OS-PageRequest

Second assignment of OS course, implementing the page request algorithm.

一、系统概述

1.1需求分析

请求调页存储管理方式模拟

内容

> 假设每个页面可存放10条指令，分配给一个作业的内存块为4。模拟一个作业的执行过程，该作业有320条指令，即它的地址空间为32页，目前所有页还没有调入内存。

模拟过程

> 在模拟过程中，如果所访问指令在内存中，则显示其物理地址，并转到下一条指令；如果没有在内存中，则发生缺页，此时需要记录缺页次数，并将其调入内存。如果4个内存块中已装入作业，则需进行页面置换。
> 所有320条指令执行完成后，计算并显示作业执行过程中发生的缺页率。
> 置换算法可以选用FIFO或者LRU算法

1.2系统功能

1.2.1 页式请求

> 设置了一个序号为0-319的指令集，每次按照一定顺序执行指令，若该页在页表中记录，则无需置换。若页表中不存在该页，则存储器中调入对应页的内容，并更新页表的索引。如需置换，置换采用LRU算法。

1.3开发工具

> PyQt 进行开发
>
> PyCharm作为开发工具
>
> 运行在Windows系统上。

二、代码设计

2.1窗体类

```
class MemoryScene(QWidget):
    def __init__(self):
        super().__init__()
        self.paras=dict()
        self.paras["分页"]={"x":200,"y":50,"w":60,"h":40}
        self.paras["页表"]={"x":900,"y":50}
        self.paras["内存"]={"x":1600,"y":50}
        self.paras["LRU"]={"x":1400,"y":700}
        self.paras["缺页"]={"x":1500,"y":900}
        self.paras["指令条数"]=320
        self.instructions=[i for i in  range(320)]

        self.notice=QLabel(self)
        self.notice.setText("缺页率:    ")
        self.notice.move(self.paras["缺页"]["x"], self.paras["缺页"]["y"])

        self.PageChart=[]

        self.PhysicsBlock=[]
        self.LostNums=0
        self.TotalNums=0
        for i in range(32):
            self.PageChart.append({"memory":-1,"unused":0})
        for i in range(4):
            self.PhysicsBlock.append(-1)


        self.PageTable = QTableWidget(self)
        PageTitle=["物理地址"]
        self.PageTable.setColumnCount(1)
        self.PageTable.setRowCount(32)
        self.PageTable.setHorizontalHeaderLabels(PageTitle)
        # self.PageTable.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        # self.PageTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        vlabel=[]
        for i in range(self.PageTable.rowCount()):
            self.PageTable.setRowHeight(i, 36)  # 设置i行的高度
            vlabel.append(str(i))
        for k in range(self.PageTable.columnCount()):
             self.PageTable.setColumnWidth(k, 180)
        self.PageTable.setVerticalHeaderLabels(vlabel)
        #self.PageTable.verticalHeader().setVisible(False)  # 隐藏垂直表头
        self.PageTable.setGeometry(self.paras["页表"]["x"] - 40, 120, 230, 1220)

        self.PhysicsMemory=QTableWidget(self)
        PhysicsTitle=["指令"]
        self.PhysicsMemory.setColumnCount(1)
        self.PhysicsMemory.setRowCount(4)
        self.PhysicsMemory.setHorizontalHeaderLabels(PhysicsTitle)
        self.PhysicsMemory.setVerticalHeaderLabels(["0","10","20","30"])
        for i in range(self.PageTable.rowCount()):
            self.PhysicsMemory.setRowHeight(i, 100)  # 设置i行的高度
        for k in range(self.PageTable.columnCount()):
```

```
            self.PhysicsMemory.setColumnWidth(k, 200)
        self.PhysicsMemory.setGeometry(self.paras["内存"]["x"] - 40, 120, 240, 460)

        self.initUi()
```

窗体类继承于QWidget，self.paras记录窗体的尺寸等参数，PageChart是页表，PageTable是对应的表格控件，PhyiscsBlock是物理内存，此外LostNums和TotalNums记录了缺页次数和总的页面请求次数。

2.2 请求调页存储管理方式模拟

2.2.1发起调页请求

```
    def requestStrategy(self,span):
        if self.TotalNums+span>320:
            return
        for i in range(span):
            piece=random.randint(0,len(self.instructions)-2)
            self.solveLogics(self.instructions[piece])
            self.solveLogics(self.instructions[piece+1])
            self.instructions.pop(piece+1)
            self.instructions.pop(piece)
```

