

江蘇大學京江學院

JIANGSU UNIVERSITY JINGJIANG COLLEGE



操作系统实验报告

题 目：	实验三 请求式分页调度
上机时间：	2023.11.30
授课教师：	潘雨青
姓 名：	马云骥
学 号：	4211153047
专 业：	软件工程
班 级：	J软件(嵌入)(专转本)2102
日 期：	2023.12.01

实验三 请求式分页调度

1 实验目的

1. 熟悉请求式分页调度算法，理解缺页中断的机理
2. 能够仿真实现请求式分页调度算法

2 实验设计

原始参考的案例代码均由  C# 编写，这里本人使用  Python 实现，力求达到同样的效果。

开发环境:  Windows 11 Pro 23H2

 Python 3.12

 PyCharm 2023.2.5

实验指导书实验内容

1. 运行实例中的 Paging 项目，了解其程序结构，并理解其中的请求式分页调度是采用了何种页面淘汰算法，并在实验报告中回答。
2. 在实例中的 Paging 项目实现按 LRU 页面淘汰算法。

先回答指导书中的问题，原实例模拟了 FIFO（先进先出）的页面淘汰算法，最早进入内存的页面会被最先淘汰。

下面是我的 Python 实现：

```
1  import datetime
2  import random
3
4  # 页面表类定义
5  class PageTable:
6      def __init__(self, page_no, block_no, hd_addr):
7          self.page_no = page_no # 页面号
8          self.block_no = block_no # 块号
9          self.is_in = False # 页面是否在内存中
10         self.is_modified = False # 页面是否被修改
11         self.hd_addr = hd_addr # 硬盘地址
12         self.last_accessed = None # 最后访问时间
13
14     def read(self):
15         self.is_in = True # 设置页面为在内存中
16         self.last_accessed = datetime.datetime.now() # 更新最后访问时间
17         print(f"Page {self.page_no} in {self.block_no} block is read")
18
19     def write(self):
```

```

20         self.is_in = True # 设置页面为在内存中
21         self.is_modified = True # 标记页面已修改
22         self.last_accessed = datetime.datetime.now() # 更新最后访问时间
23         print(f"Page {self.page_no} in {self.block_no} block is write")
24
25     def display(self):
26         # 格式化最后访问时间
27         last_accessed_str = self.last_accessed.strftime("%Y/%m/%d %H:%M:%S") if
self.last_accessed else "None"
28         print(
29             f"Page {self.page_no}, isIn {self.is_in}, isModified {self.is_modified},
BlockNo {self.block_no}, HdAddr {self.hd_addr}, LastAccessed {last_accessed_str}")
30
31 # 块列表类定义
32 class BlockList:
33     def __init__(self, capacity, scheduling_algo):
34         self.plist = [] # 页面列表
35         self.capacity = capacity # 内存容量
36         self.scheduling_algo = scheduling_algo # 页面置换算法
37
38     def push(self, page):
39         if page not in self.plist:
40             # 检查待添加的页面是否已在内存中。如果不在，则继续。
41             if len(self.plist) == self.capacity:
42                 # 如果内存已满（页面列表的长度等于内存容量），则需要进行页面置换。
43                 removed_page = None # 初始化要移除的页面为 None
44
45                 if self.scheduling_algo == "FIFO":
46                     # 如果采用的是先进先出（FIFO）算法：
47                     removed_page = self.plist.pop(0) # 移除列表中的第一个页面（最早进入内存
存的页面）
48
49                 elif self.scheduling_algo == "LRU":
50                     # 如果采用的是最近最少使用（LRU）算法：
51                     self.plist.sort(key=lambda p: p.last_accessed) # 根据每个页面的最后访问
时间进行排序
52                     removed_page = self.plist.pop(0) # 移除列表中最少使用的页面（即排序后
的第一个页面）
53
54                     # 将被移除的页面的 is_in 设置为 false
55                     if removed_page:
56                         removed_page.is_in = False
57
58                     self.plist.append(page)
59                     print("发生缺页中断")
60                     page.is_in = True
61                     self.display_memory()
62
63     def display_memory(self):
64         print("在内存的页面为：", end="")
65         for p in self.plist:
66             print(f"PageNo: {p.page_no} | ", end="")
67         print()
68
69 # 模拟分页的函数
70 def simulate_paging():

