

基于 linux 的 socket 文件传输系统

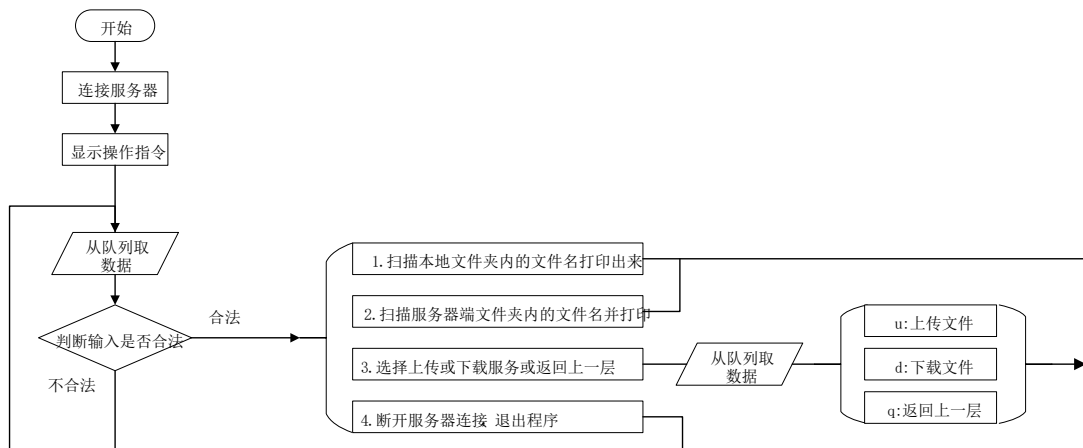
1.客户端操作流程与运行界面

1.1 客户端运行流程图

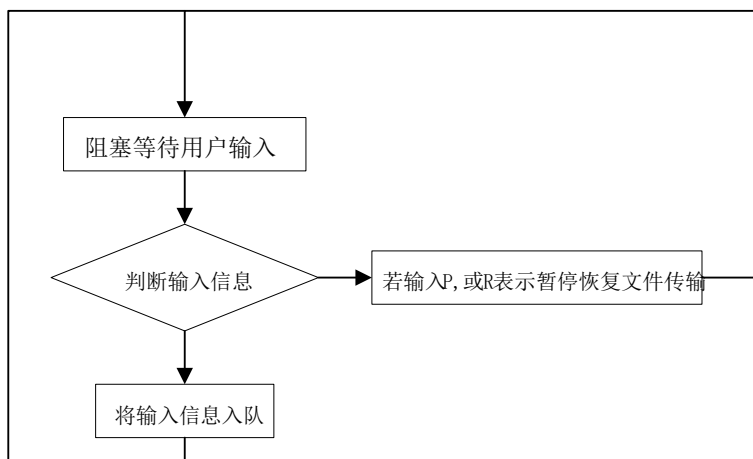
实现如下几个功能：

1. 扫描本地和服务端指定文件夹内文件列表。
2. 客户端支持下载文件，长传文件到服务器。
3. 支持文件断点续传，当传输文件时客户端或者服务任意一方或同时挂掉，再次传输文件从断点处续传文件。
4. 使用文件 md5 校验，续传文件的 md5 校验不一致，重传文件。
5. 支持文件传输时，暂停与恢复传输功能。
6. 服务器支持多客户端同时连接。

主进程：



线程：



1.2 基本运行界面截图

```
gcc file_client.c file_md5.c -lrt -o client
./client 127.0.0.1
socket connect success.
server:127.0.0.1 port:6969 connected!
输入一下命令进行选择:
1.扫描本地文件
2.扫描服务器端文件
3.开始文件传输
4.结束传输
-----
█
```

图 1 客户端运行界面

输入指令 1，和 2 显示本地和服务器端文件夹内文件内容：

```
gcc file_client.c file_md5.c -lrt -o client
./client 127.0.0.1
socket connect success.
server:127.0.0.1 port:6969 connected!
输入一下命令进行选择:
1.扫描本地文件
2.扫描服务器端文件
3.开始文件传输
4.结束传输
-----
1
扫描本地文件夹内文件名:
1.file_client.c
2.Makefile
3.client
4.file_md5.h
5.file_md5.c
-----
2
扫描服务器端的文件名:
server file names:
1. main.c
2. tit
3. server
4. Makefile
5. file_md5.h
6. file_server.c
7. file_md5.c
-----
```

图 2 显示文件名称 输入指

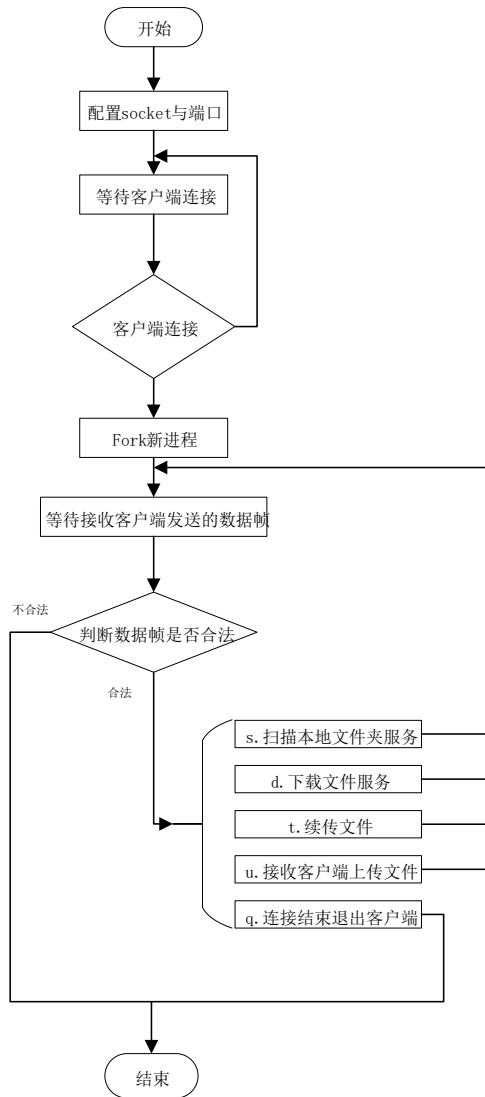
令 3，与 d 和文件名下载文件

```
2
扫描服务器端的文件名:
server file names:
1. main.c
2. tit
3. server
4. Makefile
5. file_md5.h
6. file_server.c
7. file_md5.c
-----
3
请输入u选择上传文件,d选择下载文件,q返回上一层
d
请输入文件名:
tit
文件名为: tit
文件传输结束
-----
```

