信息对抗技术大作业：木马基本功能的实现

2213092 邓沐尘 信息安全

实验要求

本实验中，在设置好源、目的 IP 地址后，便可以通过 client 发送指令，对 server 进行操作，实现木马的基本功能

1. 输出字符串：在服务端输出字符串
2. 关机：令服务端(server)主机在 60 秒内关机
3. 取消关机：在关机时限(60 秒)内，可以取消服务端主机关机
4. 获取 C 盘文件列表：获取此时服务端(server)主机的 C 盘列表
5. 截屏：截取此时服务端(server)主机的桌面图像
6. 删除：在服务端(server)主机 C 盘列表中选取并删除指定文件
7. 上传：在客户端(Client)主机中选取指定文件，将其内容传送并保存至文件"myFile.txt"
8. 下载：在服务端(server)主机 C 盘列表中选取指定文件，将其内容拷贝至文件"myFile.txt"，并保存至所选路径下

要求：

写出实验报告，含程序代码和截图，word或pdf格式

实验环境

PyCharm Community Edition 2024.1.1

Windows 11操作系统

Python解释器是anaconda3自带的

代码实现

客户端代码如下（`Client.py`）

```
import socket
import tkinter as tk
from tkinter import filedialog, simpledialog, messagebox
from PIL import ImageGrab
import io
import os


class ClientGUI:
    def __init__(self, master):
        self.master = master
        master.title("Client GUI")

        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client_socket.connect(('localhost', 12345))

        # GUI components

        self.echo_button = tk.Button(master, text="Echo Message to Server",
command=self.send_echo_command)
        self.echo_button.pack()

        self.send_screenshot_button = tk.Button(master, text="Send Screenshot",
command=self.send_screenshot)
        self.send_screenshot_button.pack()

        self.shutdown_button = tk.Button(master, text="Schedule Shutdown",
command=self.send_shutdown_command)
        self.shutdown_button.pack()

        self.cancel_shutdown_button = tk.Button(master, text="Cancel Shutdown",

 command=self.send_cancel_shutdown_command)
        self.cancel_shutdown_button.pack()

        self.list_c_drive_button = tk.Button(master, text="List C Drive on
Server", command=self.list_c_drive_on_server)
        self.list_c_drive_button.pack()


        self.delete_button = tk.Button(master, text="Delete File on Server",
command=self.delete_file_on_server)
        self.delete_button.pack()

        self.upload_button = tk.Button(master, text="Upload File to Server",
command=self.upload_file_to_server)
        self.upload_button.pack()

        self.download_button = tk.Button(master, text="Download File from
Server",
                                         command=self.download_file_from_server)

        self.download_button.pack()

        self.text_widget = tk.Text(master, height=20, width=80)  #用于显示C盘文件的
文本框
        self.text_widget.pack()



    def send_shutdown_command(self):
        self.client_socket.sendall("SHUTDOWN:".encode())
        messagebox.showinfo("Command Sent", "Shutdown has been scheduled.")

    def send_cancel_shutdown_command(self):
        self.client_socket.sendall("CANCEL_SHUTDOWN:".encode())
        messagebox.showinfo("Command Sent", "Shutdown has been cancelled.")
    def send_command(self):
        command = self.entry.get()
        if command:
            self.client_socket.sendall(command.encode())
            self.entry.delete(0, tk.END)

    def delete_file_on_server(self):
        file_path = simpledialog.askstring("Delete File",
                                           "Enter the file path on the server
(e.g., C:\\path\\to\\file.txt):")
        if file_path:
            self.client_socket.sendall(f"DELETE:{file_path}".encode())

    def upload_file_to_server(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            file_size = os.path.getsize(file_path)
            self.client_socket.sendall(f"UPLOAD:{file_size}".encode())
            self.client_socket.recv(1024)  # Wait for server ready
            with open(file_path, 'rb') as f:
                while True:
                    data = f.read(4096)
                    if not data:
                        break
                    self.client_socket.sendall(data)
            messagebox.showinfo("File Uploaded", "File uploaded successfully.")

    def download_file_from_server(self):
        file_path = simpledialog.askstring("Download File",
                                           "Enter the file path on the server
(e.g., C:\\path\\to\\file.txt):")
        if file_path:
            self.client_socket.sendall(f"DOWNLOAD:{file_path}".encode())
            file_size_data = self.client_socket.recv(16).decode()
            file_size = int(file_size_data)
            self.client_socket.sendall(b'READY')  # Send ready to server
            file_data = b''
            remaining_bytes = file_size
            while remaining_bytes > 0:
                data = self.client_socket.recv(4096)
                file_data += data
                remaining_bytes -= len(data)
            save_path = filedialog.asksaveasfilename(defaultextension=".bin",
filetypes=[("Binary files", "*.bin")])
            if save_path:
                with open(save_path, 'wb') as f:
                    f.write(file_data)
                messagebox.showinfo("File Downloaded", "File downloaded
successfully.")

    def send_screenshot(self):
        screenshot = ImageGrab.grab()
        output = io.BytesIO()
        screenshot.save(output, format='PNG')
        screenshot_data = output.getvalue()

        try:
            # 发送"SCREENSHOT:"通知服务端
            self.client_socket.sendall("SCREENSHOT:".encode())
            # 发送文件大小（字符串形式），后面跟着换行符
            self.client_socket.sendall(str(len(screenshot_data)).encode() +
b'\n')
            # 发送截图数据
            self.client_socket.sendall(screenshot_data)
            messagebox.showinfo("Screenshot Sent", "Screenshot has been sent
successfully.")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def send_echo_command(self):
        echo_message = simpledialog.askstring("Echo Message", "Enter a message to
echo on the server:")
        if echo_message:
            self.client_socket.sendall(f"ECHO:{echo_message}".encode())

    def list_c_drive_on_server(self):
            self.client_socket.sendall("LIST_C_DRIVE".encode())
            self.text_widget.delete('1.0', tk.END)  # 清空之前的文本

            while True:
                data = self.client_socket.recv(4096).decode().rstrip('\n')
                if data == 'END_OF_LIST':
                    break
                self.text_widget.insert(tk.END, data + '\n')  # 将文件路径插入到Text
widget中

    def close_connection(self):
        self.client_socket.close()
        self.master.destroy()


root = tk.Tk()
client_gui = ClientGUI(root)
root.protocol("WM_DELETE_WINDOW", client_gui.close_connection)  # Close
connection on window close
root.mainloop()
```

