

UMS：便携式无界鼠标与文件传输系统

初夏 施政言 杨俊龙

2023 年 12 月 25 日

目录

1	项目简介	2
2	相关工作	2
3	设计架构	3
3.1	项目整体框架	3
3.2	TCP 连接设计	3
3.3	文件传输设计	5
3.4	无界鼠标设计	5
4	项目运行流程	6
4.1	整体流程	6
4.2	无界鼠标流程	6
4.3	文件传输流程	7
5	结果展示	9
6	总结与展望	9
7	致谢	10
8	参考文献	10

1 项目简介

21 世纪，人们拥有的电子设备越来越多。以大学生为例，有不少人选择购买两台电脑：一台笔记本提供便携，一台台式机提供性能。当用户同时使用这两台电脑时，就会遇到麻烦：由于这两台电脑分别安装了两个单机操作系统，用户需要使用两个鼠标才能操作这两个主机，且两台电脑之间的文件传输也需要依赖外部的应用程序，非常麻烦。

为了让一个鼠标操作两台电脑，人们提出了共享鼠标的解决方案。现在市面上已经存在了一些无界鼠标的实现，比如 Symless 公司的 Synergy 和微软公司的 Mouse Without Borders，但是他们无法解决文件传输的困难。而现有的文件传输方案，比如微信的文件传输功能，则往往过于臃肿。它们常常需要用户打开一个单独的窗口，选择需要传输的文件，在文件传输完毕后打开文件的保存路径，最后手动打开该文件。这一繁琐的过程大大降低了工作效率。

为了同时解决这两个问题，并提升工作效率，我们提出了 UMS (Unbounded Mouse Sync)，一种便捷的无界鼠标与文件传输系统。该系统提供共享鼠标与一个文件管理系统，允许用户使用一个鼠标，将该系统托管的文件窗口直接从一台电脑拖拽到另一台电脑。

我们的 UMS 系统具有 3 个 U 的特点，即 Unbounded, Universal 和 Unified：UMS 系统可以无界地在多台电脑上实现统一的鼠标操控与文件传输。为了凸显出这些特点以及它的用途，我们将我们的系统命名为 Unbounded Mouse Sync，简称 UMS。

2 相关工作

微软公司开发的 Mouse Without Borders (无界鼠标) 应用 (现已加入 Powertoys¹ 组件)。无界鼠标多台计算机之间的支持鼠标和键盘共享，并且配置简单，同时具有较高的安全性和保密性。

罗技公司推出了 Logitech FFlow 功能，支持在多台计算机之间自动切换鼠标和复制粘贴文本、图片和文件。该功能以插件的方式提供，要求用户有支持该功能的 Logitech 键盘或鼠标。

Symless 公司推出的 Synergy 支持跨平台进行鼠标键盘和剪贴板共享，它允许用户将多台计算机连接在一起，并将它们视为一个扩展的工作台。Synergy 支持 Windows、Mac 和 Linux 等操作系统。

ShareMouse 是由德国公司 Bartels Media GmbH 推出的一款实现多台计算机之间的鼠标、键盘和剪贴板共享的商业软件，适用于 Windows 和 Mac 操作系统。