图 3 客户端下载文件

2.服务器端

2.1 信息传输流程



2.2 main 函数主要接收逻辑

```

while(1){
    //一直循环等待接收信息
    if(0 == (recv(new_server_socket, &server_data, sizeof(server_data), 0))){
        printf("接收出错, 客户端退出\n");
        close(new_server_socket);
        exit(0);
    }
    switch(server_data.command){ //判断Command类型
        case 's':{
            Reply_file_name_formation(new_server_socket); //回复扫描到的信息
            break;
        }
        case 'd':{
            //提取出来要接收的文件名, 客户端不存在的文件
            Read_file_to_client(new_server_socket, server_data);
            break;
        }
        case 't':{
            Read_file_tmp_to_client(new_server_socket, server_data); //判断客户端文件的md5校验, 并移动文件指针
            break;
        }
        case 'u':{
            //从客户端提取出来文件名, 开始接收文件
            Save_file_from_client(new_server_socket, server_data);
            break;
        }
        case 'q':{
            printf("结束连接, 客户端退出\n");
            close(new_server_socket);
            exit(0);
        }
        default:{
            printf("传输错误, 强制退出! \n");
            close(new_server_socket);
            exit(0);
        }
    }
} //end switch server_data.command //end switch

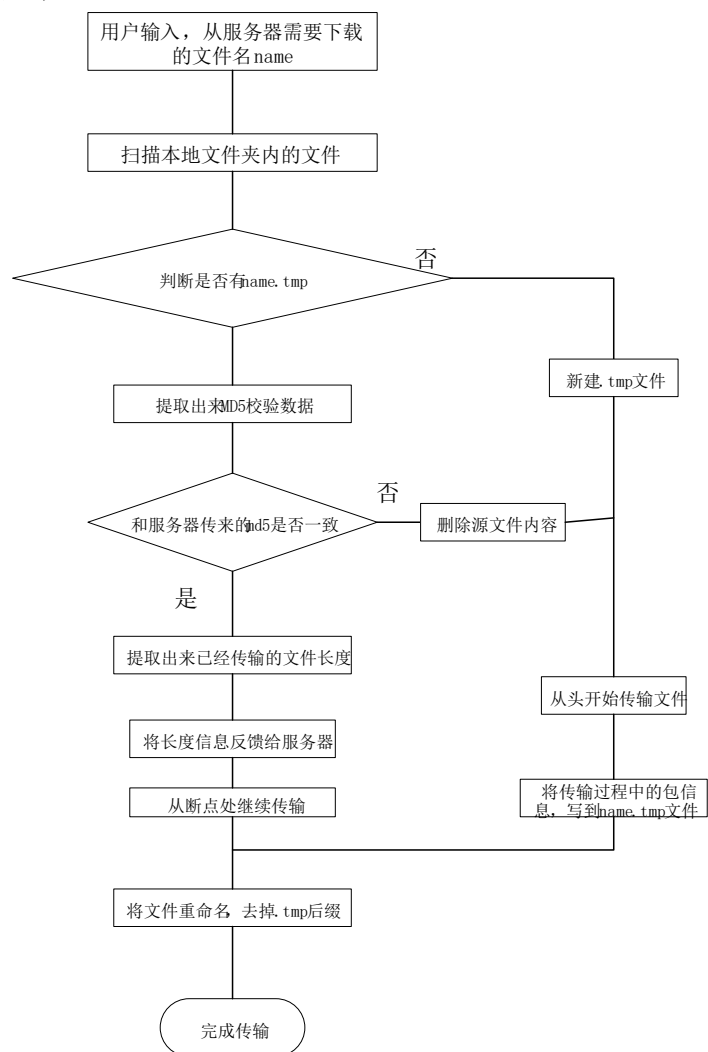
// * end while 1 *
close(new_server_socket);
exit(0);

// * end if, pclosefork() == 0 *
close(new_server_socket);
while(waitpid(-1, NULL, WNOHANG) > 0) //等待子进程结束, 杀掉僵尸进程
    continue;
// * end while 1 *
}

```

图 4 main 函数核心

3.文件传输流程:



4.传输数据帧格式:

段名称	帧格式	包数	数据段
字节大小	Char(1 字节)	Unsigned int(4 字节)	Char(1024 字节)
文件传输模式	命令字	已传包数	文件数据
指令模式 1	命令字	文件长度	md5(32 字节)+文件名字
指令模式 2	命令字	Tmp 文件已传包数	新文件的 md5 校验