**经典网络协议 第九天作业**

1. **使用SNMP获取MEM利用率!并写入数据库(采集时间超过一分钟)!然后从数据库读出绘制(只绘制最近一分钟[使用数据库时间过滤技术])MEM利用率走势的线性图**

**第一步(第一个函数): 使用SNMP读取内存信息(采集时间超过一分钟)!,并计算内存使用率, 读取周期为五秒! 把获取的数据写入数据库**

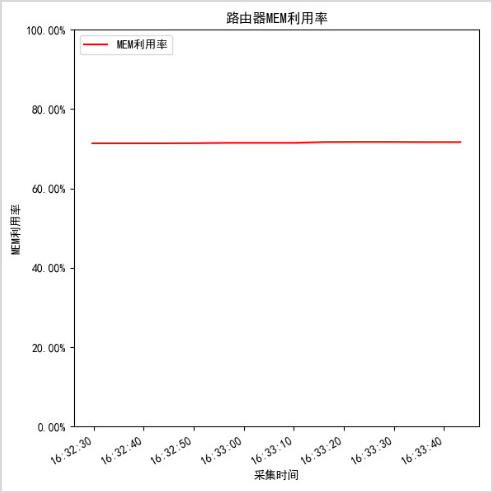
提示:创建数据库cursor.execute("create table routerdb(id INTEGER PRIMARY KEY AUTOINCREMENT, time timestamp, mem\_percent int)")

**第二步(第二个函数): 读取数据库, 返回时间和内存利用率的列表(只读取最近一分钟)**

**第三步(第三个函数): 根据第二步读取的数据,绘制内存利用率走势的线性图(只绘制最近一分钟)**

提示:ax.set\_ylim(0, 100) # 控制Y轴的取值范围

**线性图效果:**



代码：

import os  
import sqlite3  
from snmpv2\_get import snmpv2\_get  
import datetime  
import time  
import sqlite3  
from snmpv2\_get import snmpv2\_get  
import datetime  
import time  
import random  
import datetime  
  
from matplotlib import pyplot as plt  
import matplotlib  
# conn = sqlite3.connect('mem\_Total\_db.sqlite')  
# cursor = conn.cursor()  
# cursor.execute("create table routerdb(id INTEGER PRIMARY KEY AUTOINCREMENT, time timestamp, mem\_percent int)")  
  
# 读取CPU利用率写入数据库  
def write\_mem\_values\_db(seconds):  
 conn = sqlite3.connect('mem\_Total\_db.sqlite')  
 cursor = conn.cursor()  
 id = 0  
 while seconds > 0:  
 id+=1  
 now = time.time()  
 mem\_free = snmpv2\_get("1.1.1.200", "tcpipro", "1.3.6.1.4.1.9.9.48.1.1.1.6.1", port=161)[1]  
 # mem-used  
 mem\_used = snmpv2\_get("1.1.1.200", "tcpipro", "1.3.6.1.4.1.9.9.48.1.1.1.5.1", port=161)[1]  
 #mem\_total  
 mem\_total = int(mem\_free) + int(mem\_used)  
 mem\_util = int(mem\_used) / int(mem\_total)  
 mem\_pre = int(mem\_util\*100)  
 cursor.execute(f'insert into routerdb(id,time,mem\_percent) values("{id}","{now}","{mem\_pre}")')  
 # print(mem\_util)  
 cursor.fetchall()  
 conn.commit()  
 time.sleep(5)  
 seconds -= 5  
# 从数据库中读取时间和CPU利用率  
mem\_values\_dict = {}  
def read\_mem\_values\_db():  
  
 conn = sqlite3.connect('mem\_Total\_db.sqlite')  
 cursor = conn.cursor()  
 cursor.execute('select \* from routerdb')  
 all\_result = cursor.fetchall()  
 for x in all\_result:  
 mem\_values\_dict[x[1]]=x[2]  
 return mem\_values\_dict  
  
print(matplotlib.matplotlib\_fname())  
plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文  
plt.rcParams['font.family'] = 'sans-serif'  
colorlist = ['r', 'b', 'g', 'y']  
  
import matplotlib.pyplot as plt  
import matplotlib.dates as md  
import dateutil  
def mat\_line(cpu\_usage\_list):  
 # 调节图形大小，宽，高  
 fig = plt.figure(figsize=(6, 6))  
 # 一共一行, 每行一图, 第一图  
 ax = fig.add\_subplot(111)  
  
 # 处理X轴时间格式  
 import matplotlib.dates as mdate  
 # ax.xaxis.set\_major\_formatter(mdate.DateFormatter('%Y-%m-%d %H:%M:%S')) # 设置时间标签显示格式  
 ax.xaxis.set\_major\_formatter(mdate.DateFormatter('%H:%M')) # 设置时间标签显示格式  
  
 # 处理Y轴百分比格式  
 import matplotlib.ticker as mtick  
 ax.set\_ylim(0, 100)  
 ax.yaxis.set\_major\_formatter(mtick.FormatStrFormatter('%d%%'))  
  
 # 把cpu\_usage\_list的数据,拆分为x轴的时间,与y轴的利用率  
 x = []  
 y = []  
  
 for time, cpu in cpu\_usage\_list:  
 x.append(time)  
 y.append(cpu)  
  
 # 添加主题和注释  
 plt.title('路由器MEM利用率')  
 plt.xlabel('采集时间')  
 plt.ylabel('MEM利用率')  
  
 fig.autofmt\_xdate() # 当x轴太拥挤的时候可以让他自适应  
  
 # 实线红色  
 ax.plot(x, y, linestyle='solid', color='r', label='R1')  
 # 虚线黑色  
 # ax.plot(x, y, linestyle='dashed', color='b', label='R1')  
 ax.set\_ylim(bottom=0, top=100)  
 # 如果你有两套数据,完全可以在一幅图中绘制双线  
 # ax.plot(x2, y2, linestyle='dashed', color='b', label='R2')  
  
 # 设置说明的位置  
 ax.legend(loc='upper left')  
  
 # 保存到图片  
 plt.savefig('result1.png')  
 # 绘制图形  
 plt.show()  
import datetime  
if \_\_name\_\_ == '\_\_main\_\_':  
 # write\_mem\_values\_db(60)  
 cpu\_dict = read\_mem\_values\_db()  
 # print(cpu\_dict)  
  
 # 画折线图  
 line\_data = []  
 for i in cpu\_dict.items():  
 x = datetime.datetime.utcfromtimestamp(int(i[0]))  
 xy = x, i[1]  
 line\_data.append(xy)  
 print(line\_data)  
 mat\_line(line\_data)

运行结果：

