基于多线程并发的文件传输器

姓　　名　杨子易

学　　号　2003408029

班　　级　信息资源管理

2022 年 12 月 7 日

目录

一、项目目的

学习软件部分目的是锻炼学生从事开发计算机应用系统的能力和协同工作的能力。使学生初步掌握以计算机网络通信技术为核心，主要采用 C/S、B/S结构的计算机网络应用系统开发方法，增强学生的实际编码能力，在项目开发的过程中得到以下锻炼：

- 熟悉和掌握网络编程的基本方法和步骤；
- 进一步理解 client/server 交互模式；
- 加深学生对文件传输技术的设计和实现方法的理解；
- 熟悉 socket 编程接口，掌握用 socket 编程接口开发网络应用程序的方法。

二、项目要求

根据自愿的原则自行选择题目类型，功能根据自己的理解进行扩充。本次实验选择第七题——文件传输，该题目相关要求如下：

- 实现图形用户界面；
- 可进行双向文件传输（文件类型不限）；
- 可根据需要对文件内容进行加密后再传输；
- 可进行密钥交换；
- 可实现断点续传。

三、实现内容展示

3.1 项目亮点

1 互联网下的文件传输

大多数利用 socket 进行网络编程的示例程序都只能实现若干虚拟机之间的通信，或几台主机在一个局域网下进行通信。但这样的通信显然没有实际意义。本项目通过购买一个公网 ip 地址的端口，将该端口映射到本地主机的一个端口。这样，只要服务器也绑定了本地主机的这个端口，客户端便可以通过公网 ip 的 指定端口，将数据转发到本地服务器，从而实现互联网下的文件传输。

2 多线程并发传输

本项目利用多线程技术，使服务器能够与多个客户端进行连接，并且对于每个客户端而言，都可以同时下载多个文件。另外，在文件下载过程中，通过严谨的同步控制，用户可以对某个文件暂停传输，并在适合的时候恢复传输，也可以取消一批文件的下载。

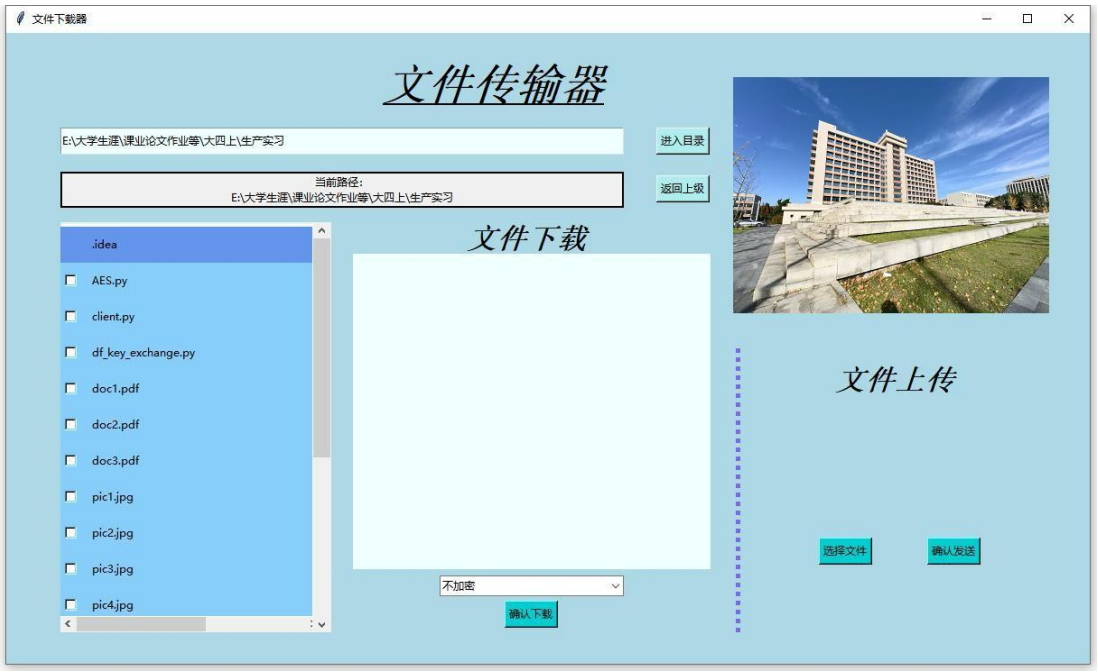
3 严谨的逻辑控制

项目在实际应用过程中，用户很可能进行一些不合理的操作或输入某些非法内容，使程序崩溃。针对这一点，本程序进行了严谨的逻辑控制，对任何非法的操作给　相应的提示并拒绝这一请求。此外，在文件传输的过程中强行关闭程序，程序会依次断开所有 socket 连接后再退　，而不是仅仅退　界面而保留原有的传输。

