emulating RIP emulator 代码

目录

1．前端界面代码

1.1 主窗口代码

```
import os

import sys

from PyQt5.QtCore import Qt

from PyQt5.QtWidgets import QApplication, QWidget, QDesktopWidget,

QHBoxLayout, QVBoxLayout

from PyQt5.QtWidgets import QPushButton, QLineEdit, QTableWidget,

QTableWidgetItem

from PyQt5.QtWidgets import QMessageBox, QMenu

from utils.dialog import ResultDialog


from qt_material import apply_stylesheet


import time


BASE_DIR = os.path.dirname(os.path.realpath(sys.argv[0]))


RUNNING = 1

STOPPING = 2
```

```
STOP = 3



class MainWindow(QWidget):

    def __init__(self):

        super().__init__()


        self.switch = STOP


        # 控件

        self.txt_asin1 = None

        self.txt_asin2 = None

        self.txt_asin3 = None

        self.table_widget1 = None

        self.table_widget2 = None


        # 窗体标题和尺寸

        self.setWindowTitle('RIP simulation')


        # 窗体的尺寸

        self.resize(1000, 450)
```

```
        # 窗体位置
        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)

        # 创建布局
        layout = QVBoxLayout()
        layout.addLayout(self.init_form())
        layout.addLayout(self.init_table())
        layout.addLayout(self.init_footer())

        # 给窗体设置元素的排列方式
        self.setLayout(layout)

def init_form(self):
        # 2.创建上面标题布局
        form_layout = QHBoxLayout()

        # 2.1 输入框 left
        txt_asin1 = QLineEdit()
        # txt_asin1.setText("N1 3 A")
        txt_asin1.setPlaceholderText("target_net distance next_hop, for example,
```

```
N1 3 A")
        self.txt_asin1 = txt_asin1
        form_layout.addWidget(txt_asin1)

        # 2.2 添加按钮
        btn_add = QPushButton("ADD")
        btn_add.clicked.connect(self.event_add_click1)
        form_layout.addWidget(btn_add)

        # # right
        txt_asin2 = QLineEdit()
        txt_asin2.setPlaceholderText("target_net distance, for example, N2 4")
        self.txt_asin2 = txt_asin2
        form_layout.addWidget(txt_asin2)

        # 2.2 添加按钮
        btn_add2 = QPushButton("ADD")
        btn_add2.clicked.connect(self.event_add_click2)
        form_layout.addWidget(btn_add2)

        return form_layout
```

```
def init_table(self):
    # 3.创建中间的表格
    table_layout = QHBoxLayout()

    # 3.1 创建表格 left
    self.table_widget1 = table_widget1 = QTableWidget(0, 3)
    table_header = [
        {"field": "target_net", "text": "target net", 'width': 130},
        {"field": "distance", "text": "distance", 'width': 130},
        {"field": "next_hop", "text": "next hop", 'width': 130},

    ]
    # 用 table_header 创建第一列
    for idx, info in enumerate(table_header):
        item = QTableWidgetItem()
        item.setText(info['text'])
        table_widget1.setHorizontalHeaderItem(idx, item)
        table_widget1.setColumnWidth(idx, info['width'])

    # 3.1 创建表格 right
    self.table_widget2 = table_widget2 = QTableWidget(0, 2)
    table_header = [
```

```
        {"field": "target_net", "text": "target net", 'width': 130},
        {"field": "distance", "text": "distance", 'width': 130},
        # {"field": "metric", "text": "metric", 'width': 150},
]
# 用 table_header 创建第一列
for idx, info in enumerate(table_header):
        item = QTableWidgetItem()
        item.setText(info['text'])
        table_widget2.setHorizontalHeaderItem(idx, item)
        table_widget2.setColumnWidth(idx, info['width'])

# 3.2 初始化表格数据
# 读取数据文件
# left
import json
file_path = os.path.join(BASE_DIR, "db", "db1.json")
with open(file_path, mode='r', encoding='utf-8') as f:
        data = f.read()
data_list1 = json.loads(data)

current_row_count1 = table_widget1.rowCount()  # 当前表格有多少行
for row_list in data_list1:
```

```
        table_widget1.insertRow(current_row_count1)

        for i, ele in enumerate(row_list):