系统每次请求是随机请求指令队列中的一条指令和其后的一条，这样可以保证指令序列既有均匀执行又有顺序执行，使得缺页率计算更加客观准确。

2.2.2 调页策略

```
    def solveLogics(self,instruction):
        #获得页数
        self.TotalNums=self.TotalNums+1
        page=int(instruction/10)
        for i in range(len(self.PageChart)):
            if self.PageChart[i]["memory"]!=-1 and self.PageChart[i]["memory"]!=page:
                self.PageChart[i]["unused"]=self.PageChart[i]["unused"]+1

        if self.PageChart[page]["memory"]==-1:
            self.LostNums=self.LostNums+1
            flag=0
            max_unused,anchor=-1,-1
            for i in range(len(self.PhysicsBlock)):
                if self.PhysicsBlock[i]==-1:
                    #把该页存入内存块
                    self.PhysicsBlock[i]=page
                    self.PageChart[page]={"memory":i,"unused":0}
                    flag=1
                    break
                else:#记录未使用次数最多的
                    if self.PageChart[self.PhysicsBlock[i]]["unused"]>=max_unused:
                        max_unused=self.PageChart[self.PhysicsBlock[i]]["unused"]
                        #要被替换的内存块
                        anchor=i
            #若内存块没有空余
```

```
            if flag==0:
                self.PageChart[page]={"memory":anchor,"unused":0}
                self.PageChart[self.PhysicsBlock[anchor]]={"memory":-1,"unused":0}
                self.PhysicsBlock[anchor] = page
        elif self.PageChart[page]!=-1:
            self.PageChart[page]["unused"]=0
        text="缺页率"+str(self.LostNums / self.TotalNums)
        self.notice.setText(text)
        print(self.LostNums / self.TotalNums)
        self.update()
```

先将请求次数加1，然后对于违背选中的页表中项，使其未使用次数加1，如果页表中没有该页对应的内存块，即没有分配内存，则在内存块中寻找空位，如果没有空位，则采用LRU替换算法寻找内存块并更新页表。如果有该页对应的内存块，则把该页未使用置0.

2.2.3 绘图逻辑

```
    def paintEvent(self,e):
        qp=QPainter()
        qp.begin(self)
        self.drawPages(qp)

    def drawPages(self,qp):
        col=QColor(0,0,0)
        col.setNamedColor("black")
        qp.setBrush(QColor(255,215,0,0))
        for i in range(32):
            qp.drawRect(self.paras["分页"]["x"]+40, self.paras["分页"]["y"]+50 +
i*self.paras["分页"]["h"], self.paras["分页"]["w"], self.paras["分页"]["h"])
            qp.drawText(self.paras["分页"]["x"]+3,self.paras["分页"]["y"]+57 +
(i+1)*self.paras["分页"]["h"],str((i+1)*10))

        col=QColor(0,0,0)
        col.setNamedColor("black")
        qp.setBrush(QColor(255,215,0,100))

        for p in range(len(self.instructions)):
            qp.drawRect(self.paras["分页"]["x"]+40,self.paras["分页"]
["y"]+50+self.instructions[p]*self.paras["分页"]["h"]*32/320,self.paras["分页"]
["w"],self.paras["分页"]["h"]*32/320)




        for k in range(len(self.PageChart)):
            phy_address=QTableWidgetItem(str(self.PageChart[k]["memory"])+" 未用次
数"+str(self.PageChart[k]["unused"]))
            self.PageTable.setItem(k,0,phy_address)
        for t in range(len(self.PhysicsBlock)):
            phy_content=QTableWidgetItem(str(self.PhysicsBlock[t]))
            self.PhysicsMemory.setItem(t,0,phy_content)
```