3.2 项目展示

1 图形界面介绍

客户端与服务器连接后，默认显示的是服务器端程序所在的路径信息，以及该路径下的所有文件信息。左侧的文件目录中，浅蓝色表示一个可以下载的具体文件，深蓝色表示一个文件夹。用户可以选中若干个具体文件进行下载，也可以点击文件夹图标，进入相应的文件夹。同时，用户还可以通过上方的输入栏切换路径，以进入服务器的任意文件夹下。界面的右下角是文件上传的区域，用户可以上传本地任意文件到服务器。



2 下载过程

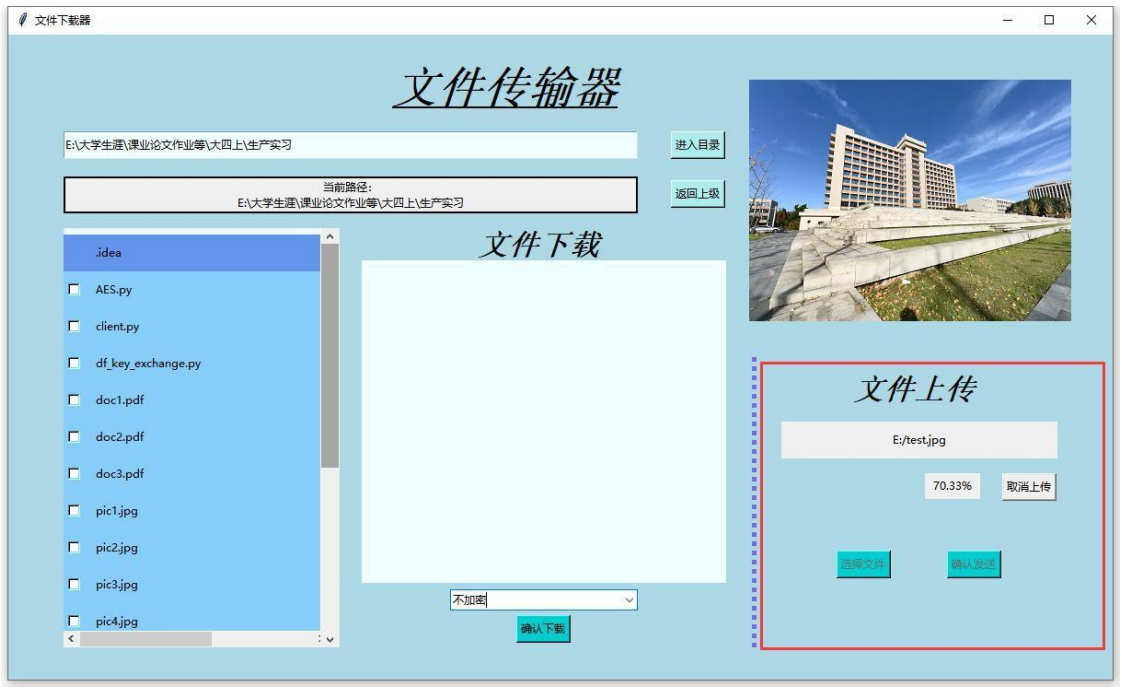
用户一次可以选择若干个文件进行下载，下载过程中程序会随时显示每个文件的下载进度。对于每个文件，程序都支持断点续传的功能。在下载完成前，用户可以选择暂停下载，并在适当的时候恢复下载。另外，下载过程支持加密传输，

具体加密类型分为 AES128、AES192 以及 AES256，在加密传输前，程序会根据加密类型，利用 diffi-hellma 协议进行密钥交换，并使用该密钥进行加解密。假设要从服务器下载 pic1、pic2、pic3、pic4、pic5 这五个文件，并采用 AES128 进行加密，传输过程的效果如下：

可见，此时 pic1 和 pic5 处于暂停状态，其余三个文件处于正在传输的状态。此外，由于文件大小不同，故传输的进度也不尽相同。

3 上传过程

上传过程和下载过程的实现方法几乎完全一致，故本程序只对上传功能进行了简单的设计，支持用户每次上传一个任意类型的文件。上传过程中随时展示传输进度，并可以在中途取消上传。假设用户选择 E:/test.jpg 进行上传，传输过程的效果如下：