            cell = QTableWidgetItem(str(ele))

            if i in [0, 1, 2]:
                # 不可修改
                cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

            table_widget1.setItem(current_row_count1, i, cell)

        current_row_count1 += 1

    # # right
    file_path = os.path.join(BASE_DIR, "db", "db2.json")
    with open(file_path, mode='r', encoding='utf-8') as f:
        data = f.read()
    data_list2 = json.loads(data)

    current_row_count2 = table_widget2.rowCount()  # 当前表格有多少行
    for row_list in data_list2:
        table_widget2.insertRow(current_row_count2)
```

```
            for i, ele in enumerate(row_list):
                cell = QTableWidgetItem(str(ele))

                if i in [0, 1]:
                    # 不可修改
                    cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

                table_widget2.setItem(current_row_count2, i, cell)

            current_row_count2 += 1

        table_widget1.setContextMenuPolicy(Qt.CustomContextMenu)
table_widget1.customContextMenuRequested.connect(self.table_right_menu1)
        table_widget2.setContextMenuPolicy(Qt.CustomContextMenu)
table_widget2.customContextMenuRequested.connect(self.table_right_menu2)
        table_layout.addWidget(table_widget1)
```

```
        table_layout.addWidget(table_widget2)

        return table_layout

def init_footer(self):
        # 2.底部菜单  left
        footer_layout = QHBoxLayout()

        footer_layout.addStretch()

        btn_sort1 = QPushButton("sort")
        btn_sort1.clicked.connect(self.event_sort_click1)
        footer_layout.addWidget(btn_sort1)

        footer_layout.addStretch()

        btn_reinit1 = QPushButton("reinit")
        btn_reinit1.clicked.connect(self.event_reinit_click1)
        footer_layout.addWidget(btn_reinit1)

        footer_layout.addStretch()
```

```
btn_delete1 = QPushButton("delete")

btn_delete1.clicked.connect(self.event_delete_click1)

footer_layout.addWidget(btn_delete1)


footer_layout.addStretch()


txt_asin3 = QLineEdit()

txt_asin3.setPlaceholderText("neighbour route")

self.txt_asin3 = txt_asin3

footer_layout.addWidget(txt_asin3)


btn_run = QPushButton("run!")

btn_run.clicked.connect(self.event_run_click)

footer_layout.addWidget(btn_run)


footer_layout.addStretch()


btn_sort2 = QPushButton("sort")

btn_sort2.clicked.connect(self.event_sort_click2)

footer_layout.addWidget(btn_sort2)


footer_layout.addStretch()
```

```
        btn_reinit2 = QPushButton("reinit")
        btn_reinit2.clicked.connect(self.event_reinit_click2)
        footer_layout.addWidget(btn_reinit2)

        footer_layout.addStretch()

        btn_delete2 = QPushButton("delete")
        btn_delete2.clicked.connect(self.event_delete_click2)
        footer_layout.addWidget(btn_delete2)

        footer_layout.addStretch()

        return footer_layout

    def event_add_click1(self):
        # 1.获取输入框中的内容
        text = self.txt_asin1.text()
        # print(text)
        text = text.strip()
        if not text:
            QMessageBox.warning(self, "ERROR", "input error!")
```

```
        return
    # # B07YN82X3B=100
    self.txt_asin1.clear()

    target_net, distance, next_hop = text.split(" ")

    # 2.加入到表格中（型号、底价）
    new_row_list = [target_net, distance, next_hop]

    current_row_count = self.table_widget1.rowCount()  # 当前表格有多少行
    self.table_widget1.insertRow(current_row_count)
    for i, ele in enumerate(new_row_list):
        cell = QTableWidgetItem(str(ele))

        if i in [0, 1, 2]:
            # 不可修改
            cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

        self.table_widget1.setItem(current_row_count, i, cell)

    #                       QTableWidget.horizontalHeader().setSortIndicator(0,
