

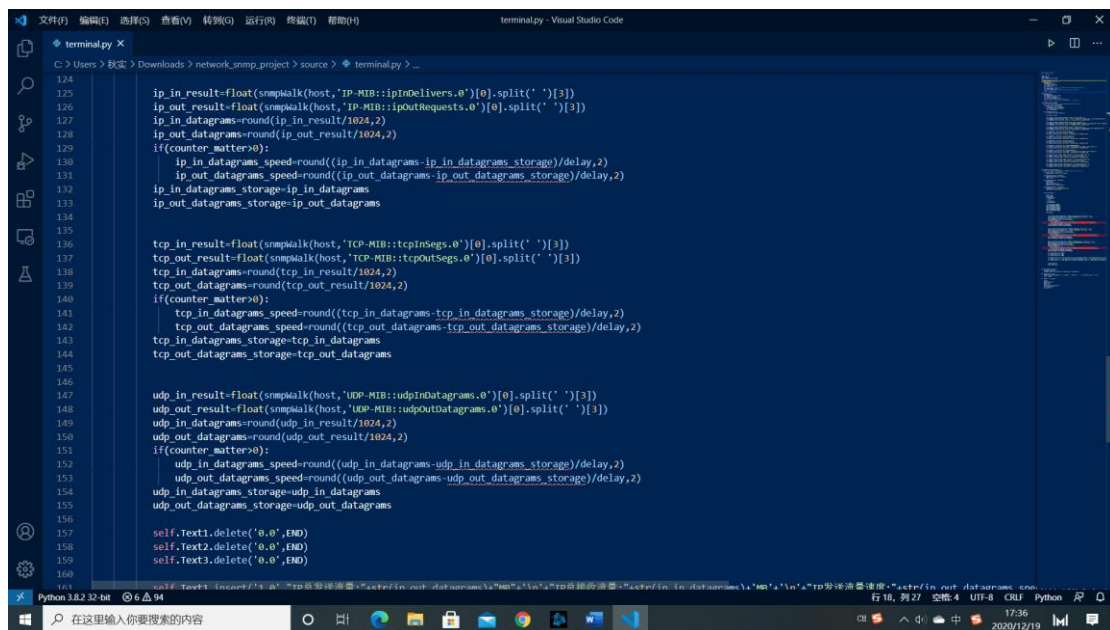
计算机通信网络大作业实验报告

一、概述

本次我选择的大作业题目是 项目三：网络流量分析。最终程序的运行环境是 Win10，要求开启 SNMP 服务并安装 netsnmp；程序设计语言为 python，ide 环境为 vscode；打包可执行文件的库为 pyinstaller。程序文件列表见文件夹下的 source 和 bin 中。