服务端代码如下(`Server.py`)

```
import socket
import threading
import os
import time
from PIL import ImageGrab
import io

shutdown_scheduled = False
```

```python
shutdown_thread = None


def shutdown_computer():
    global shutdown_scheduled
    time.sleep(5)  # 假设有一个5秒的延迟来模拟关机前的准备时间
    if shutdown_scheduled:
        print("Shutting down the computer...")
        # 在Windows系统上执行关机命令
        os.system('shutdown -s -t 60')  # 60s延迟，再次之前可随时取消


def cancel_shutdown():
    global shutdown_scheduled, shutdown_thread
    shutdown_scheduled = False
    if shutdown_thread and shutdown_thread.is_alive():
        print("Attempting to cancel shutdown...")
        os.system('shutdown -a')  # 取消关机命令

def delete_file(file_path):
    if os.path.exists(file_path):
        os.remove(file_path)
        print(f"File {file_path} deleted.")
    else:
        print(f"File {file_path} does not exist.")


def save_file(data, file_name="myFile.txt"):
    with open(file_name, 'wb') as f:
        f.write(data)
    print(f"File {file_name} saved.")


def send_file(conn, file_path):
    if os.path.exists(file_path):
        file_size = os.path.getsize(file_path)
        conn.sendall(str(file_size).encode())  # 发送文件大小
        conn.recv(1024)  # 等待客户端确认
        with open(file_path, 'rb') as f:
            while True:
                data = f.read(4096)
                if not data:
                    break
                conn.sendall(data)
        print(f"File {file_path} sent.")
    else:
        print(f"File {file_path} does not exist.")

def save_screenshot(conn, file_name="screenshot.png"):
    with open(file_name, 'wb') as f:
        while True:
            data = conn.recv(4096)
            if not data:
                break
            f.write(data)
    print(f"Screenshot {file_name} saved.")
```

```
def handle_client(conn):
    global shutdown_scheduled, shutdown_thread
    while True:
        try:
            data = conn.recv(1024).decode()
            if not data:
                break

            if data.startswith("DELETE:"):
                file_path = data[len("DELETE:"):].strip()
                delete_file(file_path)  # 假设这个函数用于删除文件
            elif data.startswith("UPLOAD:"):
                file_size_data = conn.recv(16).decode()
                file_size = int(file_size_data)
                conn.sendall(b'READY')  # 发送确认给客户端，表示准备好接收文件内容
                file_data = b''
                remaining_bytes = file_size
                while remaining_bytes > 0:
                    packet = conn.recv(4096)
                    file_data += packet
                    remaining_bytes -= len(packet)
                save_file(file_data)  # 保存文件内容到myFile.txt
            elif data.startswith("DOWNLOAD:"):
                file_path = data[len("DOWNLOAD:"):].strip()
                send_file(conn, file_path)  # 发送文件内容给客户端
            elif data.startswith("ECHO:"):
                echo_message = data[len("ECHO:"):].strip()
                print(echo_message)  # 输出字符串
            elif data == "LIST_C_DRIVE":
                # 遍历C盘目录并发送文件列表
                c_drive = "C:\\"  # Windows系统的C盘路径
                for root, dirs, files in os.walk(c_drive):
                    for file in files:
                        file_path = os.path.join(root, file)
                        # 这里可以根据需要过滤掉系统文件或隐藏文件
                        # 发送文件路径给客户端
                        conn.sendall(file_path.encode() + b'\n')
                conn.sendall(b'END_OF_LIST\n')  # 发送结束标记
            elif data == "SHUTDOWN:":
                if not shutdown_scheduled:
                    shutdown_scheduled = True
                    shutdown_thread = threading.Thread(target=shutdown_computer)
                    shutdown_thread.start()
                    print("Shutdown scheduled.")
            elif data == "CANCEL_SHUTDOWN:":
                cancel_shutdown()
                print("Shutdown canceled.")
            elif data.startswith("SCREENSHOT:"):
                    # 接收文件大小（字符串形式），直到遇到换行符
                    file_size_data = b''
                    while True:
                        size_byte = conn.recv(1)
                        if not size_byte or size_byte == b'\n':
                            break
                        file_size_data += size_byte
                    file_size = int(file_size_data.decode())

                    # 接收截图数据并保存为文件
                    with open("screenshot.png", 'wb') as f:
                        received_bytes = 0
                        while received_bytes < file_size:
                            data = conn.recv(4096)
                            f.write(data)
                            received_bytes += len(data)
                    print("Screenshot saved.")
                    print("Current working directory:", os.getcwd())#文件保存位置
            else:
                print(f"Client Commend: {data}")
        except ConnectionResetError:
            break
    conn.close()

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('localhost', 12345))
    server.listen(5)
    print("Server started.")
    while True:
        client, addr = server.accept()
        print(f"Client connected from {addr}.")
        client_thread = threading.Thread(target=handle_client, args=(client,))
        client_thread.start()


if __name__ == "__main__":
    start_server()
```

下面对以上代码做一些解释：

客户端代码 (`ClientGUI`)

初始化 (`__init__`)

- 初始化Tkinter窗口，并设置标题。
- 创建一个socket对象，并连接到服务端的`localhost`和端口`12345`。

  （客户端地连接端口每次都会改变，服务端地端口设置也可以自己调整，端口不冲突即可）
- 创建多个GUI组件，包括按钮和文本输入框，用于发送命令和显示消息。
- 创建一个`Text` widget (`self.text_widget`) 用于显示C盘的文件列表。

功能按钮命令

- **Echo Message to Server**: 弹出一个对话框让用户输入消息，并将消息发送到服务端，服务端会打印这条消息。
- **Send Screenshot**: 捕获屏幕截图，并发送到服务端保存。

  （为了方便找到图片，我设置了显示保存路径）

- **Schedule Shutdown**: 发送命令到服务端以计划关机。
- **Cancel Shutdown**: 发送命令到服务端以取消计划的关机。
- **List C Drive on Server**: 发送命令到服务端请求C盘的文件列表，并在GUI的`Text` widget中显示。
- **Delete File on Server**, **Upload File to Server**, **Download File from Server**: 这些命令允许用户在服务端执行文件操作。

发送和接收函数

- **send_echo_command**, **send_screenshot**, **list_c_drive_on_server** 等函数用于发送特定的命令或数据到服务端。
- **list_c_drive_on_server** 函数特别重要，因为它接收从服务端发送的文件列表，并显示在GUI的`Text` widget中。