Qt.AscendingOrder);
```

```
        pass


def event_add_click2(self):
        # 1.获取输入框中的内容
        text = self.txt_asin2.text()
        # print(text)
        text = text.strip()
        if not text:
            QMessageBox.warning(self, "ERROR", "route information error!")
            return
        # # B07YN82X3B=100
        target_net, distance = text.split(" ")


        self.txt_asin2.clear()


        # 2.加入到表格中（型号、底价）
        new_row_list = [target_net, distance]


        current_row_count = self.table_widget2.rowCount()  # 当前表格有多少行
        self.table_widget2.insertRow(current_row_count)
        for i, ele in enumerate(new_row_list):
```

```
                cell = QTableWidgetItem(str(ele))

                if i in [0, 1, 2]:
                    # 不可修改
                    cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

                self.table_widget2.setItem(current_row_count, i, cell)

            pass

    def event_reinit_click1(self):
        row_count = self.table_widget1.rowCount()  # 当前表格有多少行

        if not row_count:
            QMessageBox.warning(self, "ERROR", "no available rows!")
            return

        for rowNum in range(0, row_count)[::-1]:  # 逆序删除，正序删除会有一些删除不成功
            self.table_widget1.removeRow(rowNum)

    def event_reinit_click2(self):
```

```
        row_count = self.table_widget2.rowCount()  # 当前表格有多少行

        if not row_count:
            QMessageBox.warning(self, "ERROR", "no available rows!")
            return

        for rowNum in range(0, row_count)[::-1]:  # 逆序删除，正序删除会有一些删除不成功
            self.table_widget2.removeRow(rowNum)

    def event_delete_click1(self):
        # 1.获取已经选中的行
        row_list = self.table_widget1.selectionModel().selectedRows()

        if not row_list:
            QMessageBox.warning(self, "ERROR", "no available rows!")
            return

        # 2.翻转
        row_list.reverse()

        # 3.删除
```

```
        for row_object in row_list:
            index = row_object.row()
            self.table_widget1.removeRow(index)


    def event_delete_click2(self):
        # 1.获取已经选中的行
        row_list = self.table_widget2.selectionModel().selectedRows()

        if not row_list:
            QMessageBox.warning(self, "ERROR", "no available rows!")
            return

        # 2.翻转
        row_list.reverse()

        # 3.删除
        for row_object in row_list:
            index = row_object.row()
            self.table_widget2.removeRow(index)


    def event_sort_click1(self):
        self.table_widget1.sortItems(0, Qt.AscendingOrder)
```

```
def event_sort_click2(self):
    self.table_widget2.sortItems(0, Qt.AscendingOrder)


def table_right_menu1(self, pos):
    # 只有选中一行时，才支持右键
    selected_item_list = self.table_widget1.selectedItems()
    if len(selected_item_list) == 0:
        return

    menu = QMenu()
    item_copy = menu.addAction("copy")
    item_sort = menu.addAction("sort")
    item_delete = menu.addAction("delete")
    item_reinit = menu.addAction("reinit")
    item_run = menu.addAction("run")

    # 选中了那个?
    action = menu.exec_(self.table_widget1.mapToGlobal(pos))

    if action == item_copy:
        # 赋值当前型号 B08166SLDF
```

```
        clipboard = QApplication.clipboard()
        clipboard.setText(selected_item_list[0].text())

if action == item_delete:
        # if not selected_item_list:
        #         QMessageBox.warning(self, "ERROR", "no available elements!")
        #         return
        # 2.翻转
        selected_item_list.reverse()

        # 3.删除
        for row_object in selected_item_list:
                index = row_object.row()
                self.table_widget1.removeRow(index)

if action == item_reinit:
        row_count = self.table_widget1.rowCount()  # 当前表格有多少行