二、主要算法及数据结构

本次作业我选择的题目是网络流量分析，主要通过调用 netsnmp 的方式来完成大作业的各项要求。其中任务要求的显示 tcp、ip、udp 流量通过 snmpwalk 命令调取对应的 mib 库的对应键值，可以获得传输的总流量；然后通过设置刷新延迟时间并求除数的方式，可以得到每次刷新间隔内各个流量的平均传输速度。



```
terminal.py X
C:\Users\> 搜索 > Downloads > network_snmp_project > source > terminal.py > ...

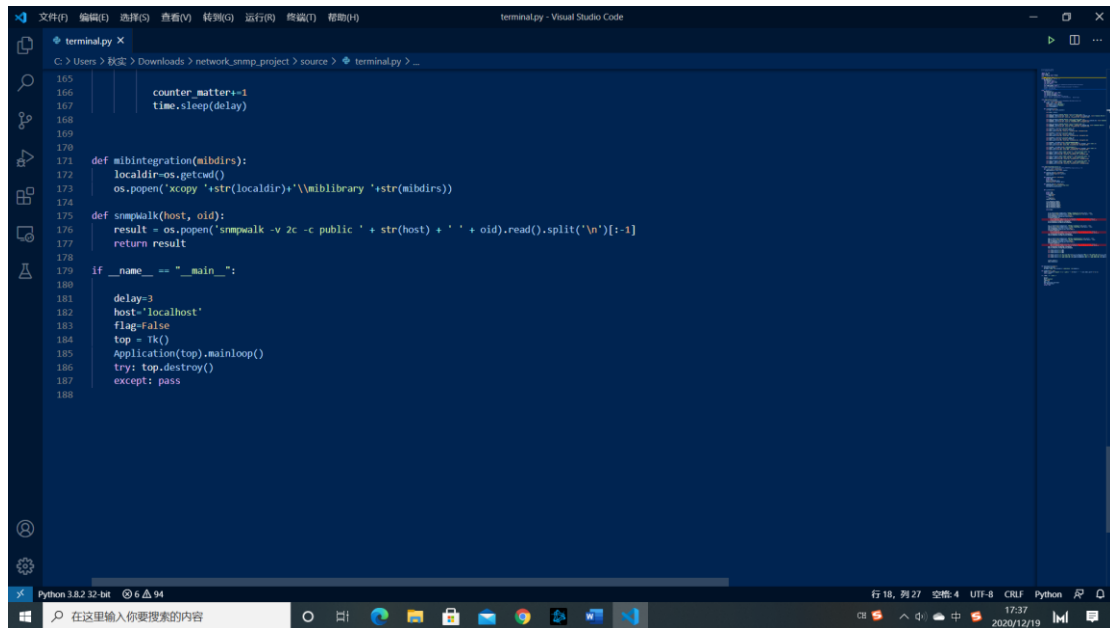
124
125 ip_in_result=float(snmpwalk(host,'IP-MIB::ipInDelivers.0')[0].split(' ')[3])
126 ip_out_result=float(snmpwalk(host,'IP-MIB::ipOutRequests.0')[0].split(' ')[3])
127 ip_in_datagrams=round(ip_in_result/1024,2)
128 ip_out_datagrams=round(ip_out_result/1024,2)
129 if(counter_matter>0):
130     ip_in_datagrams_speed=round((ip_in_datagrams-ip_in_datagrams_storage)/delay,2)
131     ip_out_datagrams_speed=round((ip_out_datagrams-ip_out_datagrams_storage)/delay,2)
132     ip_in_datagrams_storage=ip_in_datagrams
133     ip_out_datagrams_storage=ip_out_datagrams
134
135
136 tcp_in_result=float(snmpwalk(host,'TCP-MIB::tcpInSegs.0')[0].split(' ')[3])
137 tcp_out_result=float(snmpwalk(host,'TCP-MIB::tcpOutSegs.0')[0].split(' ')[3])
138 tcp_in_datagrams=round(tcp_in_result/1024,2)
139 tcp_out_datagrams=round(tcp_out_result/1024,2)
140 if(counter_matter>0):
141     tcp_in_datagrams_speed=round((tcp_in_datagrams-tcp_in_datagrams_storage)/delay,2)
142     tcp_out_datagrams_speed=round((tcp_out_datagrams-tcp_out_datagrams_storage)/delay,2)
143     tcp_in_datagrams_storage=tcp_in_datagrams
144     tcp_out_datagrams_storage=tcp_out_datagrams
145
146
147 udp_in_result=float(snmpwalk(host,'UDP-MIB::udpInDatagrams.0')[0].split(' ')[3])
148 udp_out_result=float(snmpwalk(host,'UDP-MIB::udpOutDatagrams.0')[0].split(' ')[3])
149 udp_in_datagrams=round(udp_in_result/1024,2)
150 udp_out_datagrams=round(udp_out_result/1024,2)
151 if(counter_matter>0):
152     udp_in_datagrams_speed=round((udp_in_datagrams-udp_in_datagrams_storage)/delay,2)
153     udp_out_datagrams_speed=round((udp_out_datagrams-udp_out_datagrams_storage)/delay,2)
154     udp_in_datagrams_storage=udp_in_datagrams
155     udp_out_datagrams_storage=udp_out_datagrams
156
157 self.Text1.delete('0.0',END)
158 self.Text2.delete('0.0',END)
159 self.Text3.delete('0.0',END)
160
161
self.Text1.insert(END,actr/in_out_datagrams_speed+'actr/in_out_datagrams_s...
```

因为 snmpwalk 直接获取的返回值是字符串，需要对其进行转换并计算为 mb 形式，以便于用户使用。

```
terminal.py X
C:\Users\秋实> Downloads > network_snmp_project > source > terminal.py > _
137 tcp_out_result=float(snmpwalk(host,'TCP-MIB:tcpOutSegs.0')[0].split(' ')[3])
138 tcp_in_datagrams=round(tcp_in_result/1024,2)
139 tcp_out_datagrams=round(tcp_out_result/1024,2)
140 if(counter_matter>0):
141     tcp_in_datagrams_speed=round((tcp_in_datagrams-tcp_in_datagrams_storage)/delay,2)
142     tcp_out_datagrams_speed=round((tcp_out_datagrams-tcp_out_datagrams_storage)/delay,2)
143     tcp_in_datagrams_storage=tcp_in_datagrams
144     tcp_out_datagrams_storage=tcp_out_datagrams
145
146
147 udp_in_result=float(snmpwalk(host,'UDP-MIB:udpInDatagrams.0')[0].split(' ')[3])
148 udp_out_result=float(snmpwalk(host,'UDP-MIB:udpOutDatagrams.0')[0].split(' ')[3])
149 udp_in_datagrams=round(udp_in_result/1024,2)
150 udp_out_datagrams=round(udp_out_result/1024,2)
151 if(counter_matter>0):
152     udp_in_datagrams_speed=round((udp_in_datagrams-udp_in_datagrams_storage)/delay,2)
153     udp_out_datagrams_speed=round((udp_out_datagrams-udp_out_datagrams_storage)/delay,2)
154     udp_in_datagrams_storage=udp_in_datagrams
155     udp_out_datagrams_storage=udp_out_datagrams
156
157 self.Text1.delete('0.0',END)
158 self.Text2.delete('0.0',END)
159 self.Text3.delete('0.0',END)
160
161 self.Text1.insert('1.0','IP总发送流量:'+str(ip_out_datagrams)+'MB'\n'+IP总接收流量:'+str(ip_in_datagrams)+'MB'\n'+IP发送流量速度:'+str(ip_out_datagrams_spe
self.Text2.insert('1.0','TCP总发送流量:'+str(tcp_out_datagrams)+'MB'\n'+TCP总接收流量:'+str(tcp_in_datagrams)+'MB'\n'+TCP发送流量速度:'+str(tcp_out_datagra
self.Text3.insert('1.0','UDP总发送流量:'+str(udp_out_datagrams)+'MB'\n'+UDP总接收流量:'+str(udp_in_datagrams)+'MB'\n'+UDP发送流量速度:'+str(udp_out_datagra
162
163 counter_matter+=1
164 time.sleep(delay)
165
166
167
168
169
170
171 def mibintegration(mibdirs):
172     localdir=os.getcwd()
173     os.popen('xcopy '+str(localdir)+'\\miblibrary '+str(mibdirs))
174
175
Python 3.8.2 32-bit 6 94
行 18, 列 27 空格 4 UTF-8 CRLF Python 17:37 2020/12/19
```

