摘 要

为了检验学习能力，自学制作手机Ftp客户端。

**关键词：**Ftp，Java，Android

目 录

[摘 要 I](#_Toc70928320)

[目 录 II](#_Toc70928321)

[第一章 绪论 4](#_Toc70928322)

[1.1 完成的工作任务 4](#_Toc70928323)

[第二章 java和Android基础 4](#_Toc70928324)

[2.1 Java基础 4](#_Toc70928325)

[2.1.1 基础语法 4](#_Toc70928326)

[2.1.2 面向对象 4](#_Toc70928327)

[2.1.3 文件操作 5](#_Toc70928328)

[2.1.4 IO流 5](#_Toc70928329)

[2.1.5 网络编程 6](#_Toc70928330)

[2.2 Android基础 6](#_Toc70928331)

[2.2.1 应用组件 7](#_Toc70928332)

[2.2.2 Res目录 7](#_Toc70928333)

[2.2.3 AndroidManifest 7](#_Toc70928334)

[2.2.4 常用类和控件 7](#_Toc70928335)

[第三章 Ftp协议 8](#_Toc70928336)

[3.1 Ftp的工作流程 8](#_Toc70928337)

[3.1.1 Ftp的传输方式 8](#_Toc70928338)

[3.1.2 Ftp的传输模式 8](#_Toc70928339)

[3.2 Ftp的命令与应答 9](#_Toc70928340)

[第四章 Ftp客户端实现 10](#_Toc70928341)

[4.1 创建界面 10](#_Toc70928342)

[4.2 组织结构 10](#_Toc70928343)

[4.3 主界面 11](#_Toc70928344)

[4.4 列表界面 11](#_Toc70928345)

[4.5 Ftpclient的实现 11](#_Toc70928346)

[4.6 AndroidManifest的设置 12](#_Toc70928347)

[第五章 全文总结与展望 13](#_Toc70928348)

[5.1 全文总结 13](#_Toc70928349)

[5.2 程序运行截图 13](#_Toc70928350)

第一章 绪论

1.1 完成的工作任务

通过自学java以及Android流程，实现了手机Ftp客户端。客户端使用套接字编程，没有引用第三方Ftp库，主要功能有登陆到标准Ftp服务器，切换文件目录，上传以及下载文件，移动文件，重命名文件，新建文件夹，删除文件，刷新界面等。

第二章 java和Android基础

2.1 Java基础

Java和Kotlin是Android推荐的两种编程语言，要想进行Android的开发，至少先了解其中的一种，我选择了Java作为本次实验的语言。为了进行Ftp客户端的实现，至少需要了解Java的基础语法、文件操作、IO流以及网络编程。

2.1.1 基础语法

类的定义首先要创建java文件，与类名相同。然后按照public class 类名{}来定义。在类的内部可以定义变量和方法，变量用来描述这个类所需要的信息，方法表示这个类可以有什么操作。

类在使用前一般需要创建一个类的实例，然后调用这个实例的方法来完成对数据的操作。每个实例都是在内存上开辟了一个新的空间区域，因此互不相同。

Java的Vector类实现了动态数组的功能，可以方便地更改数组的大小和数据。主要的方法有add（插入元素）、clear（删除所有元素）、clone（复制向量）、get（获取指定编号的数据）等等。

2.1.2 面向对象

Java是一种面向对象的编程语言。而面向对象编程，是一种通过对象的方式，把现实世界映射到计算机模型的一种编程方法。类是一种对象模版，它定义了如何创建实例。而实例是根据类创建的，可以创建多个实例，每个实例类型相同，但各自属性可能不相同。面向对象的封装性就是尽可能的隐藏对象内部细节，继承能使得子类的对象拥有父类的全部属性与方法，多态性使得父类中定义的属性或方法被子类继承之后，可以具有不同的数据类型或表现出不同的行为，方法重载能让类以统一的方式处理不同数据类型的手段。与方法对应的方法重写能让子类对父类的方法进行重新编写，让父类方法无法使用。

2.1.3 文件操作

File类是文件和目录路径名的抽象表示形式，因此要想操作文件需要使用File类。

File(URI uri)和File(String pathname)都可以用来新建一个File实例，前者采用Uri来确定文件的位置，后者使用路径。在Ftp客户端中，采用了前者，利用Uri来确定文件位置。File类主要的方法有getName（获取文件名）、getPath（获取文件路径）、delete（删除）、mkdir（创建目录）等等。