¹<https://github.com/microsoft/PowerToys>

3 设计架构

3.1 项目整体框架

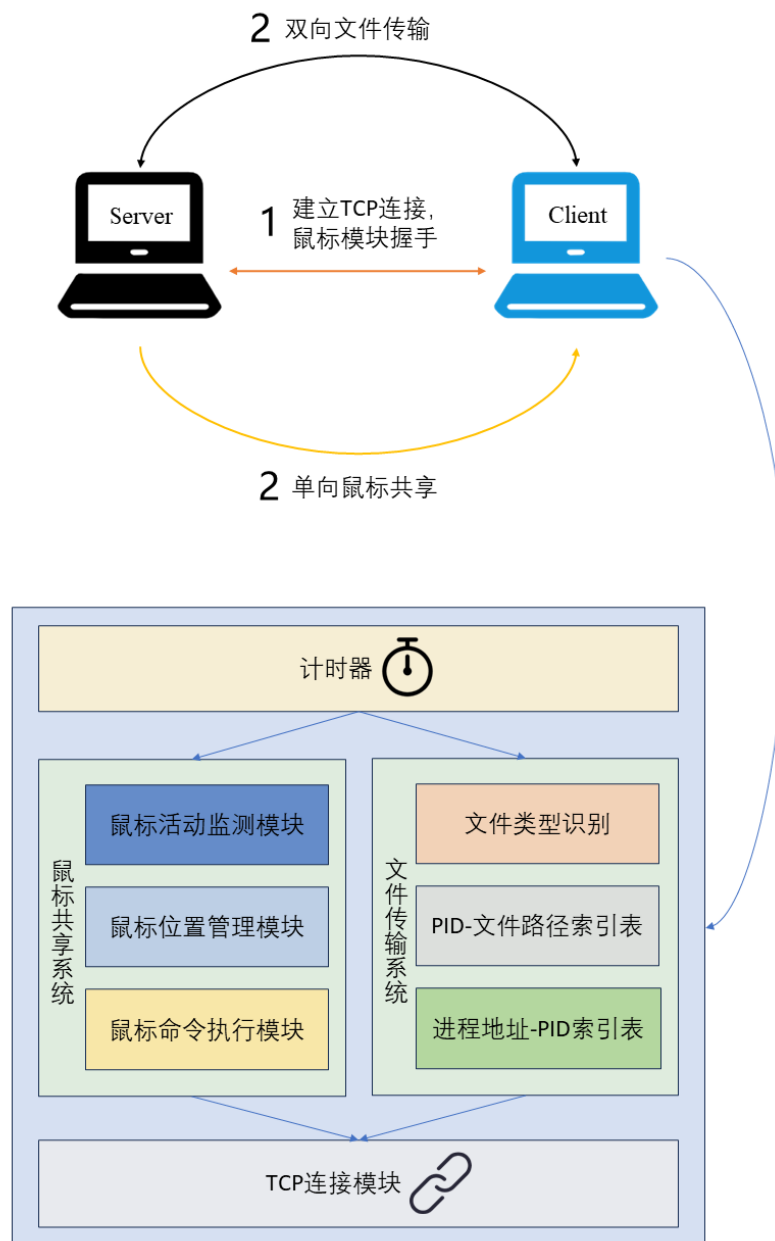


图 1: 项目整体框架示意图

3.2 TCP 连接设计

TCP 连接是该项目的基本底层模块，用于实现端到端的信息传输。在本项目中，TCP 连接主要用于传输文件信息以及鼠标位置、点击等信息，最终实现鼠标状态的共享和文件的快速传输。

