缓冲区数据发送与解析实验

1. 实验目的

本实验旨在演示如何通过网络接口发送和接收二进制缓冲区数据（buf），学习结构化数据在网络中的打包、传输和解析过程。具体包括：

- 学习如何使用struct模块对数据进行打包和解包操作
- 掌握通过NetSimAPI进行网络缓冲区数据传输的方法
- 理解数据校验机制在数据传输中的应用
- 了解多无人机数据管理的数据结构设计方法
- 掌握基于事件驱动的数据接收和处理机制

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]。
- 硬件要求：笔记本/台式电脑1台[2]，虚拟机2台。

3.实验地址

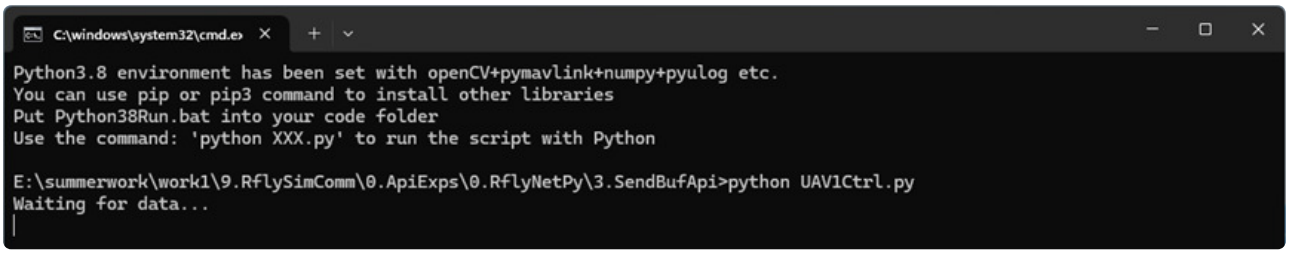
例程目录：
[安装目录]\RflySimAPIs\9.RflySimComm\0.ApiExps\e1.RflyNetAPIExps\3.SendBufApi

文件介绍：
Python38Run.bat：RflySim Python运行脚本
UAV1Ctrl.py：1号无人机控制脚本（接收并解析数据）
UAV2Ctrl.py：2号无人机控制脚本（发送数据）

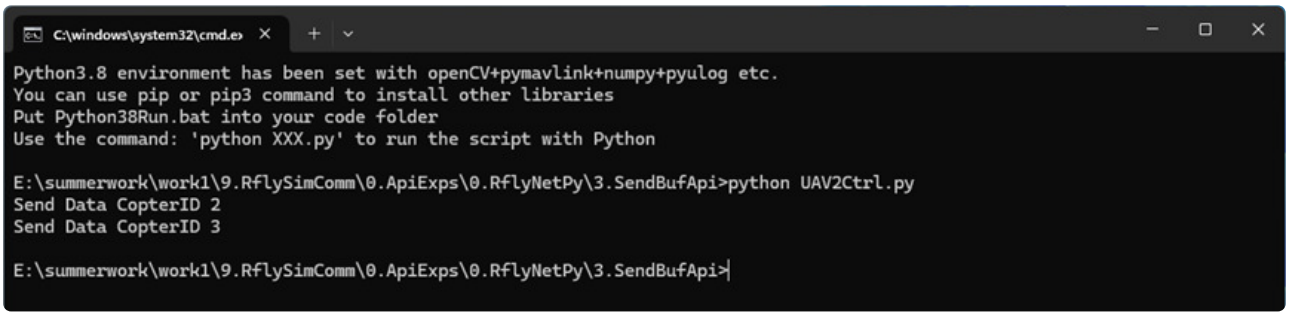
4. 实验内容或步骤

4.1. 步骤1：本机实验

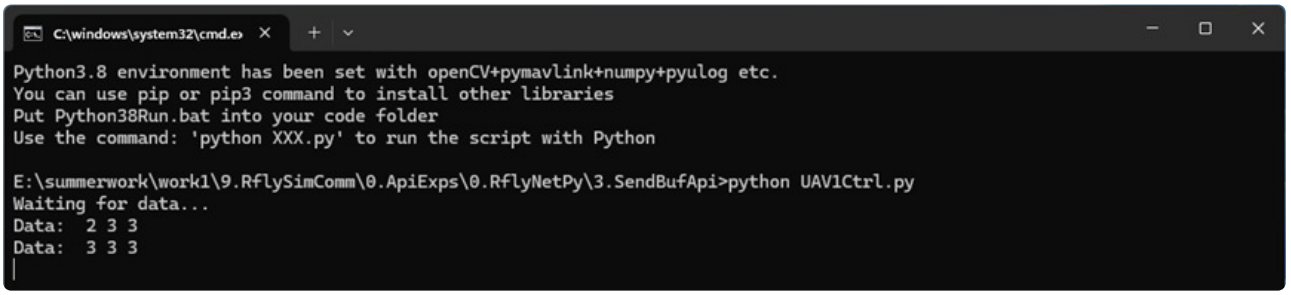
双击Python38Run.bat，输入"python UAV1Ctrl.py"，启动1号飞机接收buf并解析例程。