        if not row_count:
                QMessageBox.warning(self, "ERROR", "no available rows!")
                return
```

```
            for rowNum in range(0, row_count)[::-1]:  # 逆序删除，正序删除会有一
些删除不成功
                self.table_widget1.removeRow(rowNum)

        if action == item_run:
            route = self.txt_asin3.text()

            if not route:
                QMessageBox.warning(self, "ERROR", "no available route!")
                return

            row1 = self.table_widget1.rowCount()
            column1 = self.table_widget1.columnCount()

            if not row1:
                QMessageBox.warning(self, "ERROR", "no available route1!")
                return

            with open('./res/route1.txt', 'wt') as f:
                for i in range(0, row1):
                    for j in range(0, column1):
                        if j != column1 - 1:
```

```
                f.write(self.table_widget1.item(i, j).text() + " ")
            else:
                f.write(self.table_widget1.item(i, j).text())
        if i != row1 - 1:
            f.write("\n")

row2 = self.table_widget2.rowCount()
column2 = self.table_widget2.columnCount()

if not row2:
    QMessageBox.warning(self, "ERROR", "no available route2!")
    return

with open('./res/route2.txt', 'wt') as f:
    for i in range(0, row2):
        for j in range(0, column2):
            if j != column2 - 1:
                f.write(self.table_widget2.item(i, j).text() + " ")
            else:
                f.write(self.table_widget2.item(i, j).text())
        if i != row2 - 1:
            f.write("\n")
```

```
            with open('./res/route.txt', 'wt') as f:
                f.write(route)

            os.system("rip.exe")

            with open('./res/res.txt', 'rt') as f:
                data = f.read()
                f.close()

            dialog = ResultDialog(data)
            dialog.setWindowModality(Qt.ApplicationModal)
            dialog.exec_()

        if action == item_sort:
            self.table_widget1.sortItems(0, Qt.AscendingOrder)

    def table_right_menu2(self, pos):
        # 只有选中一行时，才支持右键
        selected_item_list = self.table_widget2.selectedItems()
        if len(selected_item_list) == 0:
            return
```

```
menu = QMenu()
item_copy = menu.addAction("copy")
item_sort = menu.addAction("sort")
item_delete = menu.addAction("delete")
item_reinit = menu.addAction("reinit")
item_run = menu.addAction("run")

# 选中了那个?
action = menu.exec_(self.table_widget2.mapToGlobal(pos))

if action == item_copy:
    # 赋值当前型号 B08166SLDF
    clipboard = QApplication.clipboard()
    clipboard.setText(selected_item_list[0].text())

if action == item_delete:
    # 2.翻转
    selected_item_list.reverse()

    # 3.删除
    for row_object in selected_item_list:
```

```
                index = row_object.row()
                self.table_widget2.removeRow(index)

        if action == item_reinit:
            row_count = self.table_widget2.rowCount()  # 当前表格有多少行

            if not row_count:
                QMessageBox.warning(self, "ERROR", "no available rows!")
                return

            for rowNum in range(0, row_count)[::-1]:  # 逆序删除，正序删除会有一
些删除不成功
                self.table_widget2.removeRow(rowNum)

        if action == item_run:
            route = self.txt_asin3.text()

            if not route:
                QMessageBox.warning(self, "ERROR", "no available route!")
                return

            row1 = self.table_widget1.rowCount()
```

```
column1 = self.table_widget1.columnCount()


if not row1:
    QMessageBox.warning(self, "ERROR", "no available route1!")
    return


with open('./res/route1.txt', 'wt') as f:
    for i in range(0, row1):
        for j in range(0, column1):