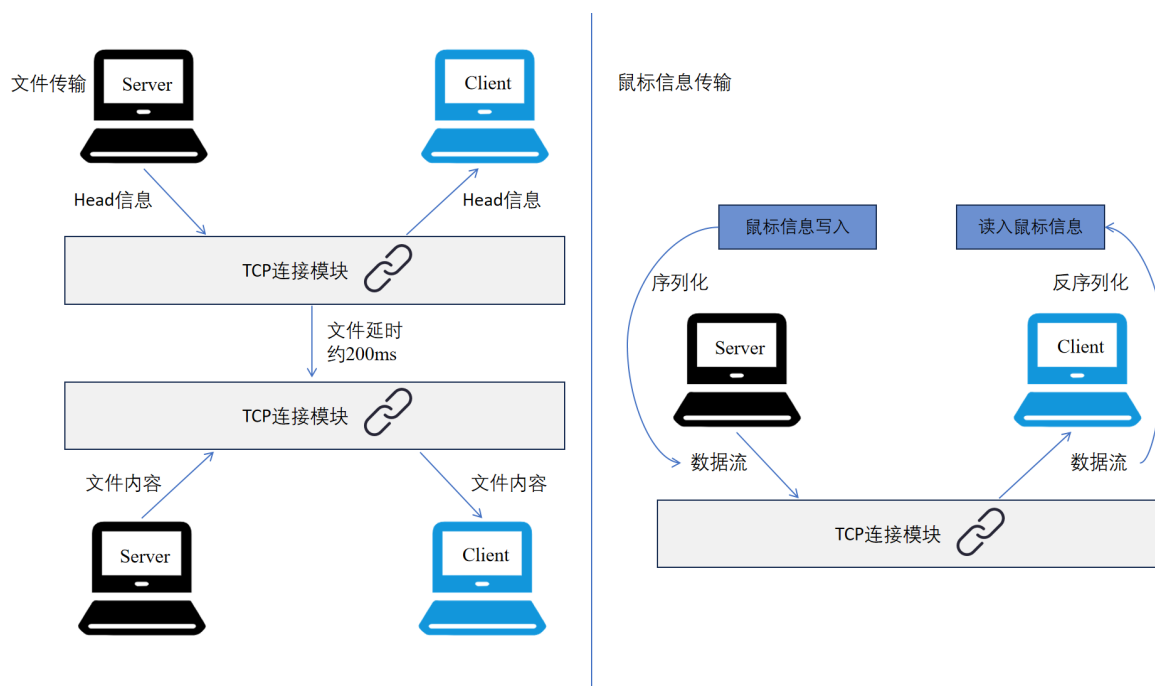


图 2: TCP 文件传输示意图

传输稳定性设计 本项目的信息分为两个部分，若在同一个端口传输，可能会出现 TCP 协议粘包以及信息混淆等错误。后者可以通过对信息添加校验码来规避，但是若想保证传输的即时性，粘包问题较难彻底解决。为了避免两个模块的信息传输冲突，我们将文件信息和鼠标信息分开传输，分别设计两个线程，保存两个任务对应的端口，同时接收信息。两个端口的传输延迟可以不同，同时保证了整体传输过程的稳定性和即时性。

由于本项目可以实现双端传输，以下“Server 端”、“Client 端”仅为单方向传输的指代。

服务器 (Server) 端与客户 (Client) 端 Server 端的功能主要是监听端口号，并向端口写入信息；而 Client 端在连接 Server 端后，可以接受 Server 端写入的信息。我们利用 Qt 的内置类 QTcpServer/QTcpSocket 来建立单向的连接。两者主要利用 Qt 的事件循环机制来异步处理连接请求和数据传输。事件循环是 Qt 框架的核心，它允许对象在不阻塞主线程的情况下处理事件。当有新的连接请求时，QTcpServer 会触发 newConnection 信号。Server 写入数据后，Client 可以通过 QTcpSocket 的 readyRead 函数来读入数据。

数据包装 两种数据必须经过合理的包装才能通过 TCP 协议传输。对于文件，我们先需要传输一个头信息 (Head)，包括文件的名称以及大小，并将它们隔开，以供 Client 端识别。文件的内容被写入一个字符数组中，在头信息发送的短暂延时后，将此字符数组发送。对于鼠标信息，为了降低信息的延时，我们首先创建了一个结构体来存储鼠标

所有信息，在传输时采用数据流（DataStream）的方式将该结构体序列化，写入 Client 端口。Client 端接收到数据流，再将其反序列化为结构体数据。

延迟传输 为了解决相同数据间的 TCP 粘包问题，我们引入了延迟传输的方案。具体来说，对于文件传输，我们在头信息发出后，设置定时器来确保头信息被 Client 端完整接收，再发送文件内容。由于文件的传输对于即时性要求并不高，并且文件内容相比鼠标信息数据量大，我们将定时器时间设置为 200ms。对于鼠标信息传输，我们设置了 Trigger 函数，该函数首先将成员数据修改为当前的鼠标信息，并设置定时器，再发送鼠标信息。由于鼠标信息的传输对于即时性要求较高，并且鼠标信息数据量相对小，我们将定时器时间设置为 30ms。

3.3 文件传输设计

对于文件传输模块，我们的设想是：打开一个文件后，就能够将打开的文件窗口在两台电脑之间来回拖拽，从而跳过传输文件再手动打开文件的繁琐过程。为了实现这一功能，我们需要设计一个文件管理系统来保存和维护文件以及窗口的相关信息。该系统主要由 PID-文件路径索引表和进程地址-PID 索引表组成。