```

```

71     # 初始化分页系统
72     print("选择要模拟的页面置换算法")
73     print("1: FIFO 先进先出")
74     print("2: LRU 最近最少使用")
75     choice = input("请输入你的选择: ")
76     algorithms = {"1": "FIFO", "2": "LRU"}
77     if choice not in algorithms:
78         print("无效的输入! ")
79         return
80     block_list = BlockList(5, algorithms[choice])
81     pages = [PageTable(i + 1, i * 10 + 1, f"Hd{i * 1000}") for i in range(20)]
82
83     # 模拟循环
84     while True:
85         page_no = random.randint(1, 20)
86         page = pages[page_no - 1]
87         read_or_write = random.choice(["read", "write"])
88
89         if read_or_write == "read":
90             page.read()
91         else:
92             page.write()
93
94         block_list.push(page)
95
96         for p in pages:
97             p.display()
98
99         print("-----")
100        input("按回车模拟下一步页面操作...")
101
102    simulate_paging()
103

```

我将本次实验需要模拟的两种页面置换算法实现在了一个 Python 程序中，程序运行后可以根据需要，选择使用对应的算法进行模拟。它包含两个主要类（PageTable 和 BlockList）和一个模拟函数（simulate_paging）。下面是每个部分的详细解释：

2.1 PageTable 类

PageTable 类定义了页面的基本属性和方法。每个页面对象包含以下属性：

- `page_no`：页面号。
- `block_no`：该页面所在的内存块号。
- `is_in`：标识页面是否在内存中。
- `is_modified`：标识页面自上次加载后是否被修改。
- `hd_addr`：页面在硬盘上的地址。
- `last_accessed`：页面最后访问时间。

此外，该类包含以下方法：

- `read()`：模拟读取页面，更新页面状态和最后访问时间。

- `write()`：模拟写入页面，更新页面状态、修改标识和最后访问时间。
- `display()`：打印页面的当前状态。

2.2 `BlockList` 类

`BlockList` 类模拟物理内存中的块列表，负责管理页面的加载和置换。其主要属性包括：

- `plist`：当前在内存中的页面列表。
- `capacity`：内存的容量（可容纳页面的数量）。
- `scheduling_algo`：使用的页面置换算法。

主要方法包括：

- `push(page)`：处理新页面请求。如果内存已满，根据置换算法选择并移除一个页面，然后将新页面加入内存。
- `display_memory()`：显示当前内存中的页面列表。

2.3 `simulate_paging` 函数

这是模拟的核心函数，执行以下步骤：

1. 用户选择页面置换算法（FIFO 或 LRU）。
2. 初始化 `BlockList` 对象和一系列 `PageTable` 对象，代表内存块和一组页面。
3. 进入模拟循环，随机生成页面读取或写入请求。
4. 根据页面请求，调用相应的 `PageTable` 方法（读或写）更新页面状态。
5. 调用 `BlockList` 的 `push` 方法处理页面加载和置换。
6. 循环展示每个页面的状态和整个内存的状态。