服务端代码

初始化 (`start_server`)

- 创建一个socket对象，并绑定到`localhost`和端口`12345`。
- 开始监听连接，并为每个新连接创建一个新的线程来处理。

处理客户端 (`handle_client`)

- 在一个无限循环中，服务端接收来自客户端的命令或数据。
- 根据接收到的数据执行相应的操作，如删除文件、发送文件、接收截图、输出字符串到控制台、发送C盘文件列表等。
- 对于发送C盘文件列表的命令，服务端遍历C盘，并将每个文件的路径发送给客户端，直到发送完整个列表。

特定功能处理

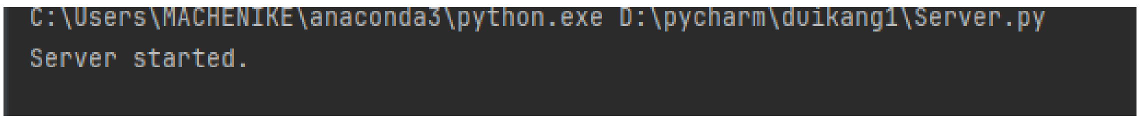
- **ECHO**: 打印从客户端接收到的字符串。
- **LIST_C_DRIVE**: 遍历C盘，并将文件列表发送给客户端。

  特别注意：遍历C盘可能需要管理员权限，尤其是对于系统文件和隐藏文件
- **SCREENSHOT**: 接收截图的大小和数据，并保存到本地文件。
- **其他命令**: 如DELETE、UPLOAD、DOWNLOAD等，都根据命令的内容执行相应的文件操作

实验结果截图

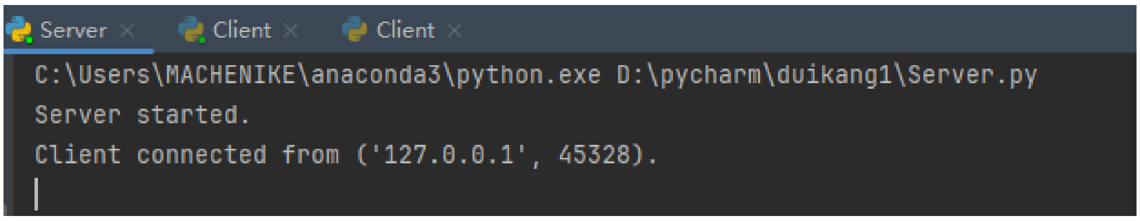
下面我们逐步验证各项功能是否实现

首先我们运行服务端，



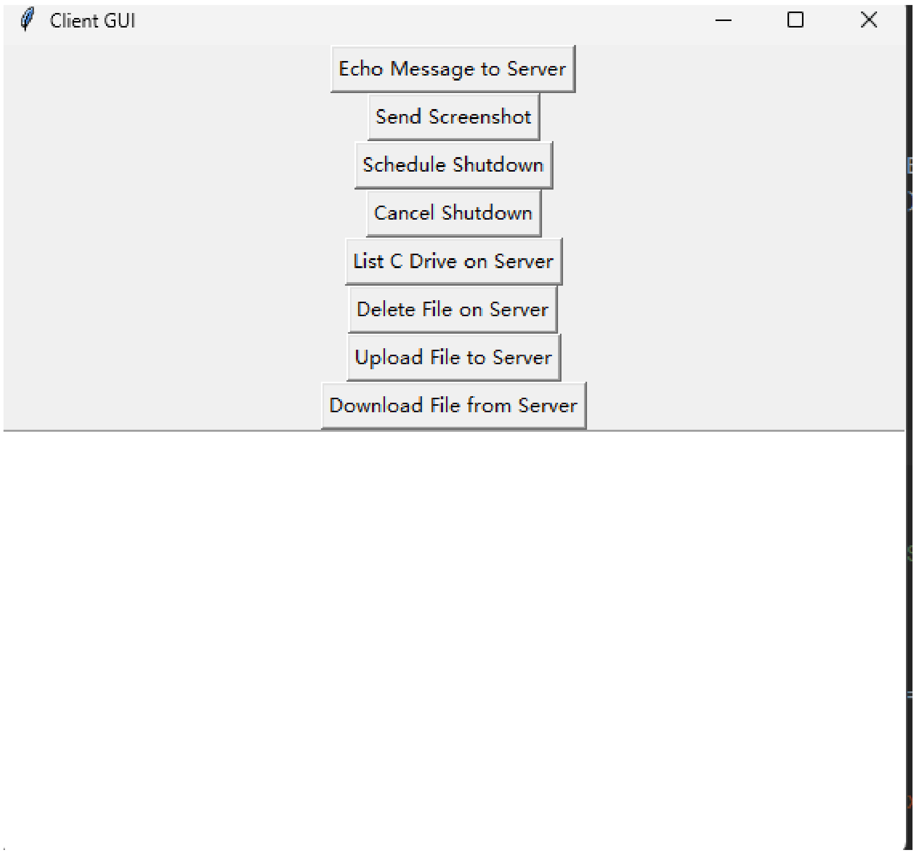
```
C:\Users\MACHENIKE\anaconda3\python.exe D:\pycharm\duikang1\Server.py
Server started.
```

然后再运行客户端，成功在服务端和客户端之间建立连接后，会显示客户端的IP和端口



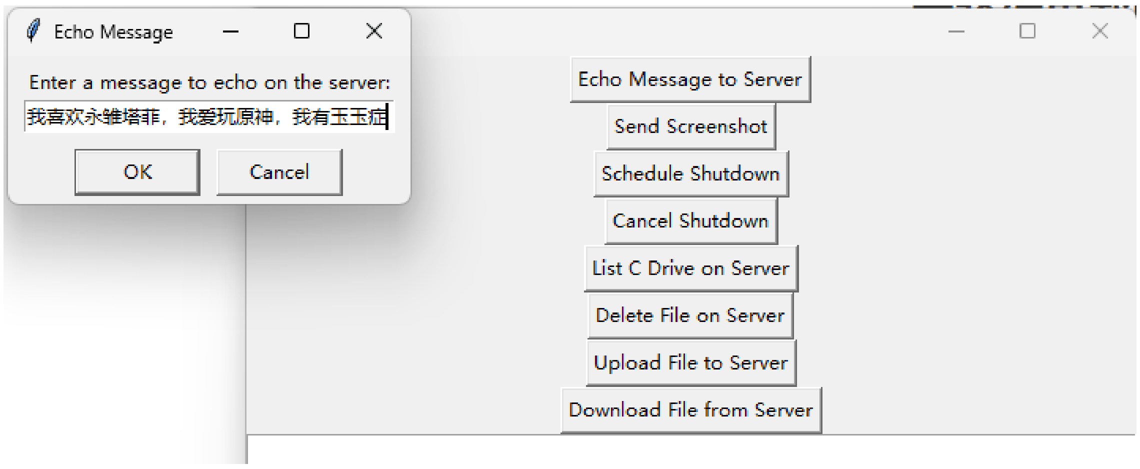
```
C:\Users\MACHENIKE\anaconda3\python.exe D:\pycharm\duikang1\Server.py
Server started.
Client connected from ('127.0.0.1', 45328).
```