PID-文件路径索引表 PID-文件路径索引表用一个字典实现，PID 为 key，文件路径为 value。该表记录了打开文件的窗口的 PID 与打开文件的路径之间的对应关系。当鼠标拖拽一个窗口到屏幕边缘时，系统会根据该表查询需要传输的文件路径。

进程地址-PID 索引表 进程地址-PID 索引表用一个字典实现，进程的内存地址为 key，PID 为 value。该表记录了系统托管的进程的内存地址与其进程号之间的对应关系。该表主要用于辅助 PID-文件路径索引表的维护与更新。当文件不再需要系统托管，比如用户关闭文件窗口或文件被传输到另一台电脑上时，系统根据该表查询文件对应的 PID，并删除 PID-文件路径索引表中响应的条目。

其他文件处理模块 其他文件处理模块主要负责文件的打开，文件路径的获取，文件类型的识别等。用于将文件交给 UMS 系统托管以及辅助 PID-文件路径索引表的更新和维护。

3.4 无界鼠标设计

无界鼠标模块是该项目的核心组件，主要由三个部分构成，分别为：全局鼠标监测模块、鼠标管理模块、鼠标命令执行模块。

全局鼠标监测模块 全局鼠标监测模块需要实时监测鼠标的位置信息以及按键信息，并发送相应的信息到鼠标管理模块进行处理。监测鼠标位置信息方面使用了 Qt 的内置函数来获取，而监测按键信息则是采用了 Windows API 中的 HOOK（鼠标钩子）来实现。具体而言，定义了一个时钟按一定采样频率获取鼠标的位置信息，同时利用全局鼠标钩子获取鼠标的按键输入，在获取到上述两个输入后，利用 Qt 中信号与槽的方法将对应的信息打包传输给鼠标管理模块，交由鼠标管理模块进行处理。

鼠标管理模块 鼠标管理模块需要对全局鼠标监测模块发来的信息进行处理，同时需要维护一些与无界鼠标有关的属性信息如位置信息、主从信息等等，并根据当前的信息判断是否需要向鼠标命令执行模块发送命令请求。鼠标信息经过打包后分成四类信息：鼠标握手信息、鼠标切换信息（当鼠标在主从电脑的屏幕间切换时，需要该信息作为标识）、鼠标位置信息、鼠标按键信息。Server 端的鼠标管理模块将对应的信息发送到 Client 端的鼠标命令执行模块，从而实现远端鼠标的控制。

鼠标命令执行模块 Client 端的鼠标命令执行模块在接收到来自 Server 端的鼠标信息后，将根据自身情况解析鼠标命令并执行，执行方式采用了 Qt 的内置函数和 Windows API 中的 SendInput 方法来实现对鼠标位置和鼠标按键操作的模拟。

总的来看，我们的无界鼠标的核心设计思路是基于 TCP 连接，将 Server 端鼠标的信息在经过处理后传输到 Client 端进行处理，中间利用鼠标管理模块进行双端的同步和鼠标管理，从而在 Server 端和 Client 端之间实现一个简单的无界鼠标。

4 项目运行流程

4.1 整体流程

用户在运行该项目时，主要有三个流程：首先用户需要配置 Server 端和 Client 端以及对应的 ip（同一局域网内）和端口（可采用默认值）；然后用户需要在 ui 界面进行 Server 端和 Client 端的互连以及鼠标握手；在完成握手协议后，无界鼠标与文件传输功能即可正常运行，在网络通畅的情况下，即可实现两台电脑 Server 端和 Client 端的鼠标共享和文件传输了。

4.2 无界鼠标流程

在建立了 TCP 连接后，无界鼠标首先需要经过一个握手协议才会开始工作。这个握手协议要求 Server 端和 Client 端互相发送自身的屏幕数据，双方接收到该数据并确认接收后，Server 端和 Client 端均会进入初始化模式。