双击Python38Run.bat，输入"python UAV2Ctrl.py"，启动2号飞机编码并发送buf例程。

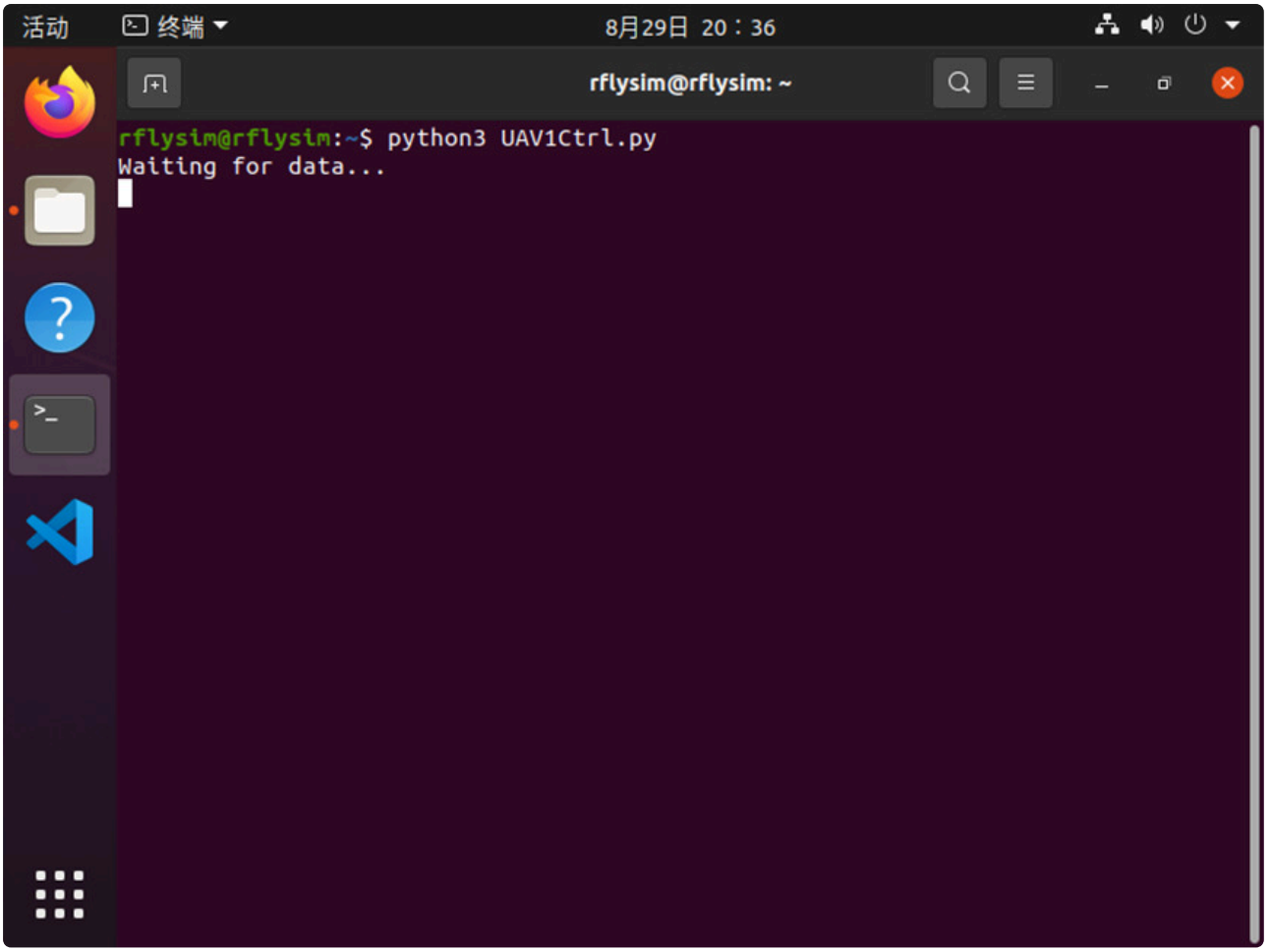


回到运行"UAV1Ctrl.py"的命令框中，可以发现接收到了信息。1号飞机，将结构体转为buf，并发出，在命令提示符窗口打印。2号飞机，收到包后，解析并打印。

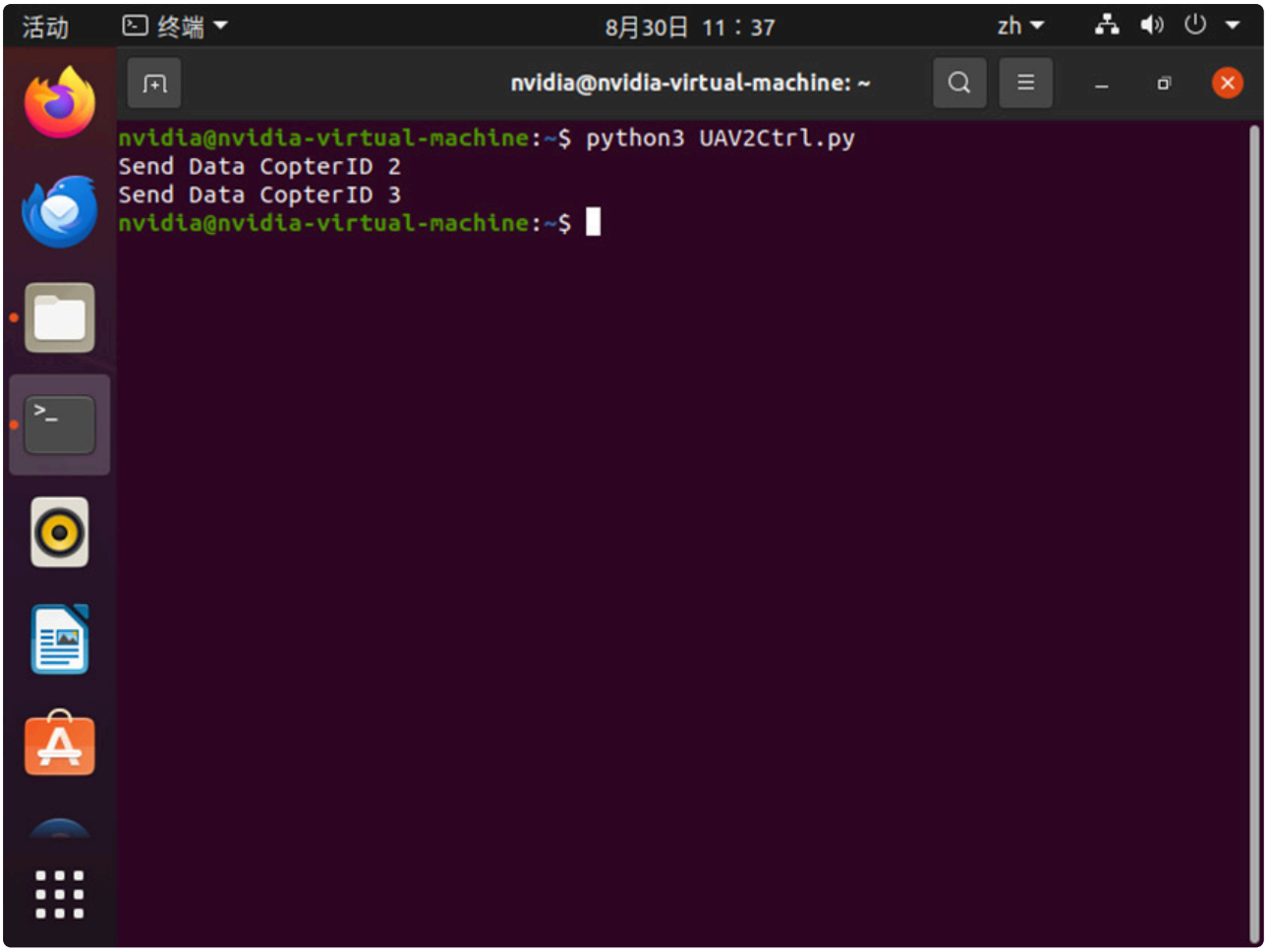


4.2. 步骤2：联机实验