2.1.4 IO流

Java有很多种IO流，都是从基类逐步派生的，下图为层次图。

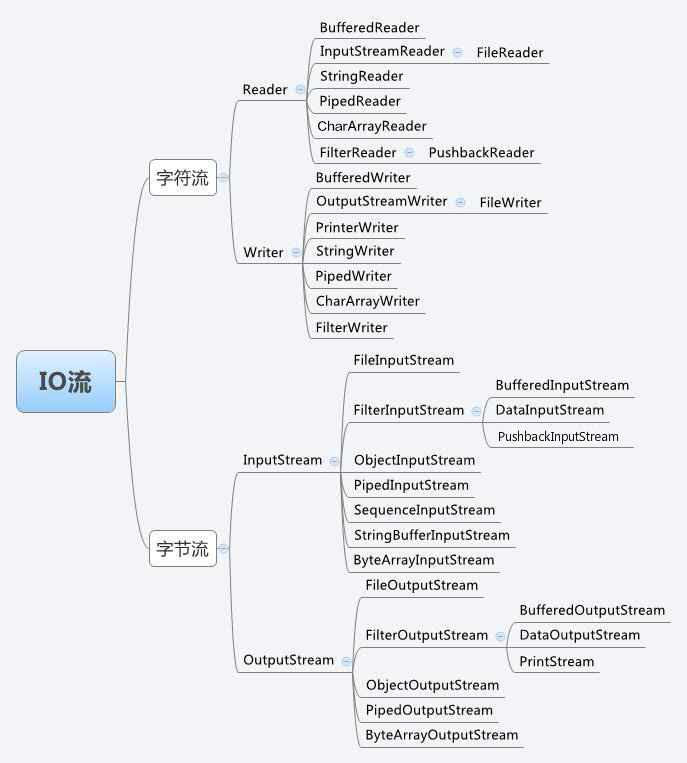


图2-1 IO流层次图

在本次实验中主要使用了字符流中的BufferedReader、PrintWriter以及字节流中的BufferedInputStream、BufferedOutputStream。这些IO流有缓冲输入输出的能力，不需要一个一个的进行传输，极大地降低了网络传输中的网络开销，提高效率。

以BufferedInputStream为例子，它的构造函数BufferedInputStream(InputStream in)，主要使用的方法是int read(byte[] b, int off, int len)。这个方法读取从off开始的len个字节，将它们保存到b数组中，返回保存成功的个数。通过这个方法可以方便的获取文件。

要注意的是在InputStreamReader中，因为读取的信息包含中文，为了防止中文乱码的情况，需要手动指定编码方式，中文有GBK和UTF-8两种编码方式，在使用时要灵活选择对应的编码方式。

2.1.5 网络编程

Ftp协议是搭建在TCP协议上的，TCP协议是面向连接的通信协议，即在传输数据前先在发送端和接收端建立逻辑连接，然后再传输数据，它提供了两台计算机之间可靠无差错的数据传输。在TCP连接中必须要明确客户端与服务器端，由客户端向服务端发出连接请求，每次连接的创建都需要经过“三次握手”。

TCP的两端都需要创建socket对象来建立连接。使用Socket(String host，int port)可以创建一个socket实例，系统会自动经过三次握手来连接到服务器。在本次实验中，主要使用getInputStream()和getOutputStream()函数来获得输入输出流，用来发送和接受数据。

2.2 Android基础

Android是一个操作系统，为了在上面构建应用，首先需要了解相应的基础知识。官方的文档十分友好，可以从中了解到大量内容。 应用组件是 Android 应用的基本构建块，系统或用户可通过组件进入应用。Res目录中是应用资源，指的是代码使用的附加文件和静态内容，例如位图、布局定义、界面字符串、动画说明等。AndroidManifest.xml是应用清单概览文件，里面包含了应用的软件包名称、应用的组件、应用为访问系统或其他应用的受保护部分所需的权限和应用需要的硬件和软件功能。

2.2.1 应用组件

应用组件有四种：Activity、服务、广播接收器、内容提供程序。

Activity是与用户交互的入口点，这代表了一个界面。服务用于使应用在后台保持运行状态。广播接收器能够让系统在常规用户流之外向应用传递事件，从而允许应用响应系统范围内的广播通知。内容提供程序管理一组共享的应用数据。

本次实验主要使用了Activity，为了启动Activity，需要使用Intent来沟通2个Activity，使它们之间可以传递数据。最简单的方法是用Intent intent = new Intent(A.this, B.class)来创建一个A到B的Intent，然后使用startActivity(intent)来从A界面跳转到B界面。

2.2.2 Res目录

在res目录下的layout文件夹，里面放置的是各个界面的布局文件。布局文件是xml格式，里面包含了界面下的控件信息，每个控件都有其对应的id以及布局参数。通过修改布局文件可以改变控件之间的依赖关系等。

2.2.3 AndroidManifest

AndroidManifest文件是Android项目关键的文件，它也被称为清单文件。在其中的Activity属性中，android:parentActivityName可以设置某个Activity的父界面，这样就可以做到在不同的界面中通过返回键跳转。

2.2.4 常用类和控件

Toast类是一个用来提示信息的常用类，它的最简单的调用方式为使用makeText方法创建一个提示信息。用法为Toast.makeText(MainActivity.this, "提示的内容", Toast.LENGTH\_LONG).show()，其中第一个参数是上下文对象，第三个参数是显示的时长。要想自定义更多样式的提示信息，则需要通过构造方法来定制Toast。

AlertDialog可以简单的创建一个对话框来让用户进行选择，但是和Toast不同，AlertDialog需要使用构建器来创建。要使用AlertDialog，首先需要创建一个Builder，然后使用setTitle、setMessage、setPositiveButton、setNegativeButton来分别设置对话框的标题、信息、确认按钮、取消按钮，最后使用show显示对话框。为了区分用户点击了哪个按钮并且进行对应的操作，可以使用匿名函数或者lambda重载onClick函数，设置按下该按钮后触发的行为。

第三章 Ftp协议

3.1 Ftp的工作流程

Ftp使用了两个TCP来传输文件，一个用来控制，默认端口为21，另一个用来传输数据，默认端口为20。控制连接用于传输指令，不参与文件的传输。当需要进行文件传输时，控制连接给出一个端口号，在此端口上建立一个新的连接进行文件的传输，传输完成后这个连接被关闭。

3.1.1 Ftp的传输方式

Ftp有两种传输方式：文本方式和二进制方式。

一般来说，传输文本文件使用文本方式，而其他文件例如程序，图像等则采用二进制方式传输。因为文本方式会对回车进行处理，将其转换成不同系统下的回车，所以会导致程序等乱码，因此要仔细区分文本和二进制方式。

3.1.2 Ftp的传输模式

Ftp有两种传输模式：主动模式和被动模式。

主动模式需要客户端自己打开数据端口，因此需要用户具有设置防火墙等的能力，使用不方便。

主动模式的连接流程为：

1.客户端随机打开一个本地大于1024的端口P，通过此端口连接服务器21端口。

2.客户端告诉服务器数据端口为P+1.