Server 端会将全局鼠标监测器打开并用 Client 端的屏幕数据初始化一个透明遮罩（用于屏蔽主电脑端鼠标输入），Client 端会初始化鼠标命令执行模块，准备接收来自

Server 端的鼠标命令请求。

完成初始化后，Server 端会持续监测鼠标位置，当监测到鼠标离开本地屏幕时，会发出一个鼠标切换信息，该信息会发送给鼠标管理模块，鼠标管理模块进一步将该信息同步到 Client 端，双方都会修改本地的鼠标属性信息，然后，Server 端会打开之前初始化的遮罩并将鼠标约束在遮罩范围内，这是由于此时在逻辑上无界鼠标位于 Client 端，因此其不应对 Server 端产生影响，使用该遮罩可以起到屏蔽鼠标输入的作用，同时又能继续获取本地鼠标输入并传输给 Client 端，Client 端接收这些输入并通过鼠标命令执行模块进行处理，从而将 Server 端的鼠标输入模拟到 Client 端，对于用户而言则模拟出了在 Server 端使用鼠标控制 Client 端电脑的效果。

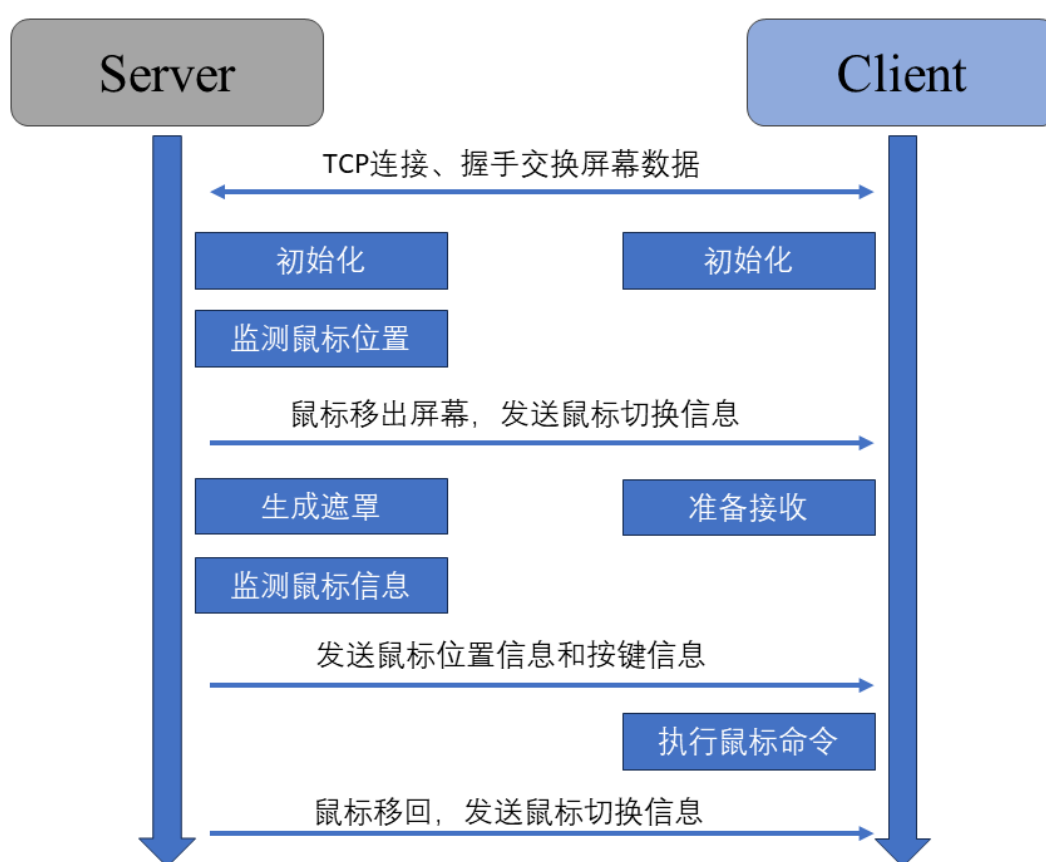


图 3: 无界鼠标流程示意图

4.3 文件传输流程

首先，用户需要选择想让 UMS 系统托管的文件。点击主界面上的“打开文件”按钮，系统会弹出一个选择文件对话框。在用户选择文件后，系统会创建一个进程将该文件打开。此时，系统会将这个进程的进程号与文件的路径记录在 PID-文件路径索引表中。此外，系统还会将进程地址与进程号记录在进程地址-PID 索引表中。