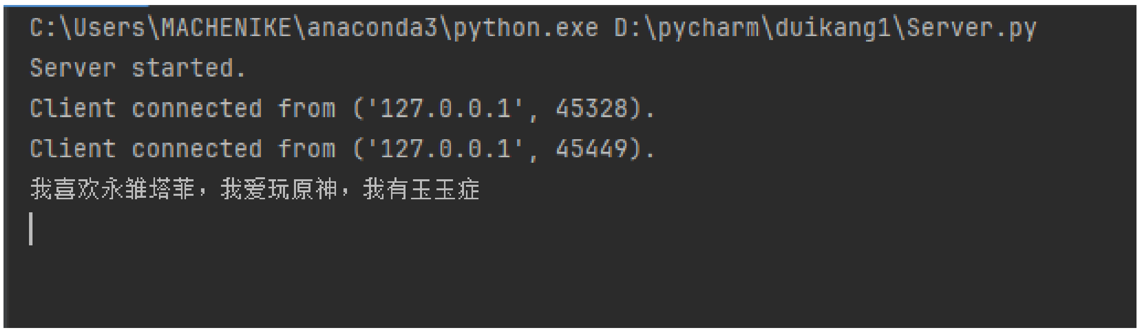
客户端窗口（Client GUI）显示效果如下，可以看到一串按钮和一个文本框



1.输出字符串：在服务端输出字符串

在Client GUI中点击第一个按钮会跳出一个空白文本框，在里面输入字符串即可发送至服务端，我们可以在服务端的命令行中看到发送的内容



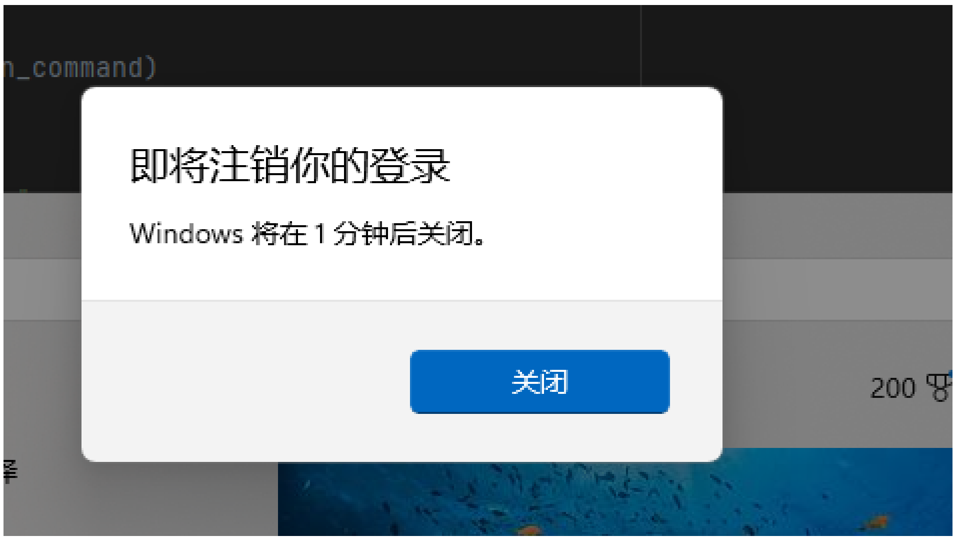


```
C:\Users\MACHENIKE\anaconda3\python.exe D:\pycharm\duikang1\Server.py
Server started.
Client connected from ('127.0.0.1', 45328).
Client connected from ('127.0.0.1', 45449).
我喜欢永雏塔菲，我爱玩原神，我有玉玉症
```