3.服务器通过P+1端口发送数据。

被动模式由客户端主动建立连接，数据传输不会被防火墙拦截，因此更加常用和普适。

被动模式的连接流程为：

1.客户端随机打开两个端口P1，P2.

2.客户端通过P1连接服务器的21端口，发送PASV命令。

3.服务器随机打开一个端口，传输给客户端内容：

227 entering passive mode(a,b,c,d,e1,e2)

4.客户端通过P2连接服务器进行数据传输，端口号为e1\*256+e2，abcd为服务器的IP地址。

3.2 Ftp的命令与应答

Ftp有一些常用的命令

表3-1 Ftp常用命令

|  |  |  |
| --- | --- | --- |
| 命令码 | 命令 | 描述 |
| USER | 用户名 | 在建立连接时发送用户名 |
| PASS | 密码 | 在用户登录时提供密码 |
| CWD | 更改当前目录 | 设定目前的服务器目录 |
| CDUP | 回到上一层目录 | 回到当前目录的上一层目录 |
| QUIT | 退出登录 | 当数据发送完毕后用户退出登录 |
| PASV | 被动模式 | 该命令让客户端向服务器主动发起连接 |
| MODE | 传输模式 | 设定数据如何传输 |
| STOR | 存储 | 客户端发送文件给服务器 |
| LIST | 列表 | 获得当前目录的所有文件名以及文件相关信息 |

Ftp有一些常用的应答

表3-2 Ftp常用应答

|  |  |
| --- | --- |
| 应答 | 描述 |
| 200 | 命令成功 |
| 220 | 服务器准备好了 |
| 227 | 进入被动模式 |
| 230 | 登陆成功 |
| 250 | 文件操作完成 |
| 331 | 用户名正确，需要密码 |
| 530 | 未登录 |

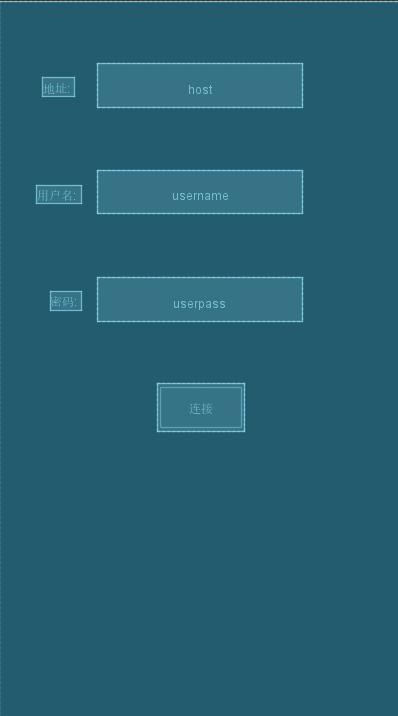
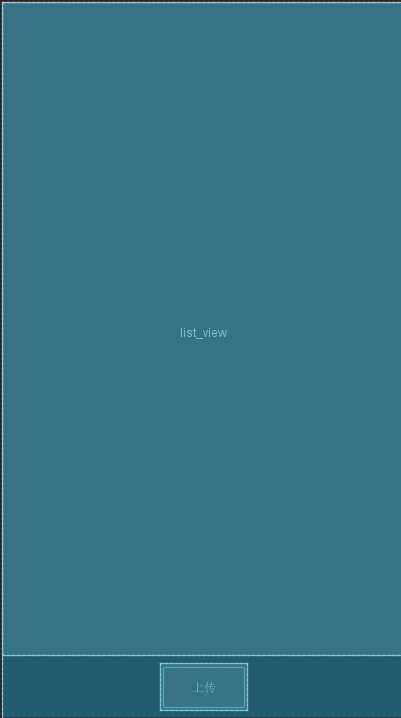
第四章 Ftp客户端实现

4.1 创建界面

Ftp客户端首先需要用户输入服务器的IP或者域名，以及用户名和密码，如果登陆成功则跳转到文件管理界面。因此需要两个界面，一个用于登陆，一个用于显示文件信息。

主界面由3个EditText、3个TextView、1个Button组成。EditText用来获取用户输入的信息，按下Button执行登陆操作。

列表界面由1个ListView、1个Button组成。ListView用来显示服务器上的文件信息，Button用来上传文件。

（a） （b）

图4-1 程序界面设计。(a)主界面；(b)列表界面

4.2 组织结构

将Ftp相关的功能放到一个Ftpclient类中，各界面通过调用这个类来完成Ftp相关的操作。

为了让每个界面都能访问Ftpclient的实例，创建了一个Global类，用于全局变量的存放和访问。对于过程中可能出现的错误，例如服务器返回错误代码等，创建一个FtpError类用于异常处理。

4.3 主界面

在主界面首先进行Ftp客户端的初始化以及权限申请。因为安卓的限制，要在主进程中使用网络等相关操作需要加上以下代码。

StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);

Login函数实现了获取用户输入的服务器IP地址或域名，用户名，密码，使用这三个参数创建了一个新的Ftpclient对象，并将它保存到全局变量中。然后尝试登陆Ftp服务器。如果登陆成功，则通过Intent调用列表界面。