界面的 GUI 由 tkinter 库实现。Mib 库在不同环境下的移植通过向 SNMP 的 mibdirs 目录写入内嵌的 mib 库的方式来实现。

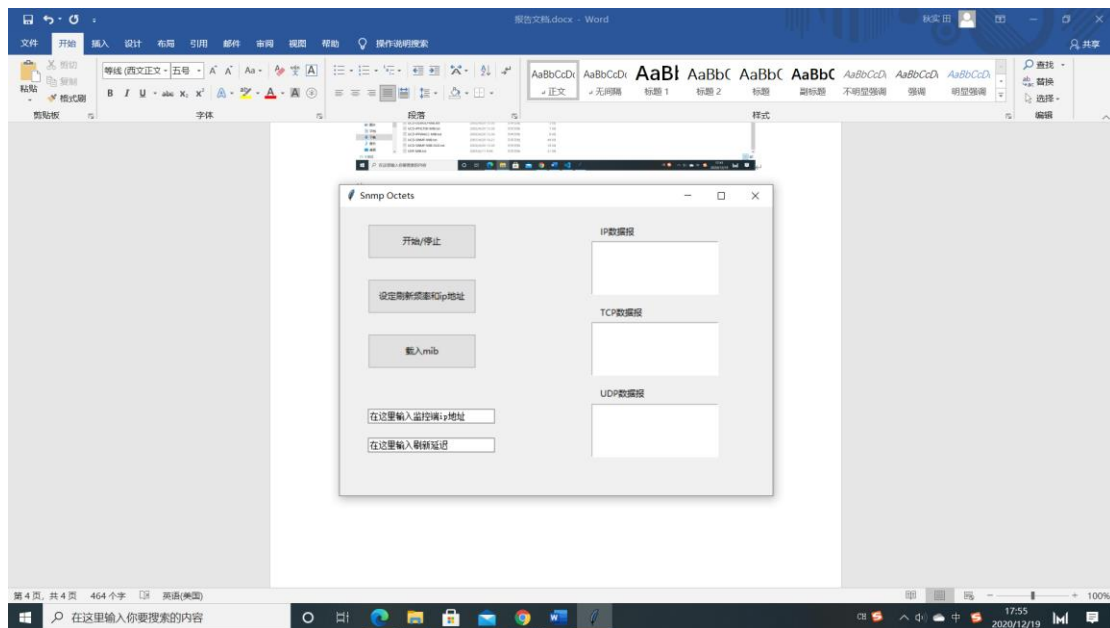
```
terminal.py X
C:\Users\秋实> Downloads > network_snmp_project > source > terminal.py > _
31 self.master.title('snmp Octets')
32 self.master.geometry('600x400')
33 self.createwidgets()
34
35 def createwidgets(self):
36     self.top = self.winfo_toplevel()
37
38     self.style = Style()
39
40     self.style.configure('Command1.TButton',font=('microsoft yahei',9))
41     self.Command1 = Button(self.top, text='开始/停止', command=self.Command1_Cmd, style='Command1.TButton')
42     self.Command1.place(relx=0.065, rely=0.06, relwidth=0.250, relheight=0.120)
43
44     self.style.configure('Command2.TButton',font=('microsoft yahei',9))
45     self.Command2 = Button(self.top, text='设定刷新频率和ip地址', command=self.Command2_Cmd, style='Command2.TButton')
46     self.Command2.place(relx=0.065, rely=0.25, relwidth=0.250, relheight=0.120)
47
48     self.style.configure('Command3.TButton',font=('microsoft yahei',9))
49     self.Command3 = Button(self.top, text='载入mib', command=self.Command3_Cmd, style='Command3.TButton')
50     self.Command3.place(relx=0.065, rely=0.44, relwidth=0.250, relheight=0.120)
51
52     self.Text1Font = Font(font=('microsoft yahei',9))
53     self.Text1 = Text(self.top, font=self.Text1Font)
54     self.Text1.place(relx=0.582, rely=0.12, relwidth=0.293, relheight=0.185)
55
56     self.Text2Font = Font(font=('microsoft yahei',9))
57     self.Text2 = Text(self.top, font=self.Text2Font)
58     self.Text2.place(relx=0.582, rely=0.401, relwidth=0.293, relheight=0.185)
59
60     self.Text3Font = Font(font=('microsoft yahei',9))
61     self.Text3 = Text(self.top, font=self.Text3Font)
62     self.Text3.place(relx=0.582, rely=0.682, relwidth=0.293, relheight=0.185)
63
64     self.Text4Var = StringVar(value='在这里输入监控端ip地址')
65     self.Text4 = Entry(self.top, text='Text4', textvariable=self.Text4Var, font=('宋体',9))
66     self.Text4.place(relx=0.065, rely=0.700, relwidth=0.293, relheight=0.050)
67
68     self.Text5Var = StringVar(value='在这里输入要监控的ip')
69     self.Text5 = Entry(self.top, text='Text5', textvariable=self.Text5Var, font=('宋体',9))
70     self.Text5.place(relx=0.345, rely=0.700, relwidth=0.293, relheight=0.050)
71
Python 3.8.2 32-bit 6 94
行 18, 列 27 空格 4 UTF-8 CRLF Python 17:37 2020/12/19
```



```
165
166     counter_matter+=1
167     time.sleep(delay)
168
169
170
171 def mibintegration(mibdirs):
172     localdir=os.getcwd()
173     os.popen('xcopy '+str(localdir)+'\\miblibrary '+str(mibdirs))
174
175 def snmpwalk(host, oid):
176     result = os.popen('snmpwalk -v 2c -c public ' + str(host) + ' ' + oid).read().split('\n')[1:-1]
177     return result
178
179 if __name__ == "__main__":
180
181     delay=3
182     host='localhost'
183     flag=False
184     top = Tk()
185     Application(top).mainloop()
186     try: top.destroy()
187     except: pass
188
```