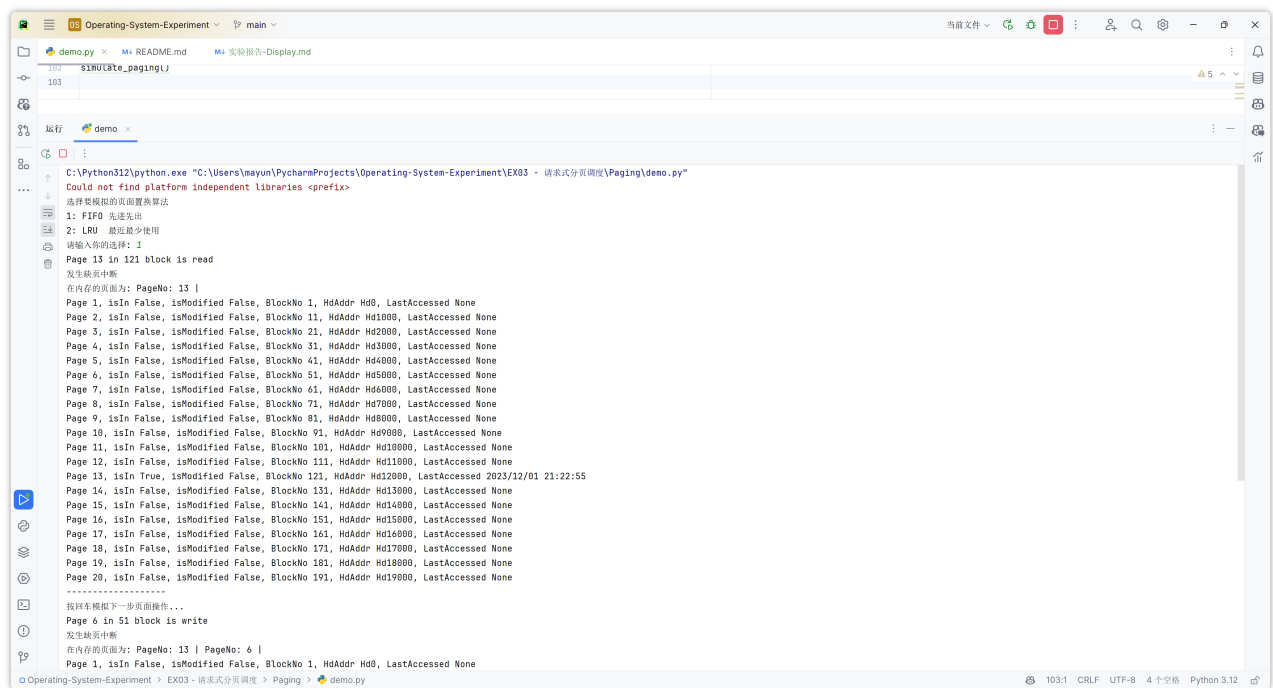
2.4 实验过程

在实验过程中，程序会持续生成随机的页面读写请求，模拟内存中页面的动态变化。用户可以通过观察每一步的输出，理解不同置换算法下，内存中页面组合是如何随着时间和页面请求而变化的。例如，在 FIFO 算法下，最先进入内存的页面会首先被置换；而在 LRU 算法下，则是最久未被访问的页面首先被置换。