4.4 列表界面

首先创建一个ArrayList用来存放文件夹以及文件的信息。在初始化时创建一个SimpleAdapter设置列表的样式。通过一个map保存文件的名字和相应的图片id，这样可以对于不同种类的文件显示不同的图片，方便区分。并且新建一个onItemClick监听器，用来实现下载文件和切换目录的功能。同时设置一个长按的监听器，通过长按可以选择剪切、重命名、删除的操作。

当用户点击列表中的文件夹时，调用Ftpclient的cwd函数，并且更新当前listview中的数据。

当用户点击列表中的文件时，则触发了下载操作。downloadDialog函数用来创建一个提示框，需要用户确定下载才会继续下载流程，防止误触。在用户确定后，因为安卓11的限制，需要调用存储访问框架（SAF）由用户自己选择下载文件保存的路径和名字。在调用SAF时，会返回一个Intent包含文件相关的信息，首先要从Intent中获取文件的Uri，Uri作为文件的标识可以从中提取出输入流或者输出流，并且可以得到文件名。

当用户点击菜单时，则可以进入菜单界面，在菜单里面可以选择上传文件、新建文件夹、粘贴、刷新、退出。选择上传文件时直接调用SAF框架由用户选择要上传的文件，将此文件上传到Ftp服务器的当前目录。而选择其他命令时则通过FtpClient的相应函数处理。

4.5 Ftpclient的实现

Ftpclient是这个客户端最核心的部分，负责了与服务器交互的所有函数。

首先需要创建BufferedReader和PrintWriter用于控制连接的输入和输出，以及两个向量用来保存文件夹名和文件名，还有其他的变量用来保存服务器地址，端口，用户名，密码等。

Connect函数创建了一个Socket用来连接服务器，并且创建了输入和输出。

Login函数用于登陆服务器，因为有些Ftp服务器会发送不止一行的欢迎信息，因此使用一个循环读取一行，直到不是以220开头为止。输出USER+用户名来请求登陆，根据返回的代码判断是否能继续登陆。然后输出PASS+密码，根据返回代码开头是否为230判断登陆是否成功。这个流程中如果有错误则抛出FtpError类型的自定义异常，里面包含错误信息。

getFileList函数用于获取当前目录下的所有文件夹和文件。因为在UNIX下的Ftp服务器可能使用二进制模式传输数据，因此首先使用TYPE A命令切换为ascii模式。然后向服务器请求使用UTF-8编码，用来防止传输中文时的乱码。如果服务器应答切换失败，则默认以GBK编码读取中文。到这里初始化完毕，先使用pasv函数进入被动模式，根据获取到的端口好创建一个新的socket连接，之后发送LIST命令请求数据，按行读入信息，选择最后一块文件名保存到对应的向量中。

Pasv函数用于进入被动模式，先想服务器发送PASV命令，如果应答不是以227开头则抛出异常。首先获取括号中有用的信息，用逗号进行分割，读取倒数第二个乘以256，再加上最后一个数字，则获得了服务器开放的端口号。

Upload函数用于文件的上传。首先发送TYPE I命令将传输模式调整为二进制传输，然后使用pasv函数进入被动模式，发送STOR+文件名开始上传。其中有一个输入流获取本地文件的信息，一个输出流用于向服务器传输文件。在传输完成之后需要使用flush函数将缓冲区的数据强制输出。

Download函数用于文件的下载，实现原理和upload函数相似。

Pwd，Mkd等进行当前目录更改、重命名文件、删除文件、新建文件的函数则不再赘述，实现原理与上面类似。

4.6 AndroidManifest的设置

为了让程序能够访问网络，需要在AndroidManifest添加

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

这样获取访问网络的权限后才能正常联网。

第五章 全文总结与展望

5.1 全文总结

在这次实验中，为了完成Android客户端的制作，自学了Java和Android，在上手实践的时候使用最新的Android 11，发现其中的内容与教程上有较大的不一致。原因是谷歌对于开发权限的收紧和对用户隐私的关注，有些功能不能像老版本一样轻松完成，而是需要使用更安全，权限更低的方式来完成。例如对于文件的操作，Android限制了开发者直接访问大部分数据，即使在用户授予了权限的情况下也不行，而是要求只能使用给每个应用单独划分的空间，或者使用SAF框架，由用户主动选择程序可以访问的位置。在学习SAF框架的时候，让我对于Android的流程有了更加清楚的了解，理解了Intent在Android中重要的作用。