三、程序测试截图和说明

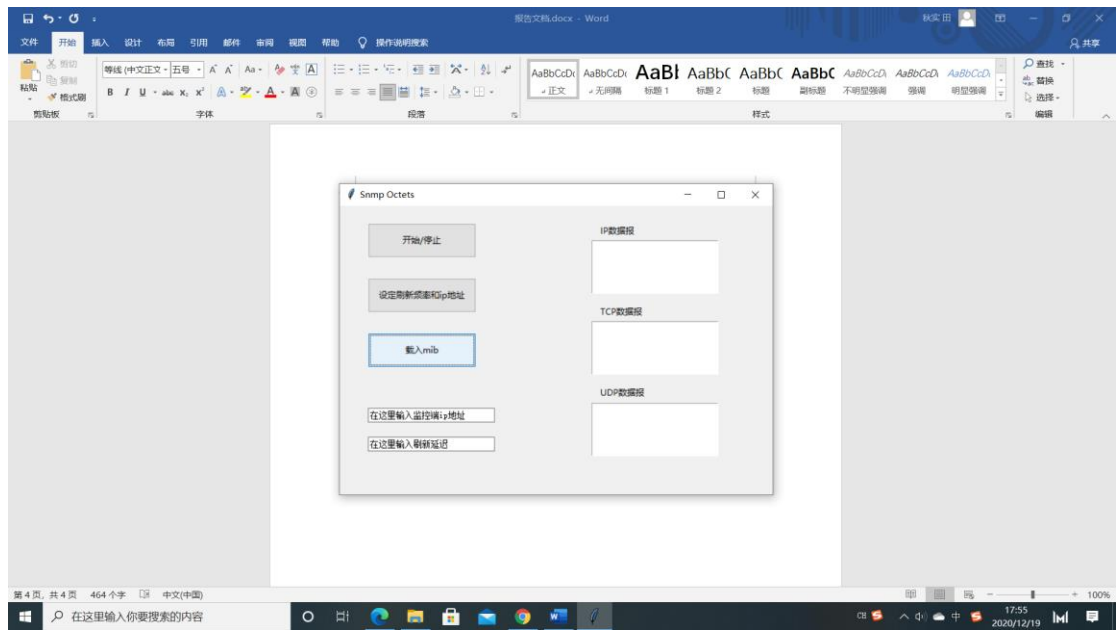
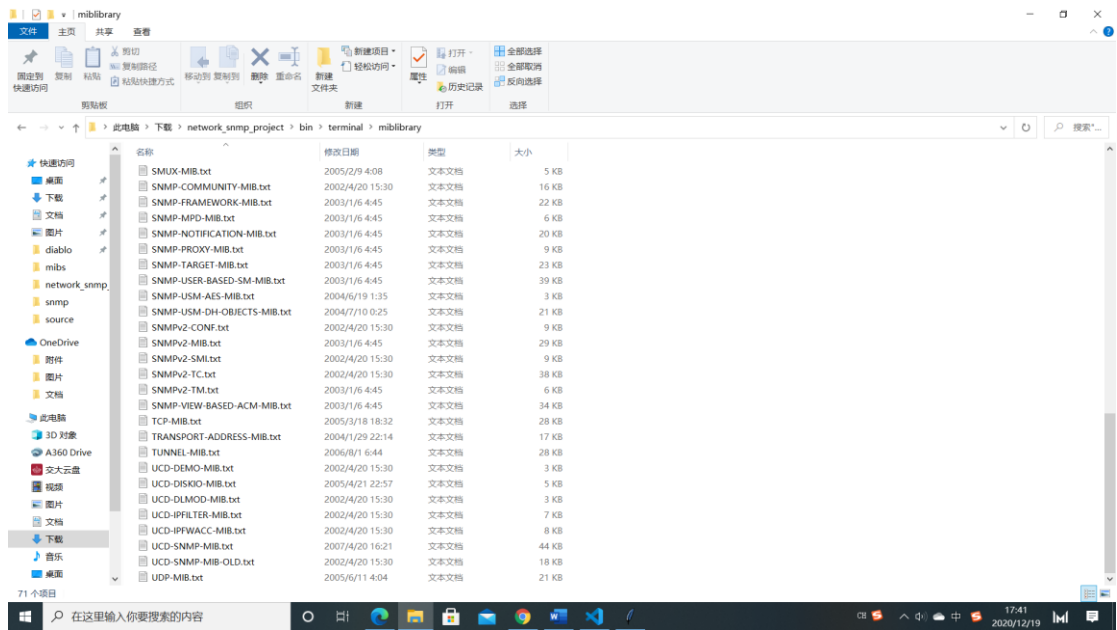
功能 1：实现 GUI。