2.关机：令服务端(server)主机在 60 秒内关机

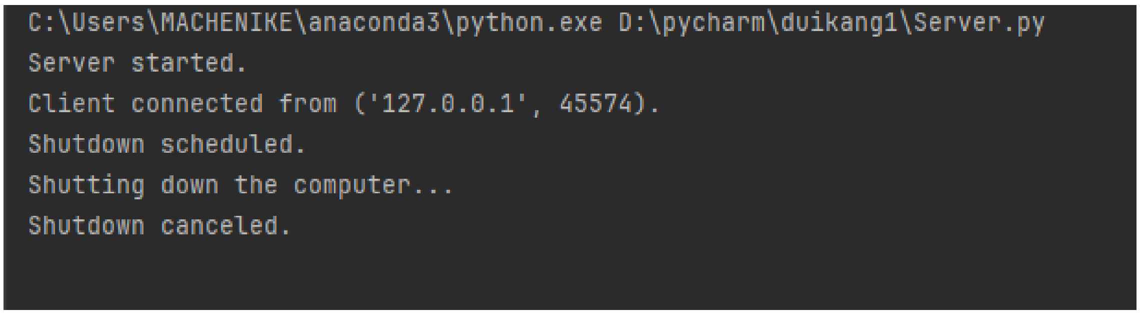
3.取消关机：在关机时限(60 秒)内，可以取消服务端主机关机。

点击第三个按钮程序会让计算机在一分钟后关机，此时Windows 11系统会给出弹窗提示，在计时完成前可以随时通过第四个按钮取消关机



即将注销你的登录

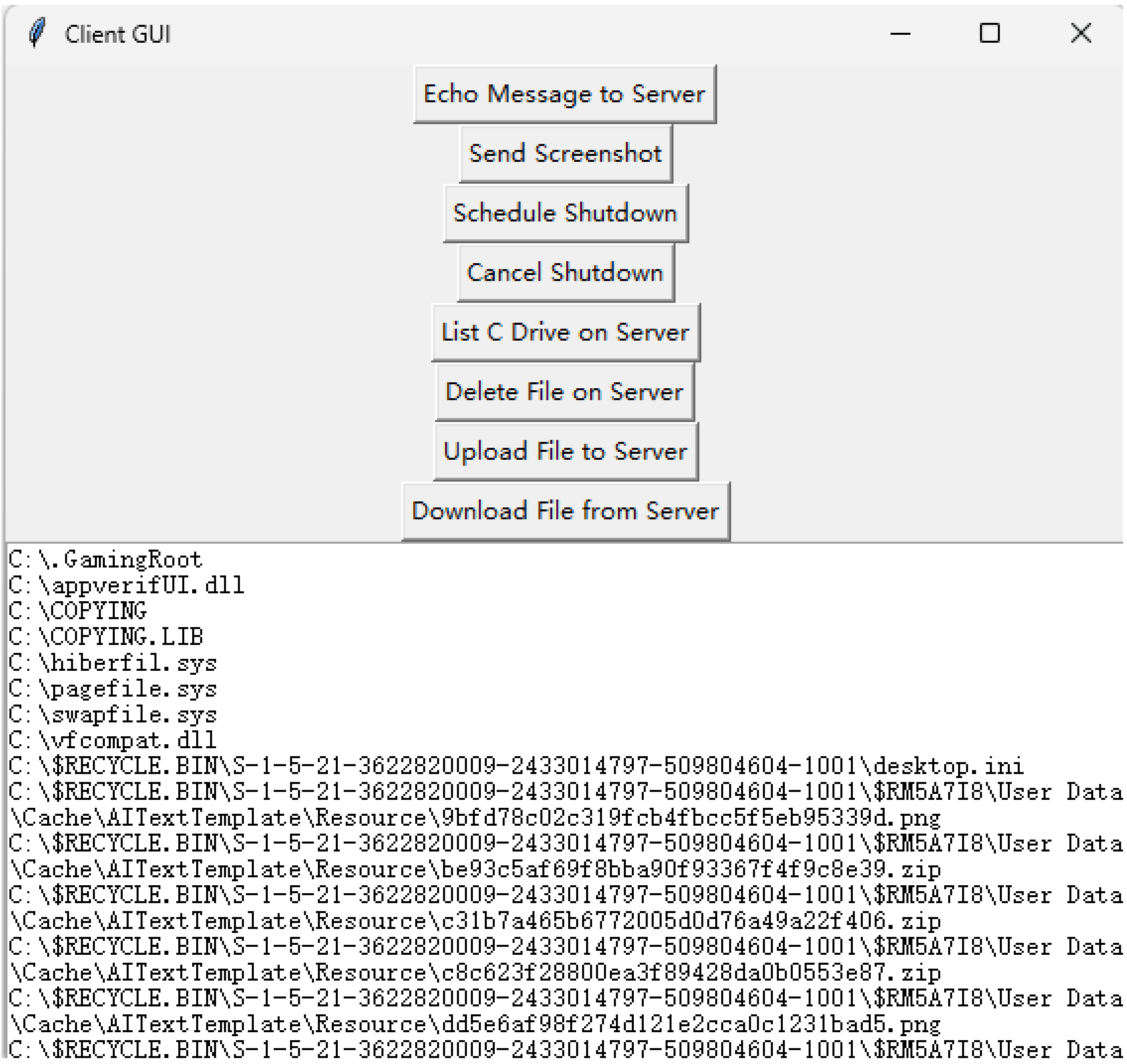
Windows 将在 1 分钟后关闭。



```
C:\Users\MACHENIKE\anaconda3\python.exe D:\pycharm\duikang1\Server.py
Server started.
Client connected from ('127.0.0.1', 45574).
Shutdown scheduled.
Shutting down the computer...
Shutdown canceled.
```

4.获取 C 盘文件列表：获取此时服务端(server)主机的 C 盘列表

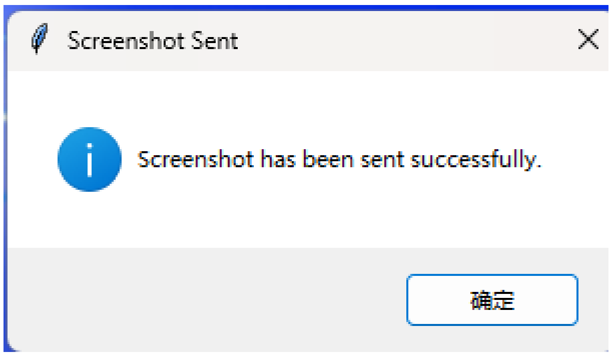
本来是将C盘文件列表输出在命令行的，为了方便查看和保存，我就将它输出在了Client GUI的文本框中
因为C盘文件比较多，该命令的执行时间比较长，可能还会导致Client GUI卡死，我尝试使用多线程优化，但是效果不明显，还整出一堆bug...



```
C:\.GamingRoot
C:\appverifUI.dll
C:\COPYING
C:\COPYING.LIB
C:\hiberfil.sys
C:\pagefile.sys
C:\swapfile.sys
C:\vfcompat.dll
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\desktop.ini
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
\Cache\AITextTemplate\Resource\9bfd78c02c319fcb4fbcc5f5eb95339d.png
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
\Cache\AITextTemplate\Resource\be93c5af69f8bba90f93367f4f9c8e39.zip
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
\Cache\AITextTemplate\Resource\c31b7a465b6772005d0d76a49a22f406.zip
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
\Cache\AITextTemplate\Resource\c8c623f28800ea3f89428da0b0553e87.zip
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
\Cache\AITextTemplate\Resource\dd5e6af98f274d121e2cca0c1231bad5.png
C:\$RECYCLE.BIN\S-1-5-21-3622820009-2433014797-509804604-1001\$RM5A7I8\User Data
```