5.2 程序运行截图

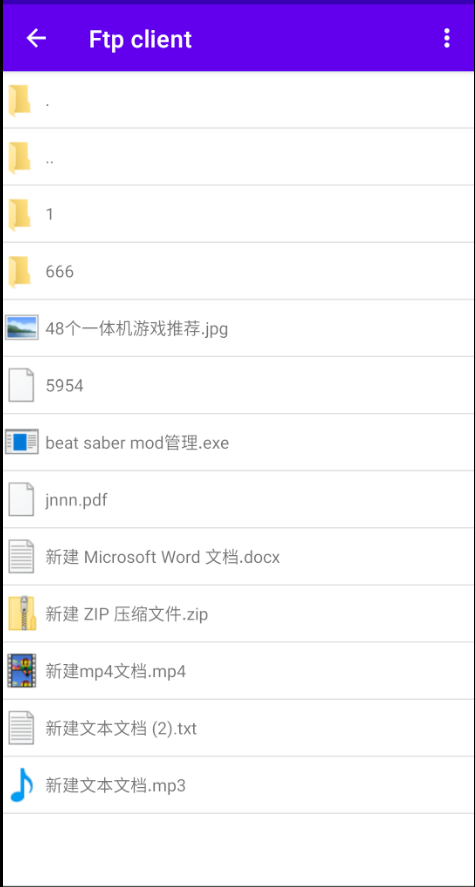
  

图5-1 程序图标 图5-2 程序初始界面 图5-3 登陆后界面

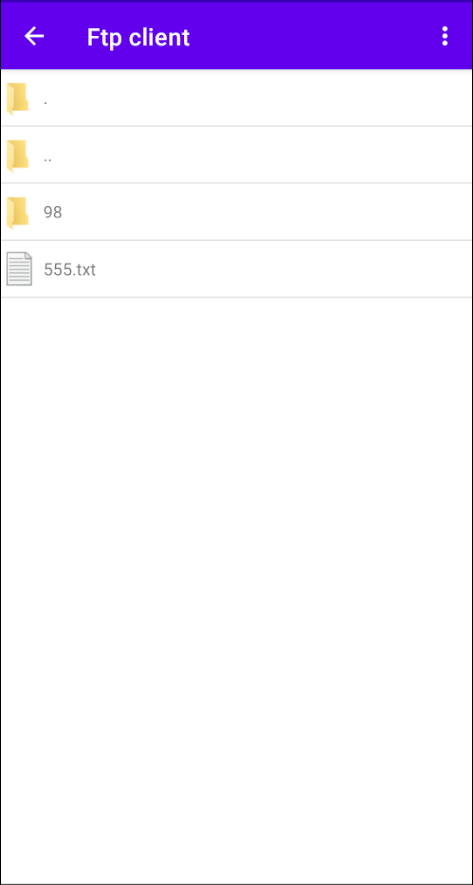
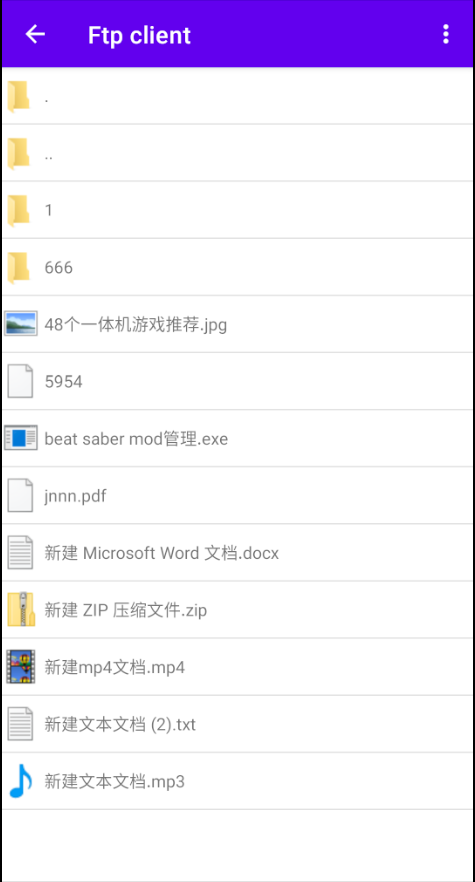
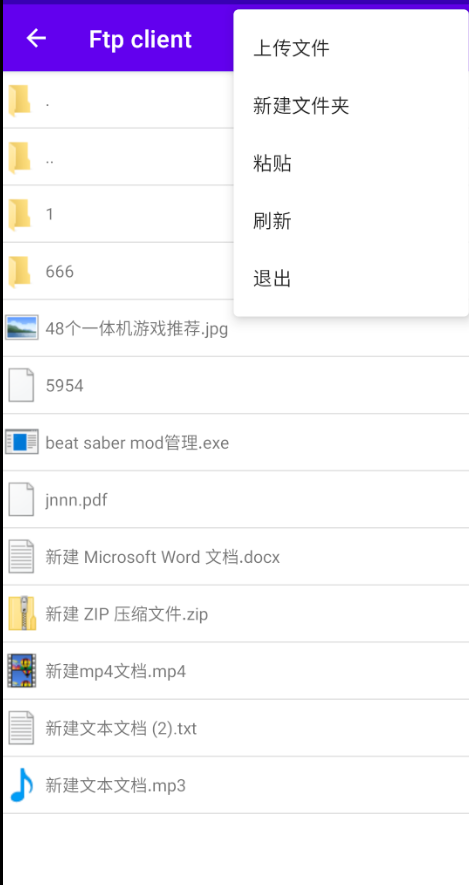
  