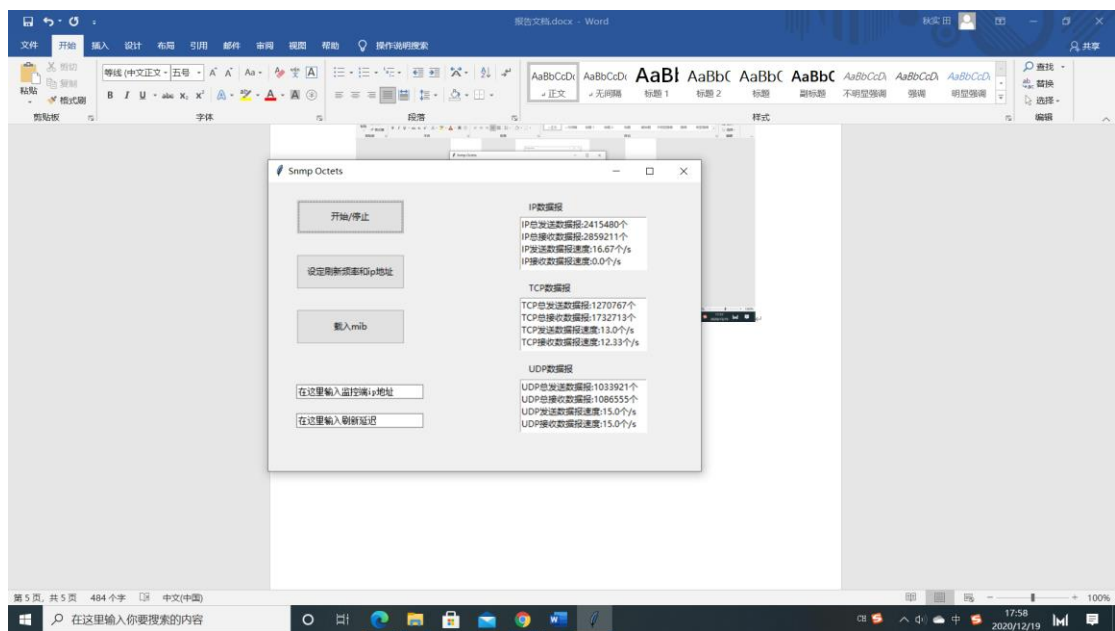
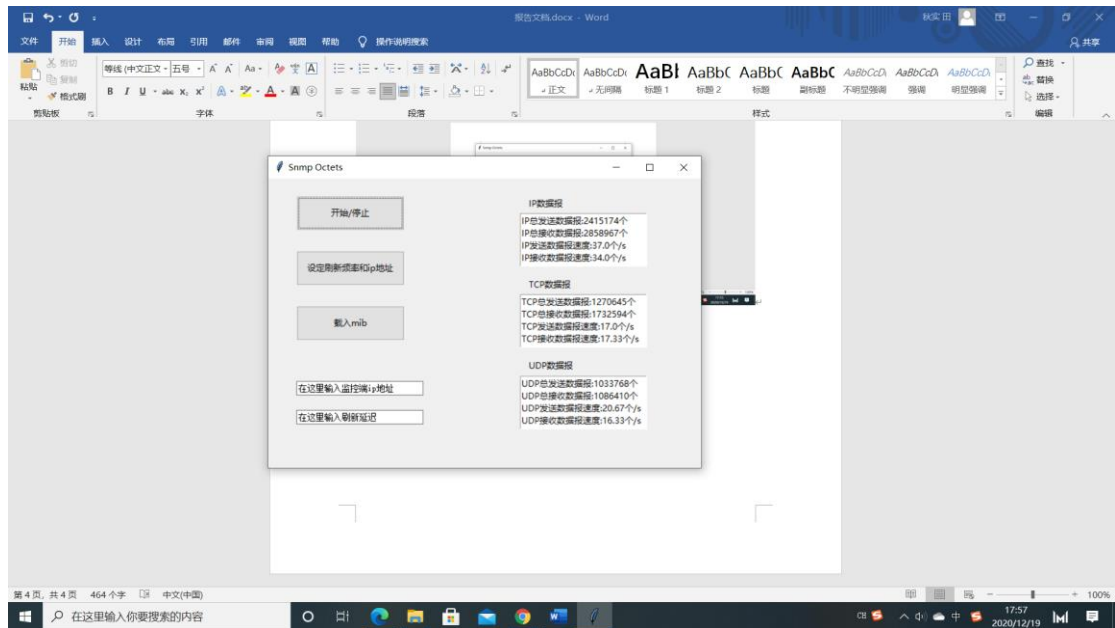
如图，可见实现的 GUI 界面。