5.截取此时服务端(server)主机的桌面图像

点击第二个按钮，如果截图发送成功，会有一个弹窗反馈

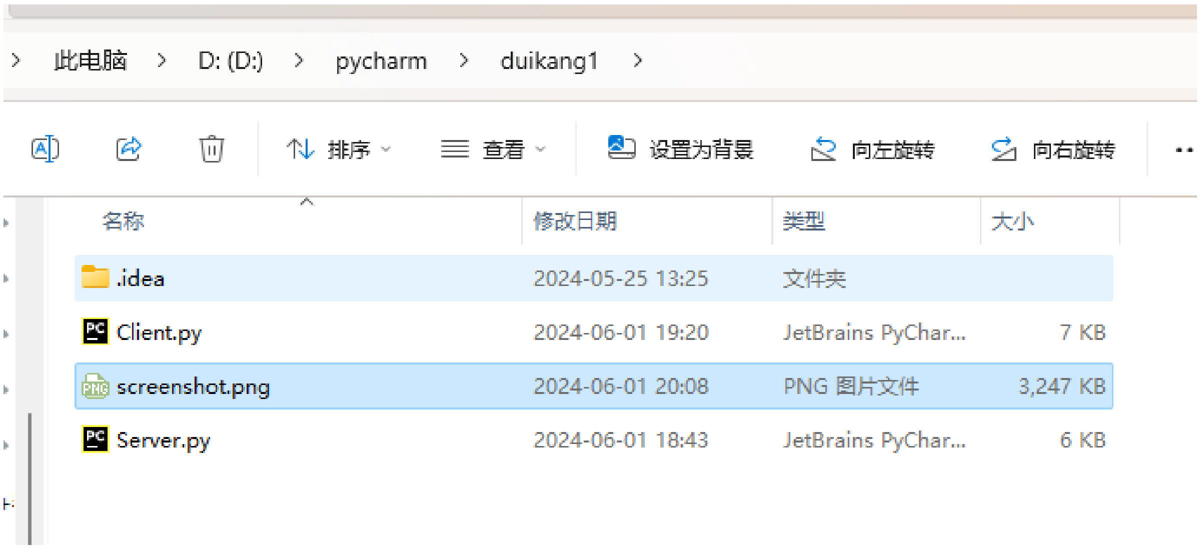


Screenshot has been sent successfully.

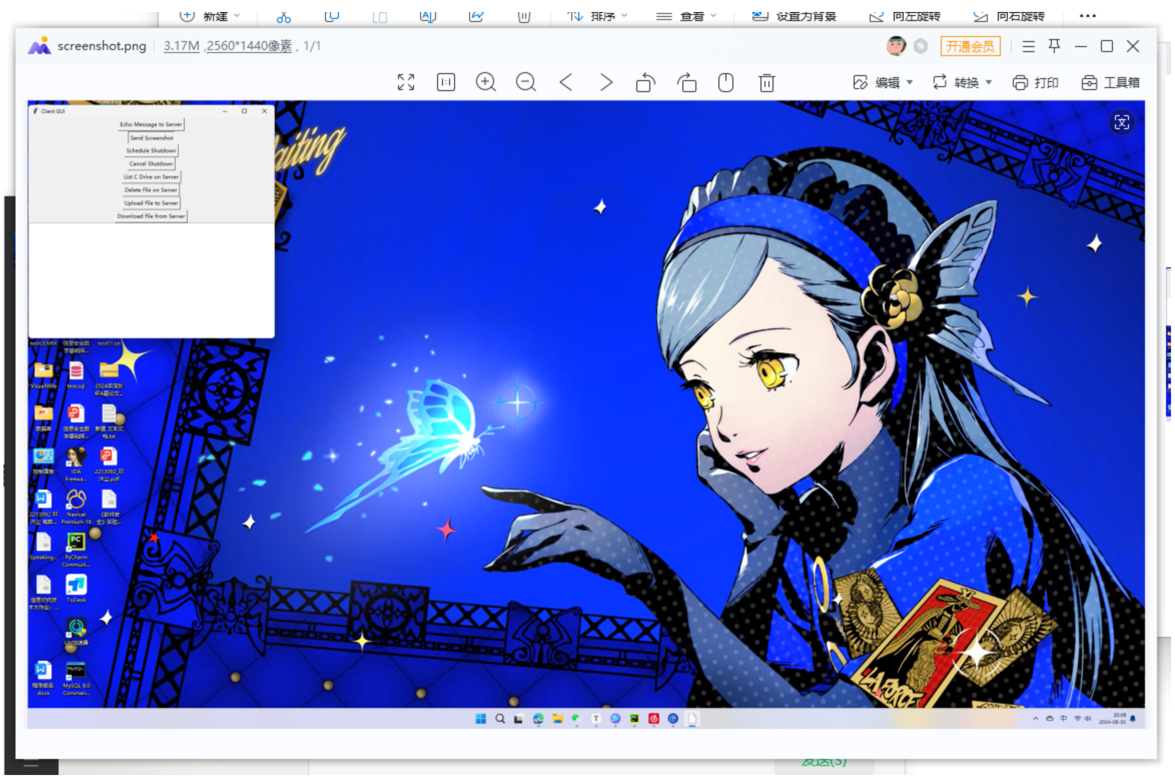
在服务端的命令行我们可以看到图片保存的路径(默认和Server.py同路径)

```
Screenshot saved.
Current working directory: D:\pycharm\duikang1
```

在`D:\pycharm\duikang1`就可以找到刚才发送的图片了

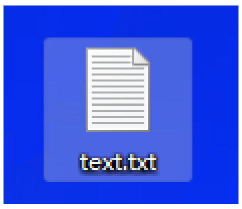


名称	修改日期	类型	大小
.idea	2024-05-25 13:25	文件夹	
Client.py	2024-06-01 19:20	JetBrains PyChar...	7 KB
screenshot.png	2024-06-01 20:08	PNG 图片文件	3,247 KB
Server.py	2024-06-01 18:43	JetBrains PyChar...	6 KB



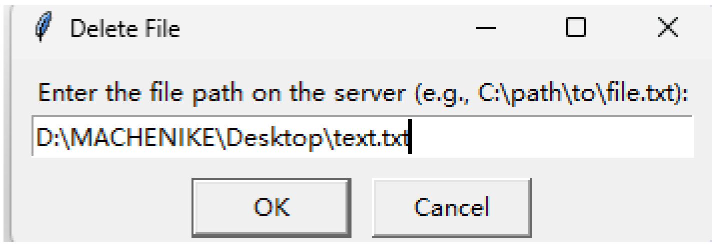
6. 删除：在服务端(server)主机 C 盘列表中选取并删除指定文件

为了方便测试，我们现在桌面上新建了一个名为`test.txt`的文本文件



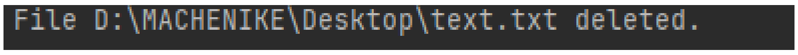
然后我们可以输入文件的地址删除该文件（记得去掉引号），服务端的命令行会给出相应反馈