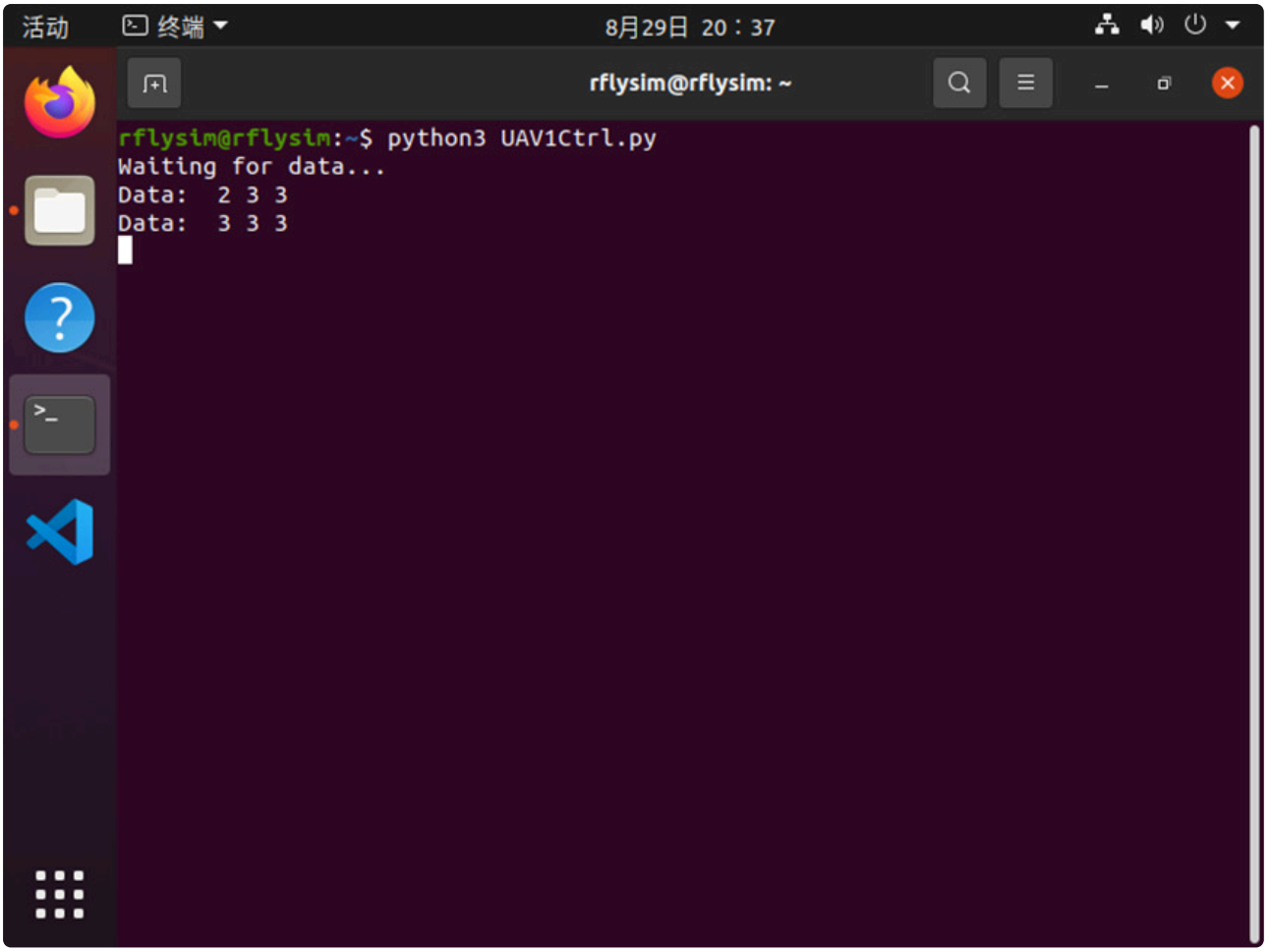
1）准备2台任意系统主机（可以是NX主机、Ubuntu虚拟机、Windows电脑）。确保两台机器是在同一个局域网内。
2）按照[RflySim按安装理解]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e3.PythonConfig实验将[RflySim安装路径]\RflySimAPIs\RflySimSDK\RflySimSDK库复制到每台主机上，并确保RflySimSDK已经导入当前主机的Python环境中。
在一台Ubuntu虚拟机中，运行UAV1Ctrl.py，启动1号飞机接收buf并解析例程。



在另一台Ubuntu虚拟机中，运行UAV2Ctrl.py，启动2号飞机编码并发送buf例程。



回到运行"UAV1Ctrl.py"的虚拟机中，可以发现接收到了信息。1号飞机，将结构体转为buf，并发出，在命令提示符窗口打印。2号飞机，收到包后，解析并打印。能看到和本地测试一样的效果，说明测试正确。



