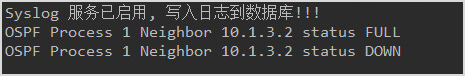
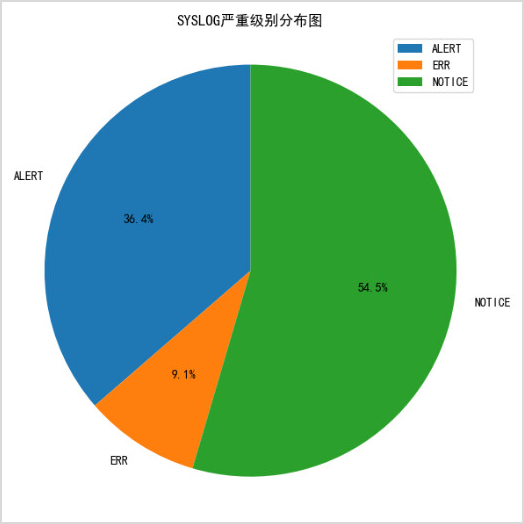
**经典网络协议 第十一天作业**

1. **使用Python监控Syslog Trap!监控OSPF邻居状态!再对Syslog进行基本分词(与上课相同)写入数据库并分析**

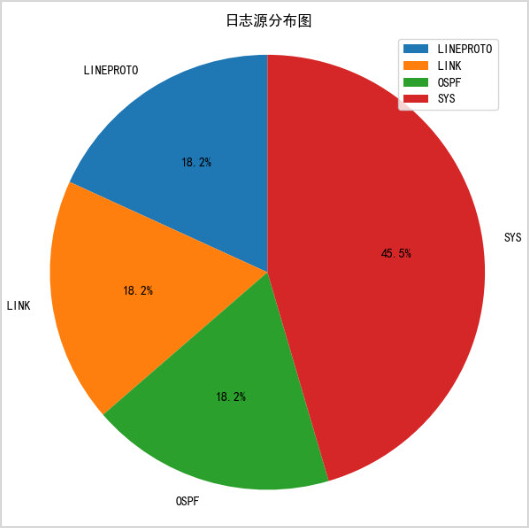
**最终结果如下图, Python对OSPF邻居状态改变的Syslog Trap进行分析,并打印!**



**读取数据库!产生SYSLOG严重级别分布图**



**读取数据库!产生SYSLOG日志源分布图**



**使用Python监控Syslog Trap!监控OSPF邻居状态!再对Syslog进行基本分词(与上课相同)写入数据库并分析**

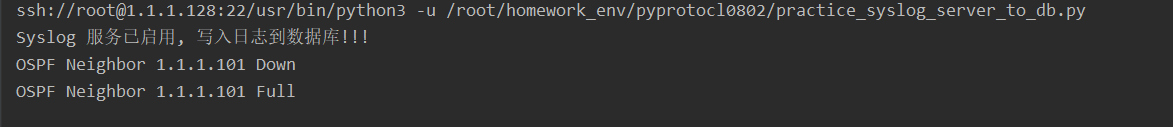
**Server\_to\_db代码：**

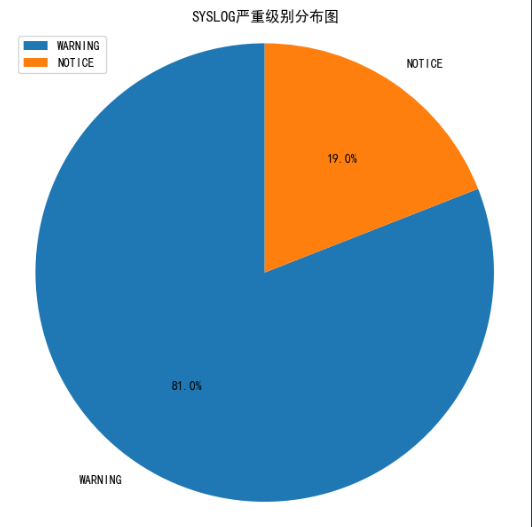
#!/usr/bin/env python3  
# -\*- coding=utf-8 -\*-  
# 本脚由亁颐堂现任明教教主编写，用于乾颐盾Python课程！  
# 教主QQ:605658506  
# 亁颐堂官网www.qytang.com  
# 教主技术进化论拓展你的技术新边疆  
# https://ke.qq.com/course/271956?tuin=24199d8a  
  
  
import logging  
import socketserver  
import threading  
import re  
from dateutil import parser  
import os  
import sqlite3  
from monitor\_ospf\_state import snmp\_trap\_receiver  
from datetime import datetime  
  
