp,"New space is ready for you %s", name);
    strcpy(response,tmp);
    //send_to_buffer(write_buf, tmp, strlen(tmp) + 1);
    return 0;
}
```