根据页面参数，先画出初始内存，然后根据PageChart中对应的项的情况，更新页表和物理内存的标签，每次发生变化就调用Update重绘。

2.3 函数和类

MemoryScene类

内存块的抽象

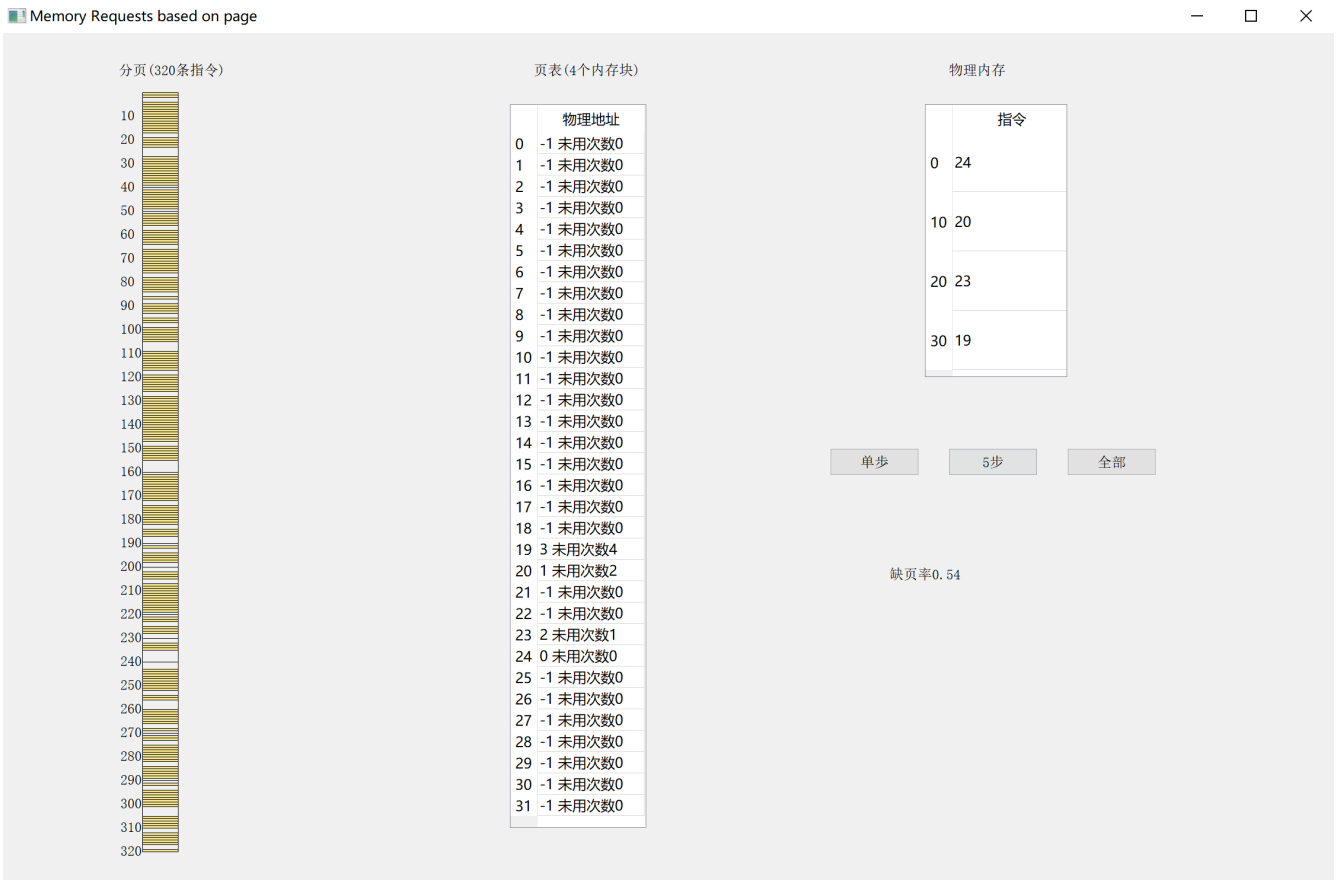
函数名	作用
setBasicElements	设置页面基本元素
requestStrategy	表示对页面的请求
solveLogics	响应请求调页

成员变量	作用
paras	保存参数
instructions	指令池
notice	缺页率提示标签
PhysicsBlocks	物理内存表
PageChart	页表
PageTable	页表控件