# facility与ID的对应关系的字典  
facility\_dict = {0: 'KERN',  
 1: 'USER',  
 2: 'MAIL',  
 3: 'DAEMON',  
 4: 'AUTH',  
 5: 'SYSLOG',  
 6: 'LPR',  
 7: 'NEWS',  
 8: 'UUCP',  
 9: 'CRON',  
 10: 'AUTHPRIV',  
 11: 'FTP',  
 16: 'LOCAL0',  
 17: 'LOCAL1',  
 18: 'LOCAL2',  
 19: 'LOCAL3',  
 20: 'LOCAL4',  
 21: 'LOCAL5',  
 22: 'LOCAL6',  
 23: 'LOCAL7'}  
  
# severity\_level与ID的对应关系的字典  
severity\_level\_dict = {0: 'EMERG',  
 1: 'ALERT',  
 2: 'CRIT',  
 3: 'ERR',  
 4: 'WARNING',  
 5: 'NOTICE',  
 6: 'INFO',  
 7: 'DEBUG'}  
  
  
  
class SyslogUDPHandler(socketserver.BaseRequestHandler):  
 def handle(self):  
 data = bytes.decode(self.request[0].strip()) # 读取数据  
 print(data)  
 syslog\_info\_dict = {'device\_ip': self.client\_address[0]}  
 try:  
 # <187>83: \*Apr 4 00:03:12.969: %LINK-3-UPDOWN: Interface GigabitEthernet2, changed state to up  
 syslog\_info = re.match(r'^<(\d\*)>(\d\*): \\*(.\*): %(\w+)-(\d)-(\w+): (.\*)', str(data)).groups()  
 # print(syslog\_info[0]) 提取为整数 例如 185  
 # 185 二进制为 1011 1001  
 # 前5位为facility >> 3 获取前5位  
 # 后3位为severity\_level & 0b111 获取后3位  
 syslog\_info\_dict['facility'] = int(syslog\_info[0]) >> 3  
 syslog\_info\_dict['facility\_name'] = facility\_dict[int(syslog\_info[0]) >> 3]  
 syslog\_info\_dict['logid'] = int(syslog\_info[1])  
 syslog\_info\_dict['time'] = parser.parse(syslog\_info[2])  
 syslog\_info\_dict['log\_source'] = syslog\_info[3]  
 syslog\_info\_dict['severity\_level'] = int(syslog\_info[4])  
 syslog\_info\_dict['severity\_level\_name'] = severity\_level\_dict[int(syslog\_info[4])]  
 syslog\_info\_dict['description'] = syslog\_info[5]  
 syslog\_info\_dict['text'] = syslog\_info[6]  
 except AttributeError:  
 # 有些日志会缺失%SYS-5-CONFIG\_I, 造成第一个正则表达式无法匹配 , 也无法提取severity\_level  
 # 下面的icmp的debug就是示例  
 # <191>91: \*Apr 4 00:12:29.616: ICMP: echo reply rcvd, src 10.1.1.80, dst 10.1.1.253, topology BASE, dscp 0 topoid 0  
 syslog\_info = re.match(r'^<(\d\*)>(\d\*): \\*(.\*): (\w+): (.\*)', str(data)).groups()  
 print(syslog\_info[0])  
 syslog\_info\_dict['facility'] = int(syslog\_info[0]) >> 3  
 syslog\_info\_dict['facility\_name'] = facility\_dict[int(syslog\_info[0]) >> 3]  
 syslog\_info\_dict['logid'] = int(syslog\_info[1])  
 syslog\_info\_dict['time'] = parser.parse(syslog\_info[2])  
 syslog\_info\_dict['log\_source'] = syslog\_info[3]  
 # 如果在文本部分解析不了severity\_level, 切换到syslog\_info[0]去获取  
 # 185 二进制为 1011 1001  
 # 前5位为facility >> 3 获取前5位  
 # 后3位为severity\_level & 0b111 获取后3位  
 syslog\_info\_dict['severity\_level'] = int(syslog\_info[0]) & 0b111  
 syslog\_info\_dict['severity\_level\_name'] = severity\_level\_dict[(int(syslog\_info[0]) & 0b111)]  
 syslog\_info\_dict['description'] = 'N/A'  
 syslog\_info\_dict['text'] = syslog\_info[4]  
 print(syslog\_info\_dict)  
 conn = sqlite3.connect(gl\_dbname)  
 cursor = conn.cursor()  
 cursor.execute("insert into syslogdb (time, \  
 device\_ip, \  
 facility, \  
 facility\_name, \  
 severity\_level, \  
 severity\_level\_name, \  
 logid, \  
 log\_source, \  
 description, \  
 text) values ('%s', '%s', %d, '%s', %d, '%s', %d, '%s', '%s', '%s')" % (syslog\_info\_dict['time'].strftime("%Y-%m-%d %H:%M:%S"),  
 syslog\_info\_dict['device\_ip'],  
 syslog\_info\_dict['facility'],  
 syslog\_info\_dict['facility\_name'],  
 syslog\_info\_dict['severity\_level'],  
 syslog\_info\_dict['severity\_level\_name'],  
 syslog\_info\_dict['logid'],  
 syslog\_info\_dict['log\_source'],  
 syslog\_info\_dict['description'],  
 syslog\_info\_dict['text'],  
 ))  
 conn.commit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # 使用Linux解释器 & WIN解释器  
 global gl\_dbname  
 gl\_dbname = 'syslog.sqlite'  
 if os.path.exists(gl\_dbname):  
 os.remove(gl\_dbname)  
 # 连接数据库  
 conn = sqlite3.connect(gl\_dbname)  
 cursor = conn.cursor()  
 # 创建数据库  
  