注意：删掉的文件并不会出现在回收站中，而是直接被删除

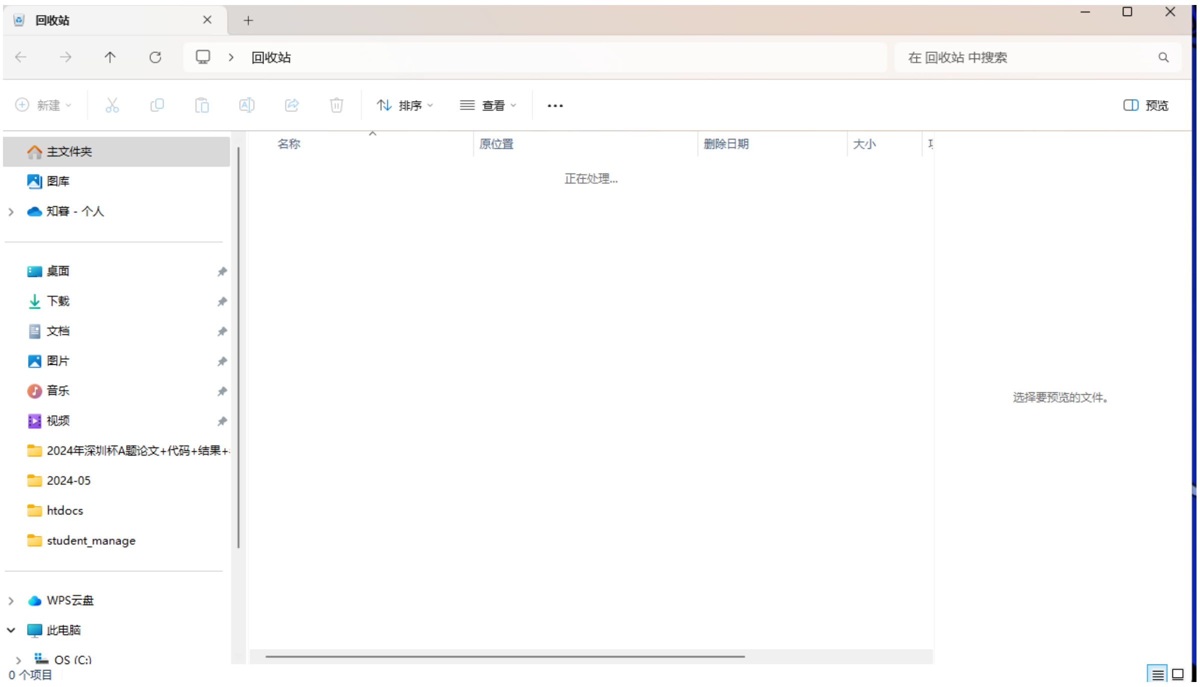


Enter the file path on the server (e.g., C:\path\to\file.txt):

D:\MACHENIKE\Desktop\text.txt



```
File D:\MACHENIKE\Desktop\text.txt deleted.
```



7.上传：在客户端(Client)主机中选取指定文件，将其内容传送并保存至文件“myFile.txt”

按理说代码是没有问题的，但是在测试过程中GUI多次卡死，非常不稳定，最终我选择用多线程来避免GUI阻塞，，并确保文件上传逻辑在一个单独的线程中执行
然后，我添加了代码来解析文件名和文件大小，这样我们就可以将文件保存为原来的名字而非 `mytext.test`(避免上传多个文件时)

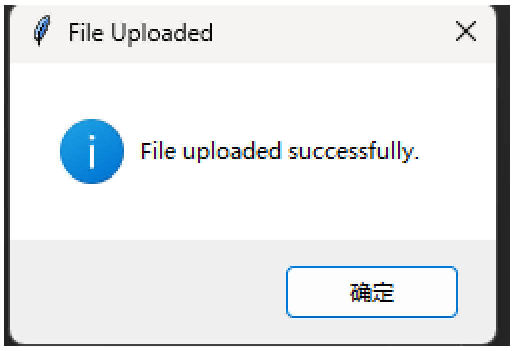
服务端代码 (`Server2.py`)

```python
import socket
import threading
import os

def handle_client(client_socket, client_address):
    print(f"Client connected from {client_address}")
    try:
        # 接收文件名和文件大小
        data = client_socket.recv(1024).decode()
        if not data.startswith("UPLOAD:"):
            print("Invalid request format")
            return
        file_info = data.split(':')
        if len(file_info) != 2:
            print("Invalid request format")
            return
        file_name, file_size_str = file_info[1].strip(), ''
        if ',' in file_name:
            file_name, file_size_str = file_name.split(',')[0],
file_name.split(',')[1]
        file_size = int(file_size_str)

        # 创建文件以写入数据
        with open(file_name, 'wb') as f:
            remaining_bytes = file_size
            while remaining_bytes > 0:
                data = client_socket.recv(4096)
                if not data:
                    print("Connection closed by client")
                    break
                f.write(data)
                remaining_bytes -= len(data)
        print(f"File {file_name} received successfully.")
    except Exception as e:
        print(f"Error handling client: {e}")
    finally:
        client_socket.close()

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('localhost', 12345))
    server.listen(5)
    print("Server started.")
    while True:
        client_socket, client_address = server.accept()
        client_thread = threading.Thread(target=handle_client, args=
(client_socket, client_address))
        client_thread.start()

if __name__ == "__main__":
    start_server()
```

客户端代码(`Client2.py`)