图5-4 点击1文件夹后 图5-5 点击..文件夹返回上级目录 图5-6 打开菜单

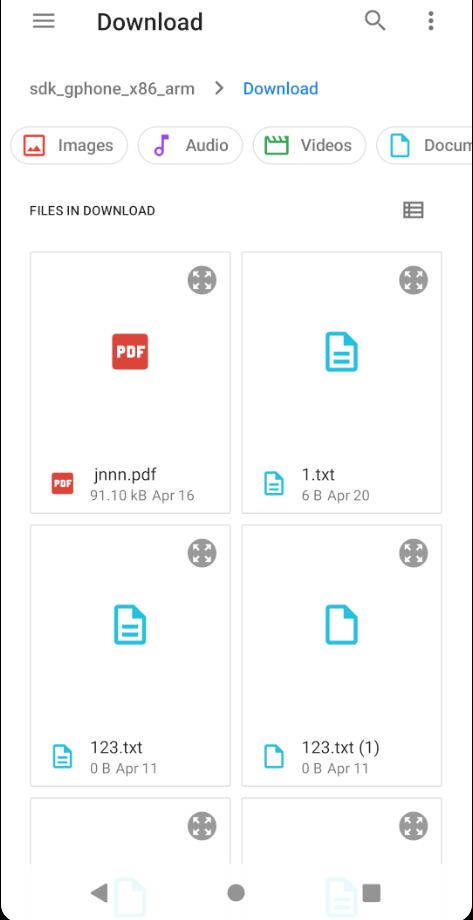
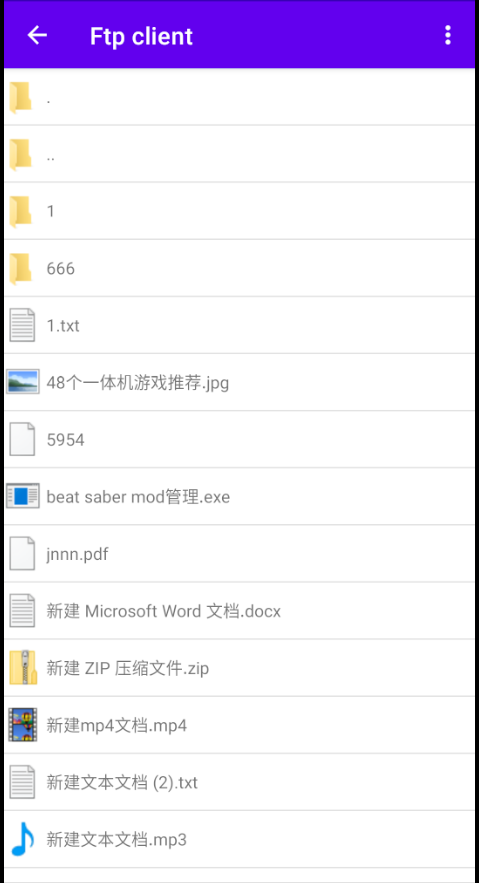
  

图5-7 点击上传文件后 图5-8 选择1.txt进行上传 图5-9 选择新建文件夹

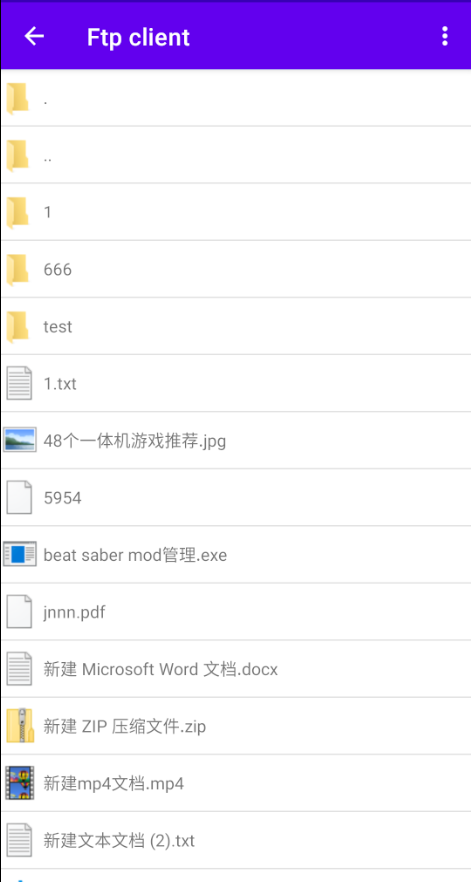
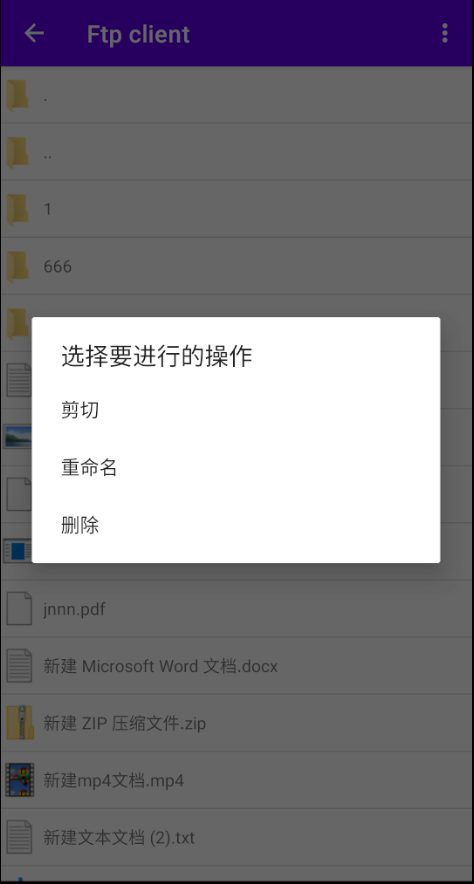
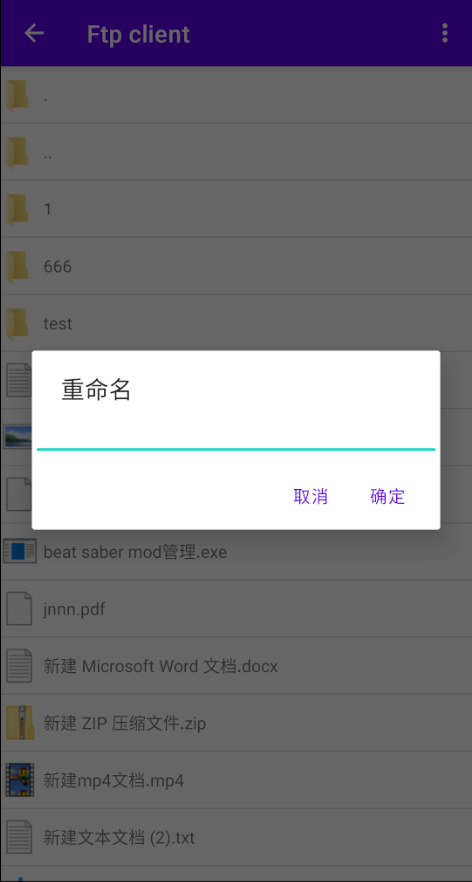
  