功能 2：导入整个 mib 库到程序中。

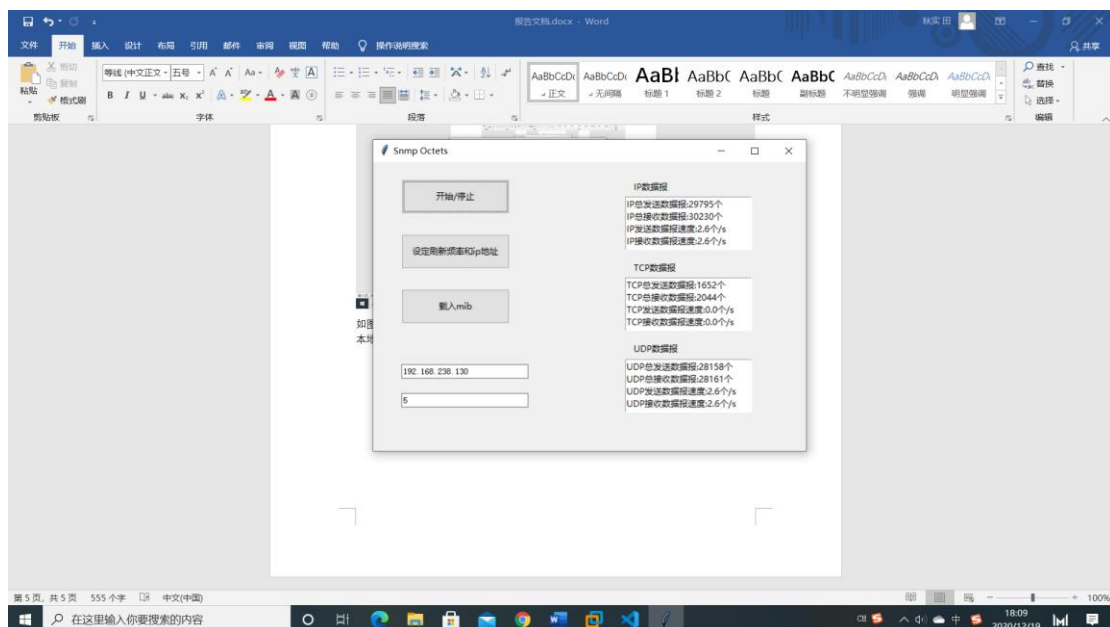
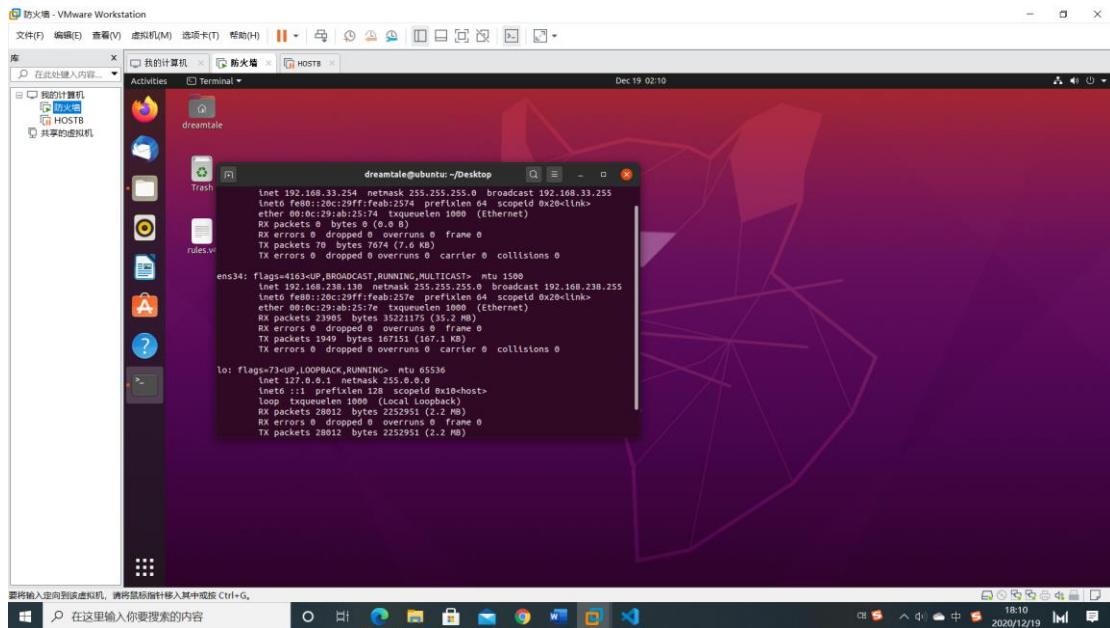
在程序目录中的 miblibrary 目录下有全部用到的 mib 库。用户也可以自行导入这些库。通过界面上的导载入 mib 按钮，可以实现在不同环境下程序的正常运行。