 cursor.execute("create table syslogdb(id INTEGER PRIMARY KEY AUTOINCREMENT,\  
 time varchar(64), \  
 device\_ip varchar(32),\  
 facility int,\  
 facility\_name varchar(32),\  
 severity\_level int,\  
 severity\_level\_name varchar(32),\  
 logid int,\  
 log\_source varchar(32), \  
 description varchar(128), \  
 text varchar(1024)\  
 )")  
 conn.commit()  
 try:  
 HOST, PORT = "0.0.0.0", 514 # 本地地址与端口  
 server = socketserver.UDPServer((HOST, PORT), SyslogUDPHandler) # 绑定本地地址，端口和syslog处理方法  
 print("Syslog 服务已启用, 写入日志到数据库!!!")  
 snmp\_trap\_receiver("ens33")  
 server.serve\_forever(poll\_interval=0.5) # 运行服务器，和轮询间隔  
  
 except (IOError, SystemExit):  
 raise  
 except KeyboardInterrupt: # 捕获Ctrl+C，打印信息并退出  
 print("Crtl+C Pressed. Shutting down.")  
 finally:  
 conn.commit()

**syslog 严重级别代码**

import sqlite3  
from dateutil import parser  
from matplotlib import pyplot as plt  
from practice\_syslog\_server\_to\_db import severity\_level\_dict  
  
  
def syslog\_show(dbname):  
 # 连接数据库  
 conn = sqlite3.connect(dbname)  
 cursor = conn.cursor()  
 # 提取时间与CPU利用率信息  
 cursor.execute("select severity\_level as level,*COUNT*(\*) as count from syslogdb group by severity\_level")  
 yourresults = cursor.fetchall()  
  
 level\_list = []  
 count\_list = []  
 print(yourresults)  
 # 把结果写入time\_list和cpu\_list的列表  
 for level\_count in yourresults:  
 level\_list.append(severity\_level\_dict[level\_count[0]])  
 count\_list.append(level\_count[1])  
  
 print(level\_list)  
 print(count\_list)  
 print([float(count) for count in count\_list])  
  
 plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文  
 # 调节图形大小，宽，高  
 plt.figure(figsize=(6, 6))  
  