图5-10 创建了一个test文件夹 图5-11 对test文件夹长按 图5-12 选择重命名

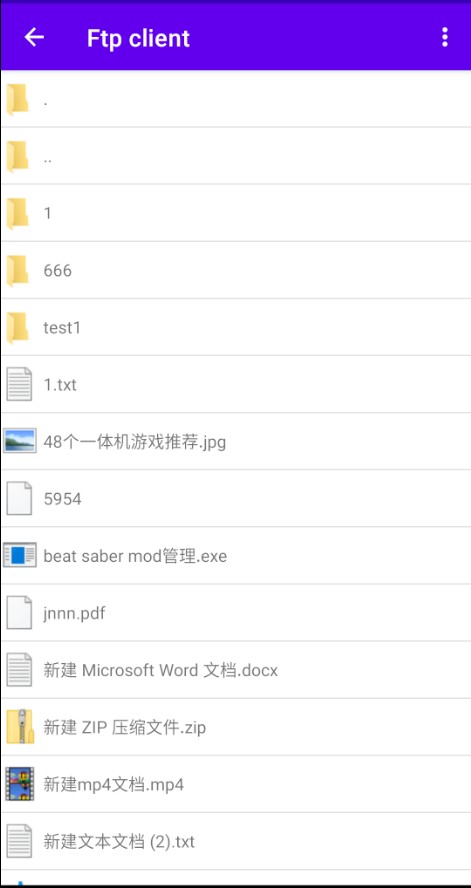
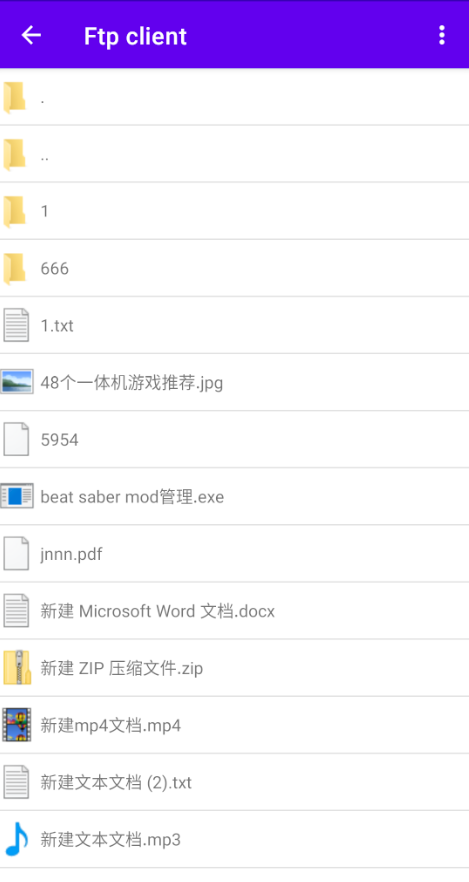
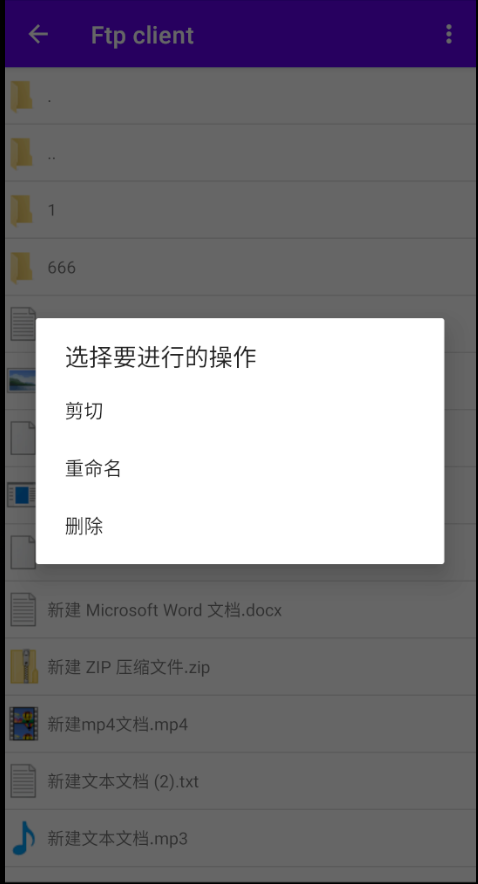
  