功能 3：对于收到、发出的 TCP、UDP、IP 包统计个数。运行比较长的一段时间，能够比较流量变化。



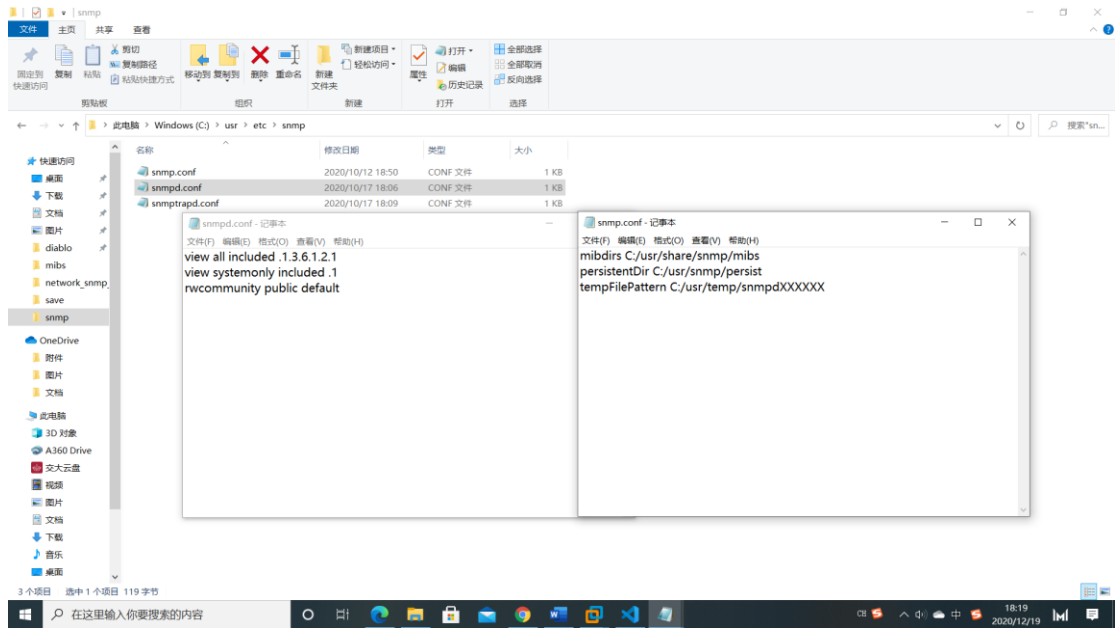
如图，可以自行设定刷新频率和监控主机，来监测流量情况。默认的刷新延迟为 3s，主机为本地（localhost）。通过虚拟机测试，程序也可以监测其他主机的流量情况：



(192.168.238.130 为和主机在同一网段的虚拟机 ip) 需要监控端安装了 snmpd 并开放 161 端口。

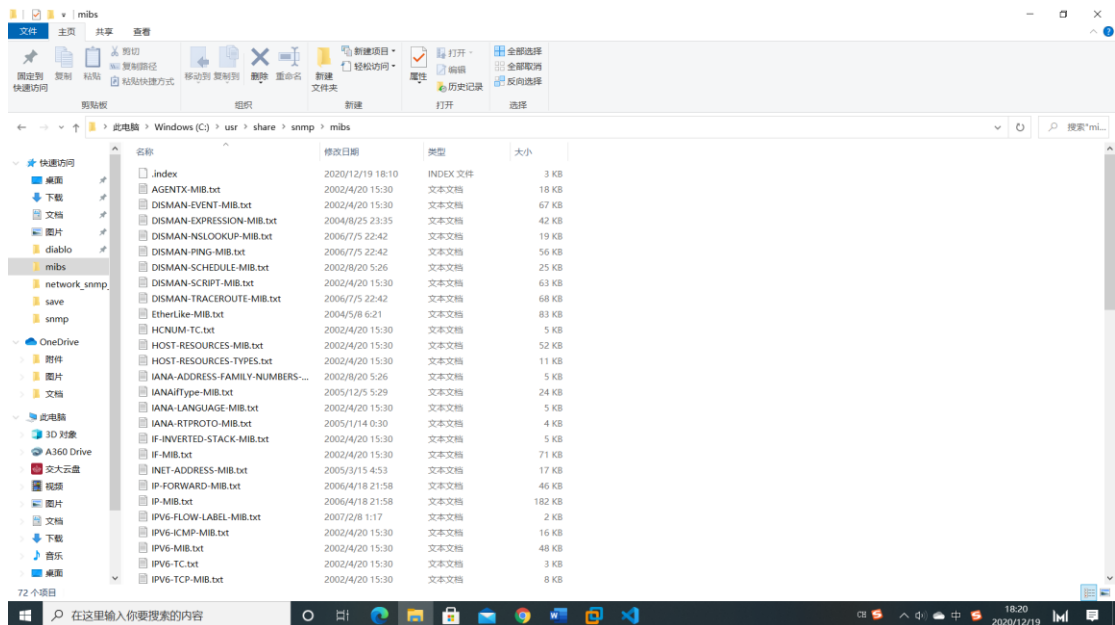
四、遇到的问题及解决方法

1. windows 下安装和配置 netsnmp 有困难。一开始我是打算在 linux 环境完成开发的，但是 vmware 屡屡报错，安装包的时候出现了各种奇怪的问题，所以我不得不使用 windows 环境。Windows 环境下安装 netsnmp 需要先安装 perl 环境。默认安装完成之后需要进行配置工作才能让 net SNMP 正常运行。