随后，当用户拖拽文件打开的窗口到达屏幕边缘时，系统会调用 WindowsAPI，根据此时鼠标所在的位置查询鼠标所在位置的窗口句柄 (HWND)，并根据窗口句柄查询该窗口的进程号 (PID)。然后，系统会向 PID-文件路径索引表发起查询请求，查找该进程对应的文件路径。如果查询失败，即该窗口没有被 UMS 系统托管，则不进行任何后续操作。如果查询成功，则发送信号给 TCP 传输模块，开启文件传输。

当接收端接收到文件时，UMS 系统会将文件保存到默认位置，并自动打开该文件。打开文件时，系统会自动记录 PID 与文件路径的对应关系。因此，在接收端打开的文件窗口还能再次以拖拽的方式传输到发送端，从而实现了双向的文件传输。

当发送端文件传输完毕或用户关闭文件窗口时，被关闭的进程会向系统发送一个信号。UMS 系统捕获到该信号时，会查询进程地址-PID 索引表，并根据查得的 PID 更新 PID-文件路径索引表，从该表中删去被关闭的文件条目，从而完成对该表的维护。

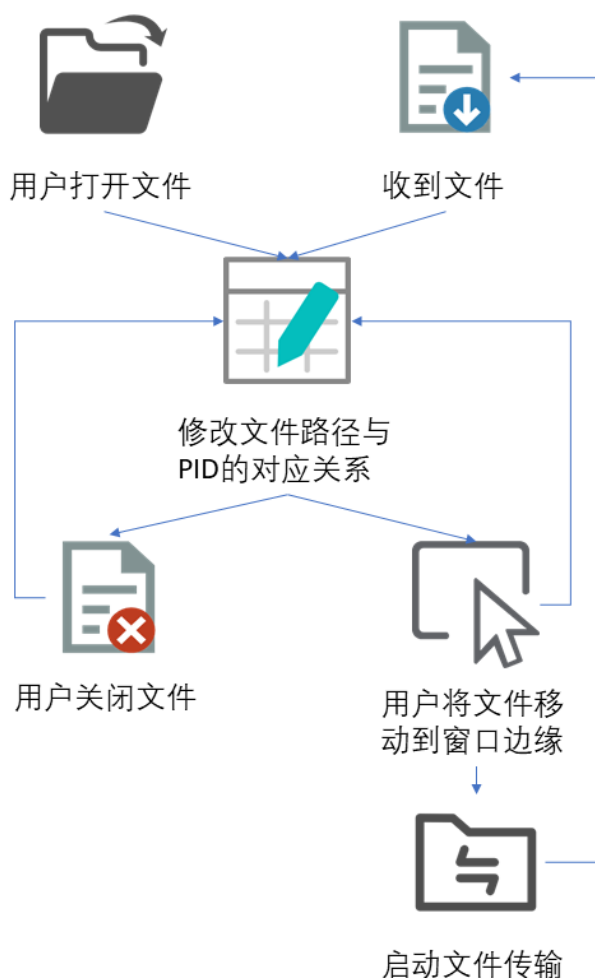
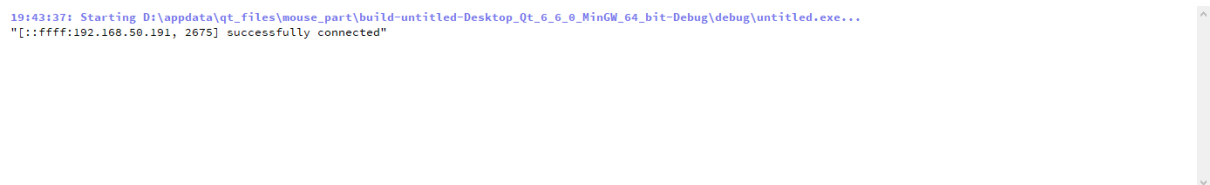