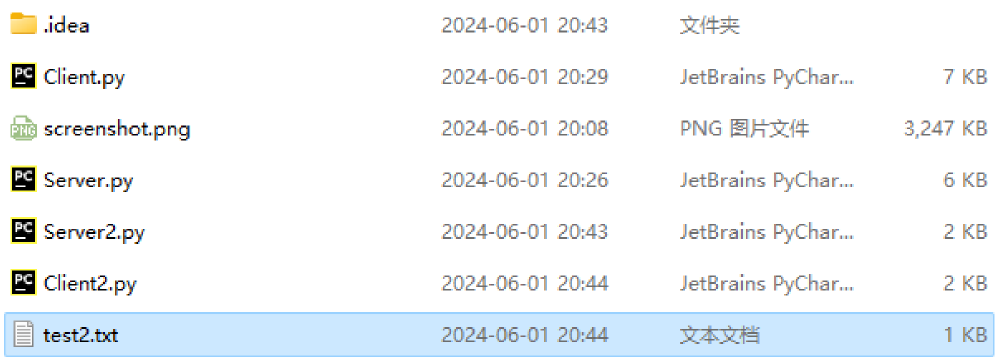
```
import socket
import tkinter as tk
from tkinter import filedialog, messagebox
import threading
import os

class ClientGUI:
    def __init__(self, master):
        self.master = master
        master.title("Client GUI")
        self.client_socket = None
        self.upload_button = tk.Button(master, text="Upload File to Server",
command=self.upload_file_to_server)
        self.upload_button.pack()
        self.connect_to_server()

    def connect_to_server(self):
        try:
            self.client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
            self.client_socket.connect(('localhost', 12345))
            print("Connected to server.")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def upload_file_to_server(self):
        if not self.client_socket:
            messagebox.showerror("Error", "Not connected to server.")
            return

        file_path = filedialog.askopenfilename()
        if not file_path:
            return

        file_name = os.path.basename(file_path)
        file_size = os.path.getsize(file_path)

        self.client_socket.sendall(f"UPLOAD:{file_name},{file_size}".encode())


        threading.Thread(target=self._upload_file, args=(file_path,)).start()

    def _upload_file(self, file_path):
        try:
            with open(file_path, 'rb') as f:
                while True:
                    data = f.read(4096)
                    if not data:
                        break
                    self.client_socket.sendall(data)
            messagebox.showinfo("File Uploaded", "File uploaded successfully.")
        except Exception as e:
            messagebox.showerror("Upload Error", str(e))

root = tk.Tk()
client_gui = ClientGUI(root)
root.mainloop()
```

改进后的代码可以充分实现该功能，在Server2.py的目录下可以找到上传的文件



File Uploaded

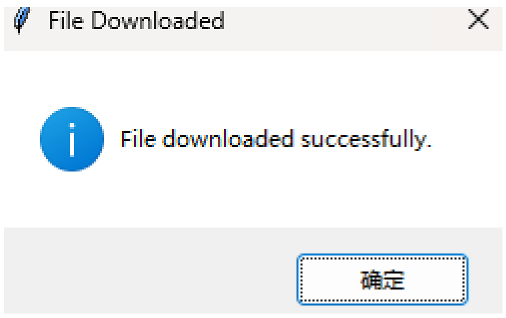
File uploaded successfully.



.idea	2024-06-01 20:43	文件夹	
Client.py	2024-06-01 20:29	JetBrains PyChar...	7 KB
screenshot.png	2024-06-01 20:08	PNG 图片文件	3,247 KB
Server.py	2024-06-01 20:26	JetBrains PyChar...	6 KB
Server2.py	2024-06-01 20:43	JetBrains PyChar...	2 KB
Client2.py	2024-06-01 20:44	JetBrains PyChar...	2 KB
test2.txt	2024-06-01 20:44	文本文档	1 KB

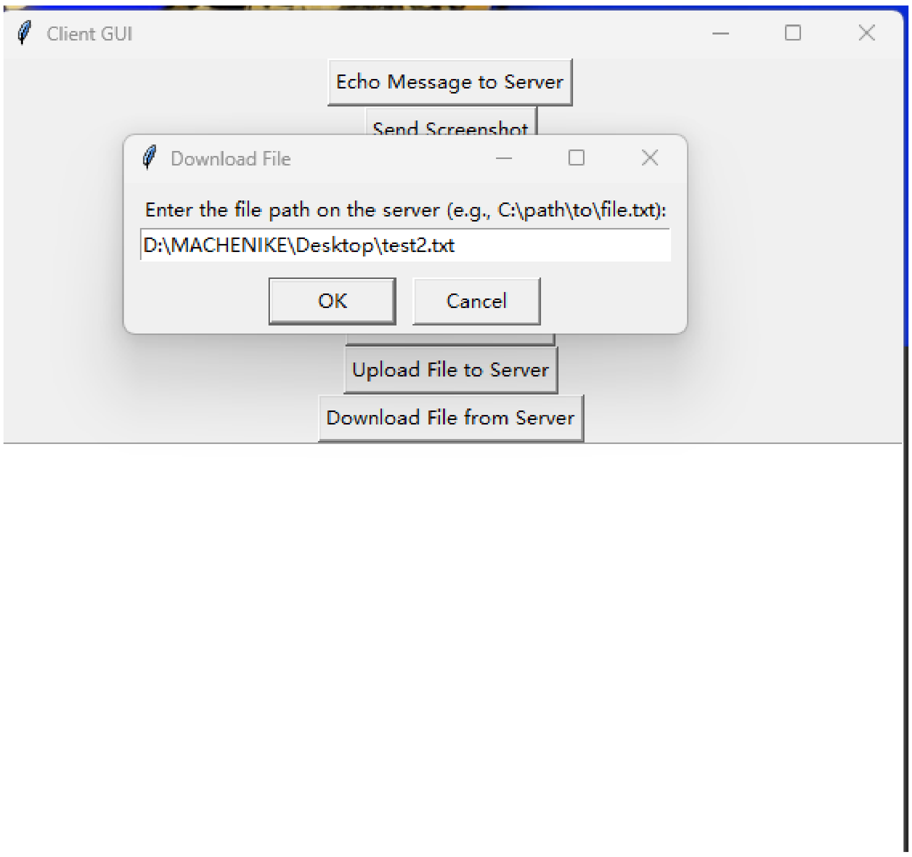
8.下载：在服务端(server)主机 C 盘列表中选取指定文件，并保存至所选路径下

点击最后一个按钮，并输入想下载的文件地址，就能下载文件并保存到指定地址



File Downloaded

File downloaded successfully.


Client GUI

Echo Message to Server

Send Screenshot

Download File

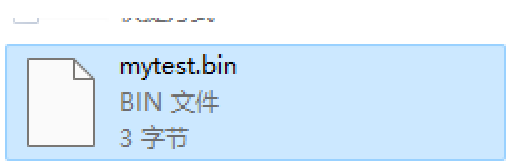
Enter the file path on the server (e.g., C:\path\to\file.txt):

D:\MACHENIKE\Desktop\test2.txt

OK Cancel

Upload File to Server

Download File from Server



心得体会

我觉得有亿点点难