 # 使用count\_list的比例来绘制饼图  
 # 使用level\_list作为注释  
 patches, l\_text, p\_text = plt.pie(count\_list,  
 labels=level\_list,  
 labeldistance=1.1,  
 autopct='%3.1f%%',  
 shadow=False,  
 startangle=90,  
 pctdistance=0.6)  
  
 # labeldistance，文本的位置离远点有多远，1.1指1.1倍半径的位置  
 # autopct，圆里面的文本格式，%3.1f%%表示小数有三位，整数有一位的浮点数  
 # shadow，饼是否有阴影  
 # startangle，起始角度，0，表示从0开始逆时针转，为第一块。一般选择从90度开始比较好看  
 # pctdistance，百分比的text离圆心的距离  
 # patches, l\_texts, p\_texts，为了得到饼图的返回值，p\_texts饼图内部文本的，l\_texts饼图外label的文本  
  
 # 改变文本的大小  
 # 方法是把每一个text遍历。调用set\_size方法设置它的属性  
 for t in l\_text:  
 t.set\_size = 30  
 for t in p\_text:  
 t.set\_size = 20  
 # 设置x，y轴刻度一致，这样饼图才能是圆的  
 plt.axis('equal')  
 plt.title('SYSLOG严重级别分布图') # 主题  
 plt.legend()  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 syslog\_show("syslog.sqlite")

**运行结果：**





**日志源分布图代码**

import sqlite3  
from dateutil import parser  
from matplotlib import pyplot as plt  
from practice\_syslog\_server\_to\_db import severity\_level\_dict  
  
  
def syslog\_show(dbname):  
 # 连接数据库  
 conn = sqlite3.connect(dbname)  
 cursor = conn.cursor()  
 # 提取时间与CPU利用率信息  
 cursor.execute("select log\_source ,*COUNT*(\*) as count from syslogdb group by log\_source")  
 yourresults = cursor.fetchall()  
 print(yourresults)  
  
 level\_list = []  
 count\_list = []  
  
 # 把结果写入time\_list和cpu\_list的列表  
 for log\_source in yourresults:  
 level\_list.append(log\_source[0])  
 count\_list.append(log\_source[1])  
  
 print(level\_list)  
 print([float(count) for count in count\_list])  
  
 plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文  
 # 调节图形大小，宽，高  
 plt.figure(figsize=(6, 6))  
  
 # 使用count\_list的比例来绘制饼图  
 # 使用level\_list作为注释  
 patches, l\_text, p\_text = plt.pie(count\_list,  
 labels=level\_list,  
 labeldistance=1.1,  
 autopct='%3.1f%%',  
 shadow=False,  
 startangle=90,  
 pctdistance=0.6)  
  
 # labeldistance，文本的位置离远点有多远，1.1指1.1倍半径的位置  
 # autopct，圆里面的文本格式，%3.1f%%表示小数有三位，整数有一位的浮点数  
 # shadow，饼是否有阴影  
 # startangle，起始角度，0，表示从0开始逆时针转，为第一块。一般选择从90度开始比较好看  
 # pctdistance，百分比的text离圆心的距离  
 # patches, l\_texts, p\_texts，为了得到饼图的返回值，p\_texts饼图内部文本的，l\_texts饼图外label的文本  
  
 # 改变文本的大小  
 # 方法是把每一个text遍历。调用set\_size方法设置它的属性  
 for t in l\_text:  
 t.set\_size = 30  
 for t in p\_text:  
 t.set\_size = 20  
 # 设置x，y轴刻度一致，这样饼图才能是圆的  
 plt.axis('equal')  
 plt.title('SYSLOG源分布图') # 主题  
 plt.legend()  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 syslog\_show("syslog.sqlite")

**运行结果：**