图 4: 文件传输流程示意图

5 结果展示

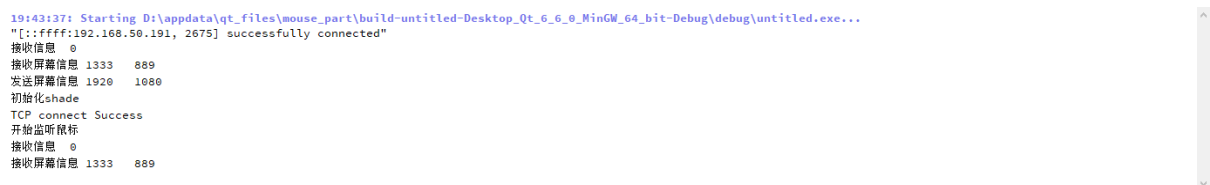
本文分别验证了无界鼠标和文件传输功能是否能够正常运行。演示视频详见附件中的 mp4 文件。

首先,在两台电脑上运行该程序,一台电脑设置为 Server 端,一台电脑设置为 Client 端,建立 TCP 连接和握手。可以看到,Client 端和 Server 端分别完成了 TCP 连接和初始化操作。



```
19:43:37: Starting D:\appdata\qt_files\mouse_part\build-untitled-Desktop_Qt_6_6_0_MinGW_64_bit-Debug\debug\untitled.exe...
[::ffff:192.168.50.191, 2675] successfully connected
```

图 5: TCP 连接成功输出窗口



```
19:43:37: Starting D:\appdata\qt_files\mouse_part\build-untitled-Desktop_Qt_6_6_0_MinGW_64_bit-Debug\debug\untitled.exe...
[::ffff:192.168.50.191, 2675] successfully connected
接收信息 0
接收屏幕信息 1333 889
发送屏幕信息 1920 1080
初始化shade
TCP connect Success
开始监听鼠标
接收信息 0
接收屏幕信息 1333 889
```

图 6: 鼠标初始化成功输出窗口

然后在 Server 端将鼠标移到左边界,可以看到鼠标光标消失,同时 Client 端开始接收 Server 端的鼠标信息,并随着 Server 端的鼠标进行移动。在 Client 端将鼠标从右边界移出,可以看到鼠标又重新出现在 Server 端,并且 Client 端停止接收输入(实现效果不使用图片展示,详见附件 mp4 文件)。

接下来测试文件传输功能,点击按钮打开指定的文档或图片,拖动对应窗口到边界,可以看到窗口在本地端被关闭并在远端打开,传输成功(实现效果不使用图片展示,详见附件 mp4 文件)。

6 总结与展望

本文从生活中的应用场景出发,设计了一款基于 TCP 连接的 UMS 工具,针对鼠标共享和文件传输这两个问题提出了我们的解决方案。利用该工具,可以实现在同一局域网内使用一个鼠标操控两台计算机,同时能够在两台计算机之间实现便捷的文件传输,仅需通过拖拽窗口即可快传文件,省去了在两台计算机之间反复操作带来的冗杂。

UMS 工具虽然一定程度上解决了双机鼠标共享和文件传输的问题，但还存在几个方向可以优化：通过调整 TCP 模块的连接设计，可以支持多台电脑的鼠标共享；通过调整无界鼠标模块的鼠标管理模块，可以实现 Client 端对 Server 端的反向控制；通过利用操作系统的窗口 API 实现部分文件在窗口形式下在两台或多台电脑间的无缝拖拽，提升用户体验。

7 致谢

感谢孔令和教授对小组成员的教导，以及为本项目提供的所有建议与指导。孔教授在大作业开题前与项目组成员交流讨论，提出了本项目的设计方向，促成了本项目顺利开展。

感谢助教学长的辛勤工作。

8 参考文献

1. <https://learn.microsoft.com/zh-cn/windows/powertoys/>
2. <https://github.com/microsoft/PowerToys>
3. https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-mouse__event