**附录**

1. 系统概述

本实验代码Python语言编写，由于Python语言的跨平台性，理论上主流的操作系统均可运行，本次试验中的运行环境为我的个人电脑，具体环境如下：

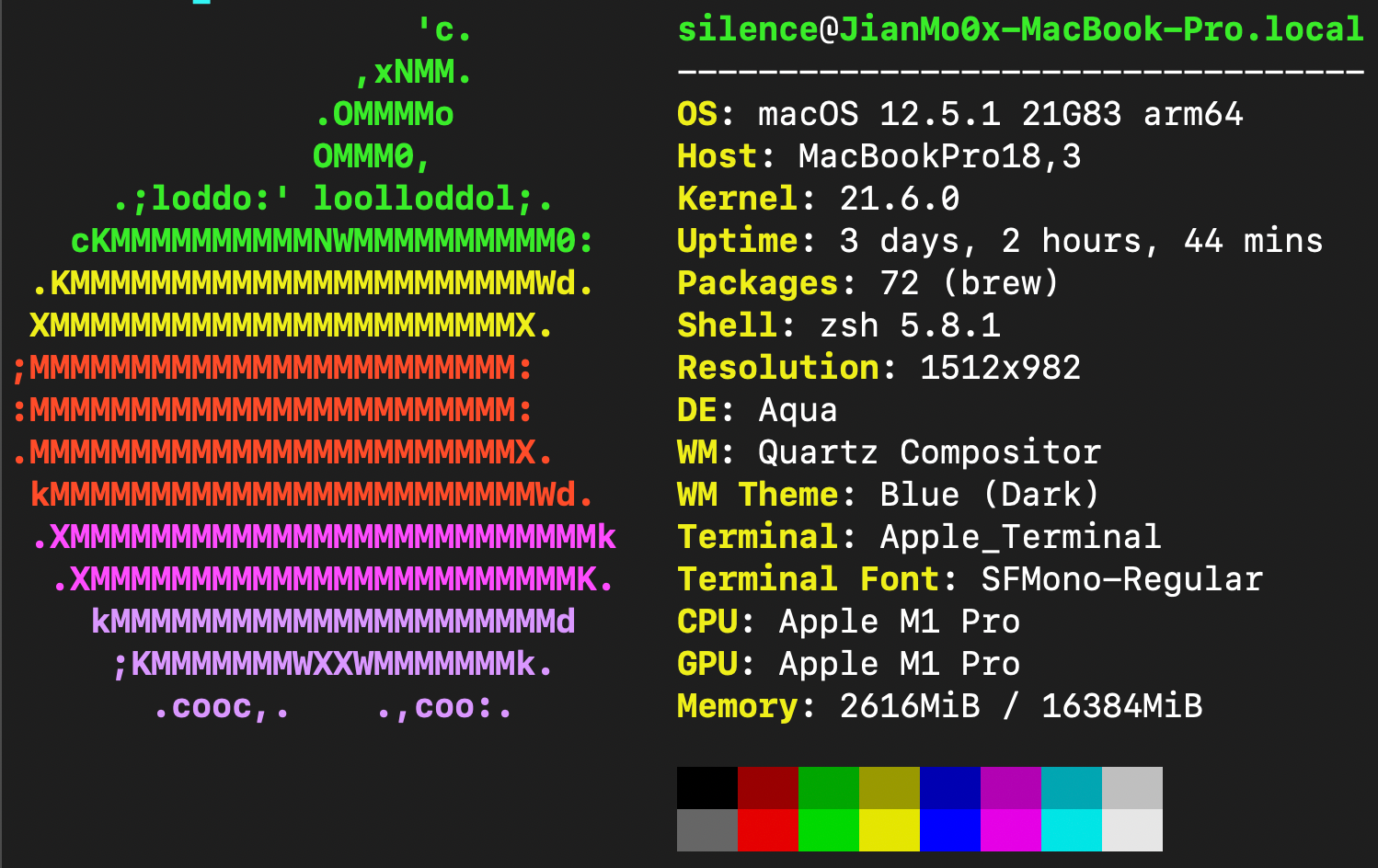


图1. 运行环境

程序文件列表如下图所示，其中，以.ui结尾的是PyQt的UI文件，负责配置图形化界面。gui\_client.py为客户端程序，TCPServer.py\UDPServer.py分别为TCP、UDP的服务器端程序。