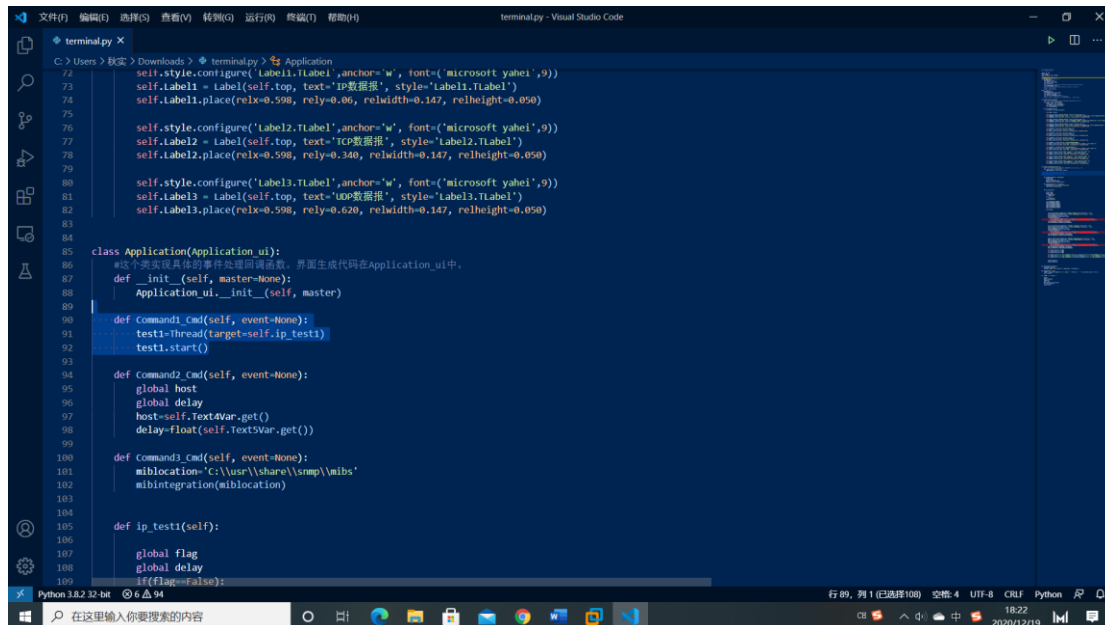
如上，编写两个控制文件。这部分的资料在网上很难找，因为大部分的资源都是 Linux 环境下的。最后要把 net SNMP 注册成服务并添加到环境变量方便使用，这一块配置因为资料少很棘手。

2.mib 库对应的资料太少，难以查询



这次作业任务要求的流量数据报所对应的 mib 值在网上基本没什么资料可查，只能对照 mib 文档一个个寻找，很花时间。

3.gui 多线程操作干扰问题



```
72 self.style.configure('Label1.TLabel', anchor='w', font=('microsoft yahei', 9))
73 self.Label1 = Label(self.top, text='IP数据报', style='Label1.TLabel')
74 self.Label1.place(relx=0.598, rely=0.06, relwidth=0.147, relheight=0.050)
75
76 self.style.configure('Label2.TLabel', anchor='w', font=('microsoft yahei', 9))
77 self.Label2 = Label(self.top, text='TCP数据报', style='Label2.TLabel')
78 self.Label2.place(relx=0.598, rely=0.340, relwidth=0.147, relheight=0.050)
79
80 self.style.configure('Label3.TLabel', anchor='w', font=('microsoft yahei', 9))
81 self.Label3 = Label(self.top, text='UDP数据报', style='Label3.TLabel')
82 self.Label3.place(relx=0.598, rely=0.620, relwidth=0.147, relheight=0.050)
83
84
85 class Application(Application_ui):
86     #这个类实现具体的事件处理回调函数，界面生成代码在Application_ui中。
87     def __init__(self, master=None):
88         Application_ui.__init__(self, master)
89
90     def Command1_Cmd(self, event=None):
91         test1=Thread(target=self.ip_test1)
92         test1.start()
93
94     def Command2_Cmd(self, event=None):
95         global host
96         host=self.Text4Var.get()
97         delay=float(self.Text5Var.get())
98
99     def Command3_Cmd(self, event=None):
100         miblocation='C:\\usr\\share\\snmp\\mibs'
101         mibintegration(miblocation)
102
103
104
105     def ip_test1(self):
106         global flag
107         global delay
108         if flag==False:
109
```

刚开始编写完程序进行测试的时候，我发现一点击开始按钮，程序进入 while 循环后就会卡死，令人百思不得其解。在查阅了相关资料后，我得知 gui 的 mainloop 和子函数的逻辑循环之间存在冲突，需要分割成两个不同的线程才能正常运行。在插入了 Thread 库线程函数后，这个问题就解决了。

五、体会与建议

在本次的大作业编写完成过程中，我初步了解了 SNMP 协议的使用方法和基本原理，基本掌握了简单的 Python GUI 程序编写流程和常用函数，对计算机网络流量监控有了一些基本认识，对于及 netsnmp、snmptuil 这几个工具的配置和使用也算是比较熟练了。希望我以后能在学习和实习中对 SNMP 这个管理工具有更深入的领会和进一步的学习了解。

关于对大作业的建议这一块，我希望以后计网大作业的任务指令能够更加明确清晰一些。比如这个 SNMP 大作业的“导入 mib 库”和第一个发包器实验的“支持 PCP 文件”就让人有些看不懂具体的意思。最后，希望课程越办越好。