5. 关键知识点

本实验的核心思路是通过网络接口实现多个无人机之间的二进制数据传输。整个系统分为发送端和接收端：

- 发送端（UAV2Ctrl.py）负责构造数据包并通过网络发送
- 接收端（UAV1Ctrl.py）负责接收数据包并解析其中的信息

UAV1Ctrl.py 代码解析

UAV1Ctrl.py 是数据接收和解析的程序，其主要功能包括：

```
import NetSimAPIV4
import struct

net = NetSimAPIV4.NetSimAPI(1)

net.StartNetRec() # 默认开始监听60000端口，即所有飞机。
```

首先导入必要的模块，然后创建NetSimAPI实例，指定无人机ID为1，接着启动网络接收功能。

定义了一个UAVData类，用于存储无人机数据：

```
class UAVData:
    def __init__(self):
        self.hasUpdate=False
        self.CopterID=False
        self.Data1=False
        self.Data2=False

    def update(self,Data):
        self.hasUpdate=True
        self.CopterID=Data[1]
        self.Data1=Data[2]
        self.Data2=Data[2]
```

在主循环中，程序等待接收数据包，并对其进行解析：

```
print('Waiting for data...')
while True:
    net.bufEvent.wait() # 会一直阻塞在这里，直到有新的数据包收到
    bufData = copy.deepcopy(net.bufData) # 立刻将包拷贝出来
    bufHead = copy.deepcopy(net.bufHead) # 立刻将包头拷贝出来
    net.bufEvent.clear() # 清除set位，使得下层能够继续更新数据
```

当接收到16字节的数据包时，进行校验和解析：

```
if len(net.bufData) == 16:
    Data = struct.unpack('iiii',net.bufData) # 获取所有数据向量
    checksum = Data[0]
    if checksum==123456789: # 校验通过，说明是对的包
        CopterID = Data[1]

        if UAVDataDict.__contains__(str(CopterID)): # 如果飞机数据已经在字典中
            UAVDataDict[str(CopterID)].update(Data) # 直接更新数据
        else:
            uData = UAVData() # 创建一个新的空结构体
            uData.update(Data) # 更新数据
            UAVDataDict[str(CopterID)] = copy.deepcopy(uData) # 拷贝并加入字典

        print('Data:
',UAVDataDict[str(CopterID)].CopterID,UAVDataDict[str(CopterID)].Data1,UAVDataDict[
str(CopterID)].Data1)
```

UAV2Ctrl.py 代码解析

UAV2Ctrl.py 是数据发送端的程序，其主要功能是构造数据包并发送：

```
import NetSimAPIV4
import time
import struct

net = NetSimAPIV4.NetSimAPI(2)

net.enNetForward() # 默认发给60000端口，即所有飞机。
```

创建NetSimAPI实例，指定无人机ID为2，启用网络转发功能。

```
time.sleep(5)

buf = struct.pack('iiii',123456789,2,3,4) # 封装一个数据包，备注2号飞机
net.netForwardBuf(buf)
print('Send Data CopterID 2')

time.sleep(5)

buf = struct.pack('iiii',123456789,3,3,4) # 封装一个数据包，备注3号飞机
net.netForwardBuf(buf)
print('Send Data CopterID 3')
```

程序首先休眠5秒，然后使用struct.pack函数将数据打包成二进制格式，数据包含校验码、无人机ID和两个自定义数据字段。随后通过netForwardBuf函数将数据包发送出去。

6.参考资料

1. RflySim官方文档： https://rflysim.com/doc/zh/
2. MavLink协议详解
3. Python Struct模块使用指南

7.常见问题

Q1：运行UAV1Ctrl.py时，程序卡在net.bufEvent.wait()处不响应。

A1：这是正常现象，程序会在wait()处阻塞，直到接收到网络数据包才会继续执行。请确保同时运行UAV2Ctrl.py发送数据包。

Q2：接收端无法解析到发送端的数据。

A2：请检查两台机器是否在同一网段，防火墙是否阻止了网络连接，以及发送端和接收端的端口号是否一致。

Q3：struct.unpack时出现错误，提示格式不匹配。

A3：请确认接收到的数据长度与预期一致，以及打包和解包使用的格式字符串相同。例如，如果发送端使用'iiii'格式，接收端也必须使用相同的格式.

---

1. https://rflysim.com/ ↩
2. 推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf ↩