文本

中度可信度描述已自动生成

图2. 程序文件列表

1. 主要数据结构

主要数据结构为Socket()类，在实验报告中均有介绍。

针对TCP的Socket为：

socket(socket.AF\_INET, socket.SOCK\_STREAM)

针对UDP的Socket为：

socket(socket.AF\_INET, socket.SOCK\_STREAM)

1. 主要算法描述；

Python 提供了两个级别访问的网络服务：

低级别的网络服务支持基本的 Socket，它提供了标准的 BSD Sockets API，可以访问底层操作系统 Socket 接口的全部方法。

高级别的网络服务模块 SocketServer， 它提供了服务器中心类，可以简化网络服务器的开发。

在Python语言中，socket编程客户端主要分为以下几个步骤，分别将相应的函数接口以及代码如以下所示：

1、UDP发送数据的步骤



图3. UDP发送数据的步骤

2、TCP发送数据的步骤



图4. TCP发送数据的步骤

在Python语言中，socket编程服务器端主要分为以下几个步骤：

1、UDP接收数据的步骤

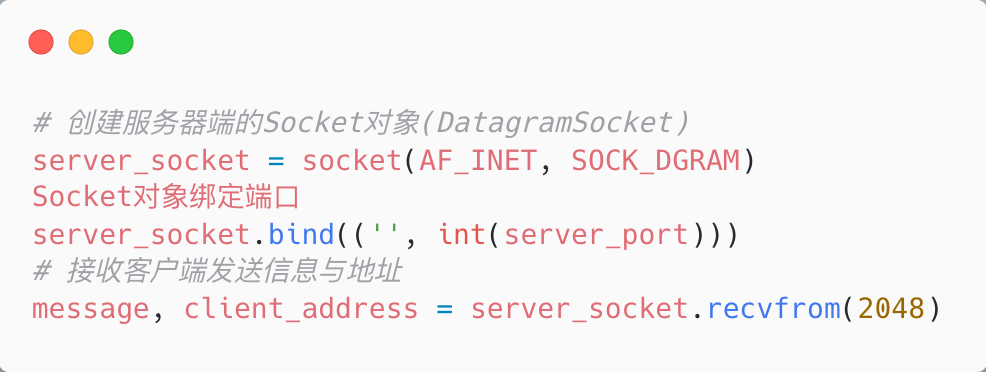


图5. UDP接收数据的步骤

2、TCP接收数据的步骤

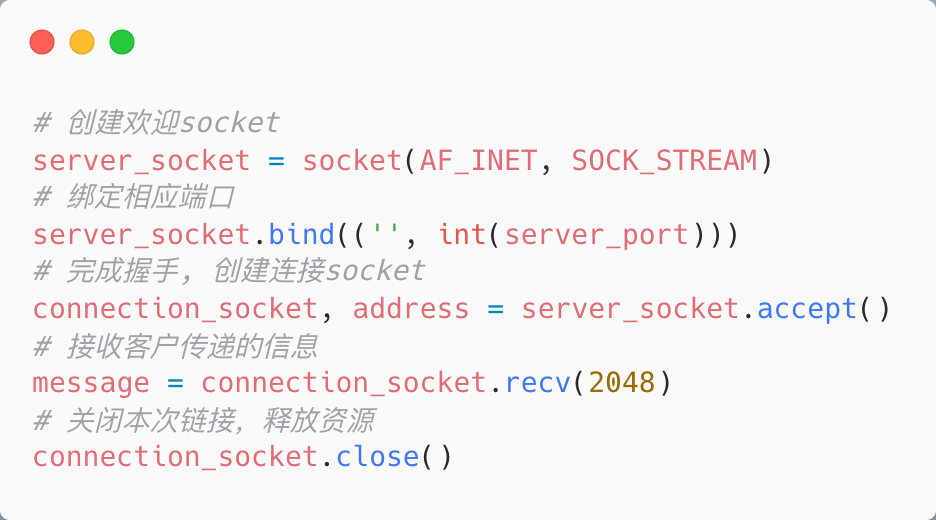


图6. TCP接收数据的步骤

1. 用户使用手册

运行本程序需要使用python分别运行gui\_client.py文件与TCPServer.py或UDPServer.py中的任一一个。随后根据图形化界面的提示操作即可。



图7. 启动客户端程序



图8. 启动服务端程序