            if j != column1 - 1:
                f.write(self.table_widget1.item(i, j).text() + " ")
            else:
                f.write(self.table_widget1.item(i, j).text())
        if i != row1 - 1:
            f.write("\n")


row2 = self.table_widget2.rowCount()
column2 = self.table_widget2.columnCount()


if not row2:
    QMessageBox.warning(self, "ERROR", "no available route2!")
    return
```

```
with open('./res/route2.txt', 'wt') as f:
    for i in range(0, row2):
        for j in range(0, column2):
            if j != column2 - 1:
                f.write(self.table_widget2.item(i, j).text() + " ")
            else:
                f.write(self.table_widget2.item(i, j).text())
        if i != row2 - 1:
            f.write("\n")

with open('./res/route.txt', 'wt') as f:
    f.write(route)

os.system("RIP.exe")

with open('./res/res.txt', 'rt') as f:
    data = f.read()
    f.close()

dialog = ResultDialog(data)
dialog.setWindowModality(Qt.ApplicationModal)
```

```
            dialog.exec_()


        if action == item_sort:
            self.table_widget2.sortItems(0, Qt.AscendingOrder)


def event_run_click(self):
        route = self.txt_asin3.text()


        if not route:
            QMessageBox.warning(self, "ERROR", "no available route!")
            return


        row1 = self.table_widget1.rowCount()
        column1 = self.table_widget1.columnCount()


        if not row1:
            QMessageBox.warning(self, "ERROR", "no available route1!")
            return


        with open('./res/route1.txt', 'wt') as f:
            for i in range(0, row1):
                for j in range(0, column1):
```

```
                if j != column1 - 1:
                    f.write(self.table_widget1.item(i, j).text() + " ")
                else:
                    f.write(self.table_widget1.item(i, j).text())
            if i != row1 - 1:
                f.write("\n")

row2 = self.table_widget2.rowCount()
column2 = self.table_widget2.columnCount()

if not row2:
    QMessageBox.warning(self, "ERROR", "no available route2!")
    return

with open('./res/route2.txt', 'wt') as f:
    for i in range(0, row2):
        for j in range(0, column2):
            if j != column2 - 1:
                f.write(self.table_widget2.item(i, j).text() + " ")
            else:
                f.write(self.table_widget2.item(i, j).text())
        if i != row2 - 1:
```

```
                    f.write("\n")


        with open('./res/route.txt', 'wt') as f:
            f.write(route)


        os.system("RIP.exe")


        with open('./res/res.txt', 'rt') as f:
            data = f.read()
            f.close()


        dialog = ResultDialog(data)
        dialog.setWindowModality(Qt.ApplicationModal)
        dialog.exec_()


        self.txt_asin3.clear()



def style():
    mm = time.strftime('%m', time.localtime())
    m = int(mm)
    hh = time.strftime('%H', time.localtime())
```

```
h = int(hh)


if 3 <= m <= 5:
    season = 1
elif 6 <= m <= 8:
    season = 2
elif 9 <= m <= 11:
    season = 3
else:
    season = 4


if season == 1 or season == 3:
    if 6 <= h <= 18:
        apply_stylesheet(app, theme='light_blue.xml')
    else:
        apply_stylesheet(app, theme='dark_blue.xml')
elif season == 2:
    if 5 <= h <= 19:
        apply_stylesheet(app, theme='light_blue.xml')
    else:
        apply_stylesheet(app, theme='dark_blue.xml')
else:
```

```
        if 7 <= h <= 17:
            apply_stylesheet(app, theme='light_blue.xml')
        else:
            apply_stylesheet(app, theme='dark_blue.xml')


if __name__ == '__main__':
    app = QApplication(sys.argv)

    window = MainWindow()

    style()
    # apply_stylesheet(app, theme='dark_blue.xml')

    window.show()

    sys.exit(app.exec_())
```