图5-13 重命名为test1 图5-14 对test1选择删除 图5-15 对1.txt选择剪切

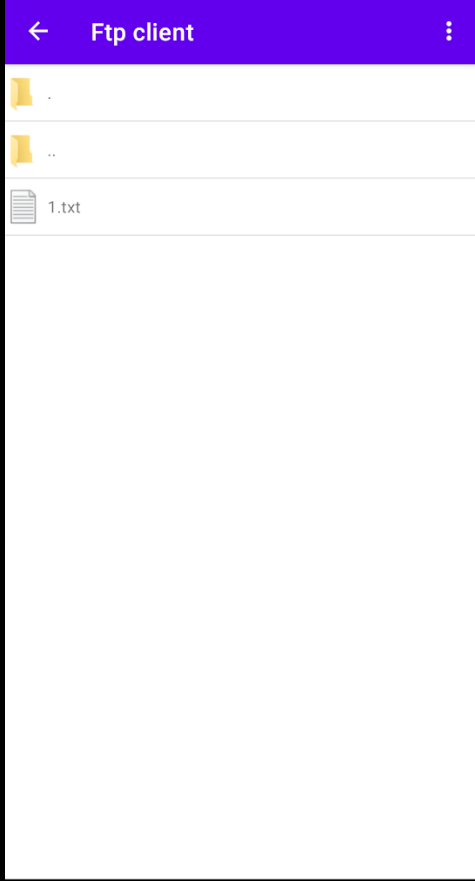
 

图5-16 进入666文件夹选择粘贴 图5-17粘贴成功

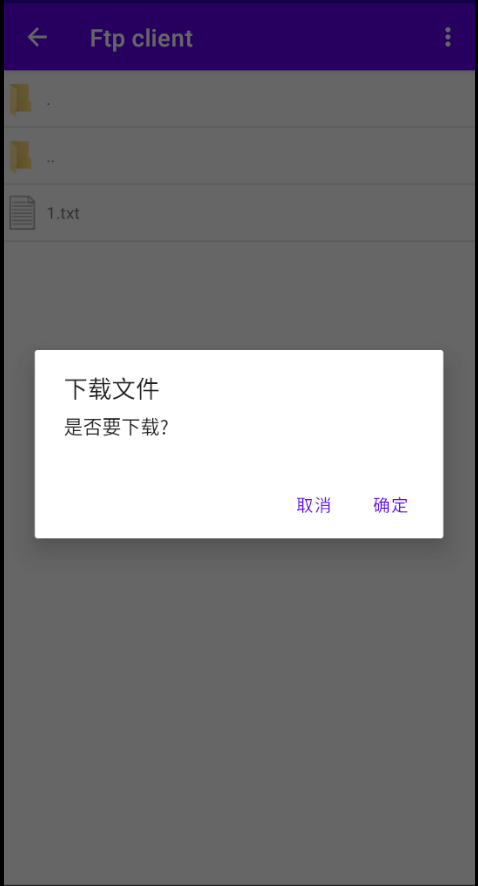
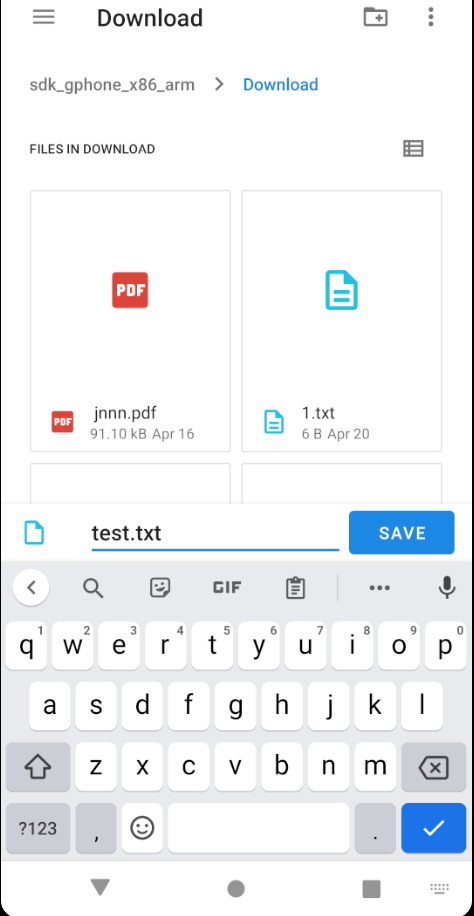
 

图5-18点击1.txt进行下载 图5-19重命名为test.txt并下载

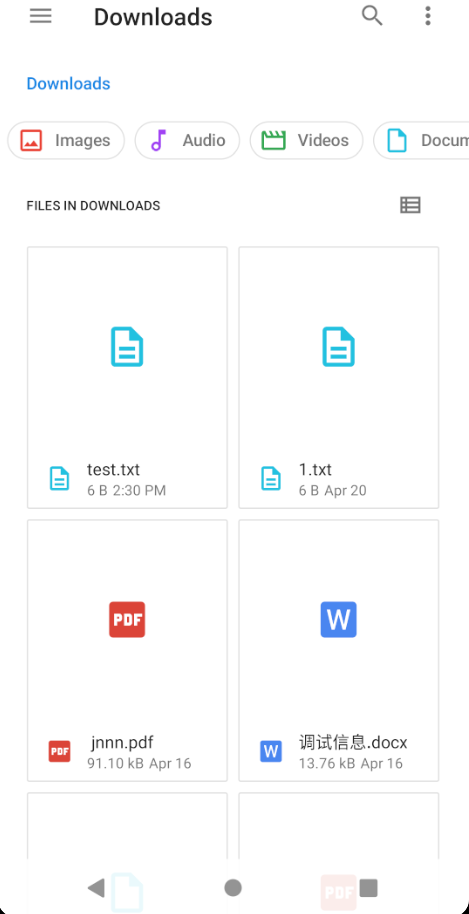


图5-20在文件浏览器中成功找到