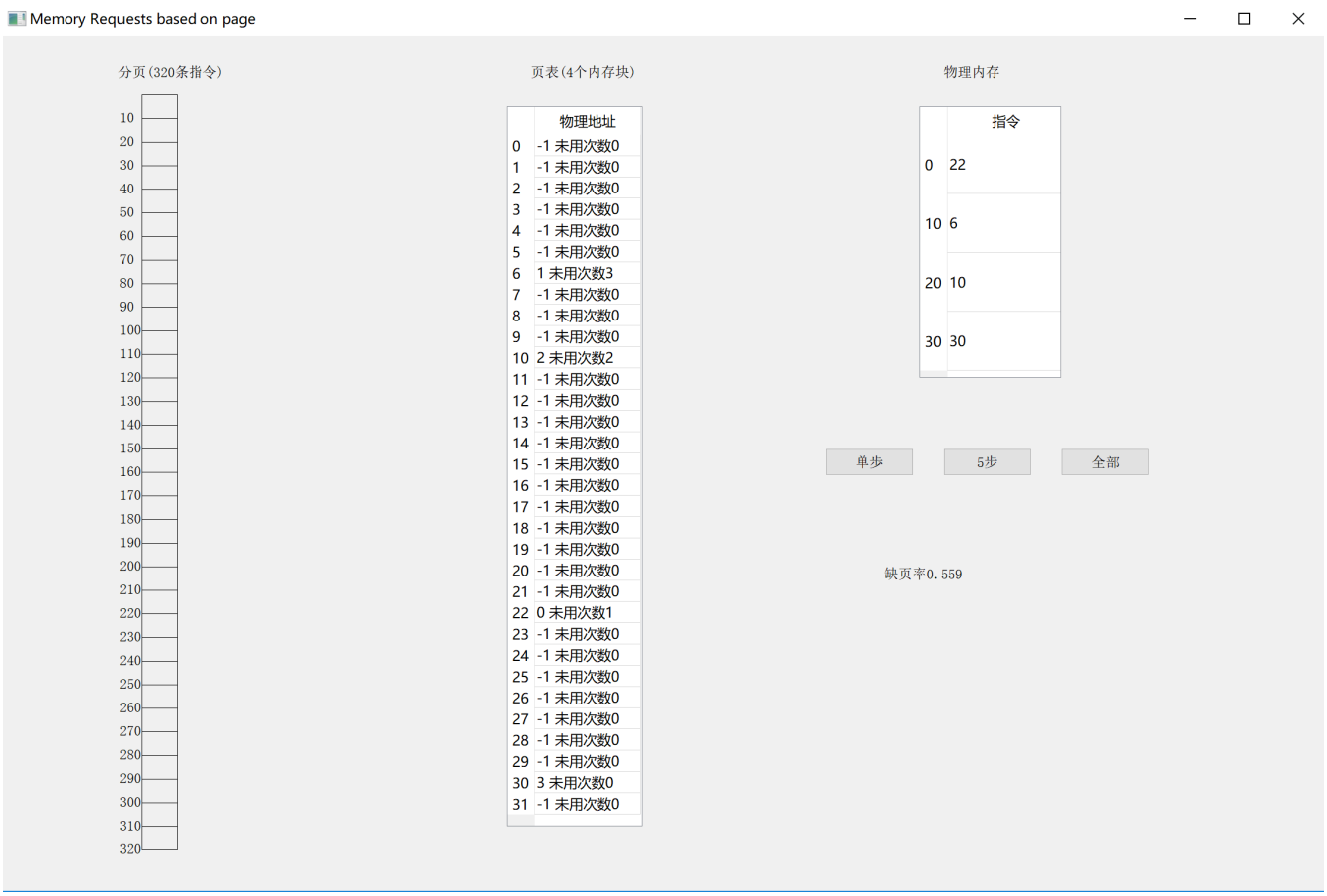
三、运行效果

初始页面，左边表示指令，右边有执行选项

有单步，五步和全部权限，被调走的指令颜色设置为透明表示不存在



调页请求结束



四、分析

对于操作系统有了更深的理解，用python模拟请求调页，程序不需要全部装入内存，也就是模拟所有的内存资源如何分配给进程，进程应该被分配到内存哪一块，理解了缺页率等概念，实现了LRU和FIFO算法 通过本程序，我了解PyQt的使用，并加深了对Python面向对象的理解。 当然这个程序还是有些地方写得不够好的,设计时思路有一些混乱,控件没有集合到一起，导致主窗口元素过多。