1.2 运行结果窗口代码

```
from PyQt5.QtWidgets import QVBoxLayout, QDialog, QTextEdit
```

```
class ResultDialog(QDialog):
    def __init__(self, data, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.field_dict = {}
        self.data = data
        # print(self.data)

        self.init_ui()

    def init_ui(self):
        """
        初始化对话框
        :return:
        """
        self.setWindowTitle("result")
        self.resize(200, 200)
        layout = QVBoxLayout()
        text_edit = QTextEdit()
        text_edit.setText("")

        layout.addWidget(text_edit)
        self.setLayout(layout)
```

```
            text_edit.setText(self.data)
```

2．后端程序代码

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <algorithm>

using namespace std;



//定义路由表结构 RTable
typedef struct node{
    char dstNet[5];
    int   distance;
    char nextSkip[5];
}RTable;

RTable RT1[1000];//当前路由器路由表
RTable RT2[1000];//相邻路由器的路由表
```

```
int i,l1,l2;
char nearR1[5],nearR2[5];

bool cmp(node a, node b)
{
    return strcmp(a.dstNet, b.dstNet) < 0;
}

//当前路由初始化
void InitRTable( RTable* RT ){
    // char message
    //读  route1.txt
    FILE *ft = fopen("./res/route1.txt", "r");
    // fscanf(ft, "%s", message);

    for(i = 0; !feof(ft); ++ i)
        fscanf(ft, "%s%d%s",RT[i].dstNet,&RT[i].distance,RT[i].nextSkip);

    fclose(ft);

    l1 = i;//记录 RT1[]的长度
```

```
}

//添加路由信息
void AddNearRouter(){

    //读  route.txt
    FILE *ft = fopen("./res/route.txt", "r");
    fscanf(ft, "%s", nearR1);



}

//相邻路由初始化
void InitNearRTable(){

    //读  route2.txt
    FILE *ft = fopen("./res/route2.txt", "r");
    // fscanf(ft, "%s %d %s", RT[i].dstNet, &RT[i].distance, RT[i].nextSkip);

    for(i = 0; !feof(ft); ++ i)
    {
        fscanf(ft, "%s%d\n",RT2[i].dstNet, &RT2[i].distance);
```

```
    }
    fclose(ft);

    l2 = i;//记录 RT2[]的长度

}

//路由信息修改
void UpdateNearRTable ( RTable* RT2,char* nearR ){
    int p;
    for( p=0;p<l2;++p ){
    RT2[p].distance = RT2[p].distance + 1;
    strcpy( RT2[p].nextSkip,nearR );
    }
}


//路由表更新
void UpdateRTable( RTable* RT1,RTable* RT2 ){
    int p,q;//p——RT2[]，q——RT1[]
    for( p=0;p<l2;++p ){
        int finded=0;
```

```
        for( q=0;q<l1;++q ){
            if( strcmp( RT2[p].dstNet,RT1[q].dstNet )==0 ){//当前表中找到与发来的表
目的网络相同的一条路由信息
                finded = 1;
                if( strcmp( RT1[q].nextSkip,RT2[q].nextSkip )==0 ){//下一跳路由器正好
是这个相邻路由器
                    RT1[q].distance = RT2[p].distance;
                }
                else{//下一跳路由器不是这个
                    if( RT2[p].distance+1<RT1[q].distance ){
                        RT1[q].distance = RT2[p].distance;
                        strcpy( RT1[q].nextSkip,RT2[q].nextSkip );
                    }
                }
            }
        }
        if( !finded ){//当前表中没有这条路由信息，就加上
            strcpy( RT1[l1].dstNet,RT2[p].dstNet );
            RT1[l1].distance = RT2[p].distance;
            strcpy( RT1[l1].nextSkip,RT2[p].nextSkip );

            ++l1;
```

```
        }
    }
}

//打印
void PrintRTable( RTable* RT,int len ){
    FILE *ft1=fopen("./res/res.txt","wb");

    for( i=0;i<len;++i ){
        fprintf(ft1,"%s %d %s\n", RT[i].dstNet, RT[i].distance, RT[i].nextSkip);
    }

    fclose(ft1);

    FILE *ft2=fopen("./res/record.txt","ab");

    for( i=0;i<len;++i ){
        fprintf(ft2,"%s %d %s\n", RT[i].dstNet, RT[i].distance, RT[i].nextSkip);
    }
    fprintf(ft2, "\n");
    fclose(ft2);
```

```
}

// 打印更新的路由表
void Print_Update(){

    FILE *ft=fopen("./res/res.txt","wb");

    UpdateRTable(RT1,RT2);
    sort(RT1, RT1 + l1, cmp);
    PrintRTable(RT1,l1);

    fclose(ft);
}

int main()
{

    //初始化当前路由器
    InitRTable(RT1);

    //添加相邻路由器
```

```
        AddNearRouter();

        //初始化相邻路由器
        InitNearRTable();

        //打印修改后的路由信息
        UpdateNearRTable(RT2,nearR1);

        //进行路由表更新
        Print_Update();
    }
```