4 端口映射信息

本项目在 www.ngrok.cc 网站上购买一个用于 tcp 连接的公网 ip 端口，来进行端口映射。服务器运行的同时开启隧道，实现互联网下的文件传输。端口映射服务如下图。

可以看到，vipgz4.91tunnel.com 的某个端口映射到了本地的 12345 端口。

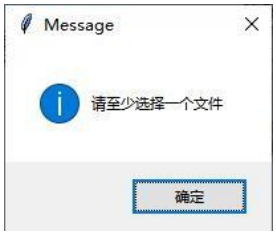
5 避免非法操作

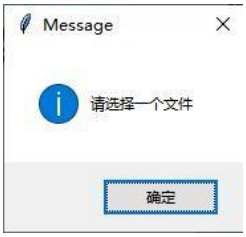
本程序认为的非法操作有：

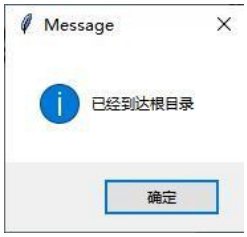
- 未选择文件就点击“确认下载”按钮；

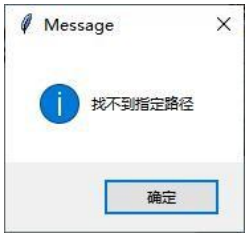
- 未选择文件就点击“确认发送”按钮；
- 已经到达根目录后再点击“返回上级”按钮；
- 在输入栏中输入空路径或不存在的路径。

这四种操作自上而下分别会弹 以下提示：









此外，程序在运行过程中还通过一些明确的逻辑控制，防止一些用户可能的的非法操作。例如：某个文件正在下载时，不可以点击“续传”按钮；文件暂停时，不可以再点击“暂停”按钮等。由于这种控制较多且逻辑较为简单，故此处不再具 体展示。

四、遇到的问题及解决方案

4.1 利用 socket 传输大文件

利用 socket 传输信息时，一次性无法传输太大的文件，必须将文件内容进行分段传输。这就需要在文件具体传输前，发送方先计算文件大小并告知接收方，之后接收方不断接收经过分段的文件内容，直到接收的大小和文件大小相等。

在这个过程中，假设接收方利用 curSocket 与服务器通信，则接收方会方使用 curSocket.recv(1024) 函数分段接收文件内容。乍一看，这个函数的作用是每次接收 1024 个字节大小的内容。然而，实际操作的时候发现文件传输经常失败。

后查阅资料得知，curSocket.recv(1024) 表示最多接收 1024 字节，但实际接收到的数据大小会根据网络状况等因素而定，通常是 256 - 1024 字节不等。因此，传输过程中必须根据实际接收到的数据量大小来进行判断。编写如下代码，成功解决了传输单个大文件的问题。

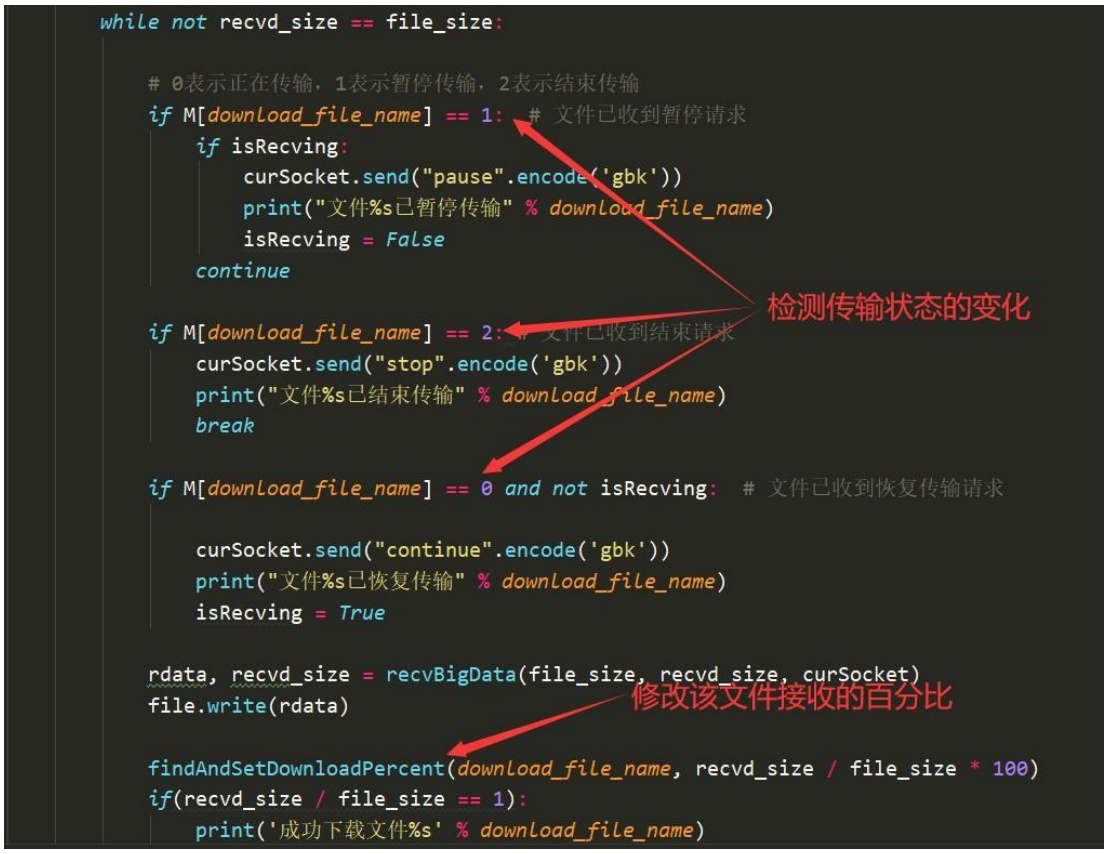
```
"""
    用来接收大文件。
    socket每次接受数据大小有限制，且每次接受数据的大小不固定，必须根据实际接收到的数据大小，进行传输控制。
"""
def recvBigData(file_size, recvd_size, curSocket):

    if file_size - recvd_size > 1024:
        rdata = curSocket.recv(1024)
    else:
        rdata = curSocket.recv(1024)

    recvd_size += len(rdata)  # 发送1024字节，但未必接收1024字节，故必须用接收到的实际长度进行文件拼接
    return rdata, recvd_size
```

4.2 多线程下的同步控制

整个项目编写过程中，最大的困难就在于多线程下的同步控制。考虑到文件传输过程中可能的一些状态变化（暂停，续传，取消等），必须对这些状态进行严谨的控制。解决这一问题的关键思路是，明确多线程下程序并行执行的顺序，并对互斥资源进行上锁。请看一段接收文件的部分代码：



这里，M 是一个字典类型的变量，其中 key 是文件名，value 是文件传输的状态。0 表示正在传输，1 表示暂停传输，2 表示取消传输。一个文件的传输过程中由两个线程来控制，一个线程用来下载文件，另一个线程随时检测该文件的状态变化。

另外，上面代码中的倒数第三行，使用了 findAndSetDownloadPercent 函数来修改文件接收的百分比。该函数的函数体如下：

```
"""
    多线程环境下，保证downloadPercent变量的同步互斥关系。
    网速较慢时，recv函数中recvBigData接收数据慢，如果interrupt将downloadPercent清空后，
recv函数重新设置downloadPercent的值，则会导致后续程序混乱，造成死锁。
"""
def findAndSetDownloadPercent(file_name, percent):

    global downloadPercent, L1
    L1.acquire()
    if file_name in downloadPercent.keys():
        downloadPercent[file_name] = percent
    L1.release()
```

其中，downloadPercent 变量存储了每个文件所对应的下载进度。这个函数的目的是，当且仅当 downloadPercent 变量中有该文件时，才对其进行修改。正如上图中的注释那样，之所以这样做，是因为如果文件传输过程中用户取消了本次下载，则会将 downloadPercent 变量中对应该文件的信息删除。此时，如果在接收文件的代码中又重新设置了该文件对应的 downloadPercent，则会导致传输混乱。最简单的情况是，用户取消了一批文件的下载后，又重复下载这一批文件。这就会因为 downloadPercent 中存在上一次下载的一些残留内容，导致程序崩溃。

此外，发送方也同样需要对文件的传输状态进行明确的控制，其思路和接收方类似，故不再赘述。

五、总结与感想

本项目进一步增强了我对较大型项目整体架构的规划和调试能力，加深了我对于 socket 网络编程的理解，同时也让我掌握了使用 tkinter 来进行图形化界面编程的一些基本操作。在实现并行文件传输的过程中，很大程度上增强了我多

线程编程的思路和能力，以及遇到各种问题时的调试方法。总之，本项目在很多方面都给了我极大的锻炼，让我收获颇丰。