这个过程有助于理解操作系统中虚拟内存管理的复杂性和页面置换算法的效果及其局限性。

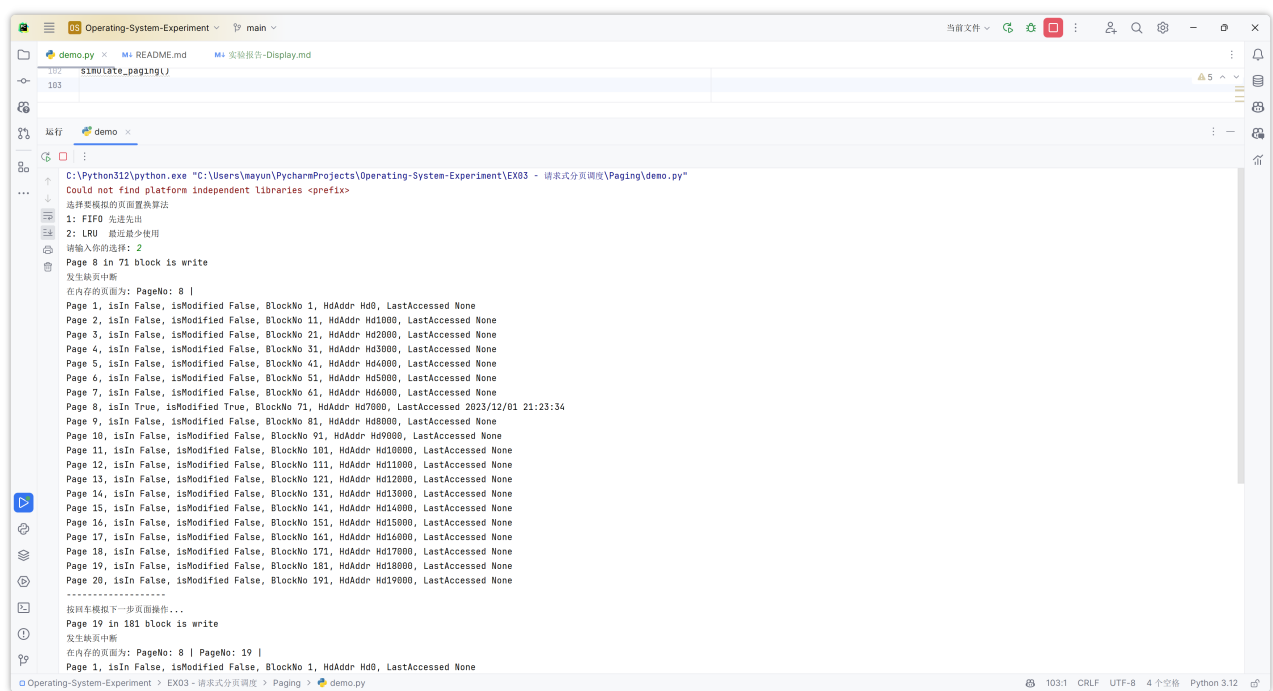
3 实验结果

3.1 FIFO（先进先出）



```
C:\Python312\python.exe "C:\Users\mayun\PycharmProjects\Operating-System-Experiment\EX03 - 请求式分页调度\Paging\demo.py"
Could not find platform independent Libraries <prefix>
选择要模拟的页面置换算法
1: FIFO 先进先出
2: LRU 最近最少使用
请输入你的选择: 1
Page 13 in 121 block is read
发生缺页中断
在内存的页面为: PageNo: 13 |
Page 1, isIn False, isModified False, BlockNo 1, HdAddr Hd0, LastAccessed None
Page 2, isIn False, isModified False, BlockNo 11, HdAddr Hd1000, LastAccessed None
Page 3, isIn False, isModified False, BlockNo 21, HdAddr Hd2000, LastAccessed None
Page 4, isIn False, isModified False, BlockNo 31, HdAddr Hd3000, LastAccessed None
Page 5, isIn False, isModified False, BlockNo 41, HdAddr Hd4000, LastAccessed None
Page 6, isIn False, isModified False, BlockNo 51, HdAddr Hd5000, LastAccessed None
Page 7, isIn False, isModified False, BlockNo 61, HdAddr Hd6000, LastAccessed None
Page 8, isIn False, isModified False, BlockNo 71, HdAddr Hd7000, LastAccessed None
Page 9, isIn False, isModified False, BlockNo 81, HdAddr Hd8000, LastAccessed None
Page 10, isIn False, isModified False, BlockNo 91, HdAddr Hd9000, LastAccessed None
Page 11, isIn False, isModified False, BlockNo 101, HdAddr Hd10000, LastAccessed None
Page 12, isIn False, isModified False, BlockNo 111, HdAddr Hd11000, LastAccessed None
Page 13, isIn True, isModified False, BlockNo 121, HdAddr Hd12000, LastAccessed 2023/12/01 21:22:55
Page 14, isIn False, isModified False, BlockNo 131, HdAddr Hd13000, LastAccessed None
Page 15, isIn False, isModified False, BlockNo 141, HdAddr Hd14000, LastAccessed None
Page 16, isIn False, isModified False, BlockNo 151, HdAddr Hd15000, LastAccessed None
Page 17, isIn False, isModified False, BlockNo 161, HdAddr Hd16000, LastAccessed None
Page 18, isIn False, isModified False, BlockNo 171, HdAddr Hd17000, LastAccessed None
Page 19, isIn False, isModified False, BlockNo 181, HdAddr Hd18000, LastAccessed None
Page 20, isIn False, isModified False, BlockNo 191, HdAddr Hd19000, LastAccessed None
-----
按回车模拟下一步页面操作...
Page 6 in 51 block is write
发生缺页中断
在内存的页面为: PageNo: 13 | PageNo: 6 |
Page 1, isIn False, isModified False, BlockNo 1, HdAddr Hd0, LastAccessed None
```

3.2 LRU（最近最少使用）



```
C:\Python312\python.exe "C:\Users\mayun\PycharmProjects\Operating-System-Experiment\EX03 - 请求式分页调度\Paging\demo.py"
Could not find platform independent Libraries <prefix>
选择要模拟的页面置换算法
1: FIFO 先进先出
2: LRU 最近最少使用
请输入你的选择: 2
Page 8 in 71 block is write
发生缺页中断
在内存的页面为: PageNo: 8 |
Page 1, isIn False, isModified False, BlockNo 1, HdAddr Hd0, LastAccessed None
Page 2, isIn False, isModified False, BlockNo 11, HdAddr Hd1000, LastAccessed None
Page 3, isIn False, isModified False, BlockNo 21, HdAddr Hd2000, LastAccessed None
Page 4, isIn False, isModified False, BlockNo 31, HdAddr Hd3000, LastAccessed None
Page 5, isIn False, isModified False, BlockNo 41, HdAddr Hd4000, LastAccessed None
Page 6, isIn False, isModified False, BlockNo 51, HdAddr Hd5000, LastAccessed None
Page 7, isIn False, isModified False, BlockNo 61, HdAddr Hd6000, LastAccessed None
Page 8, isIn True, isModified True, BlockNo 71, HdAddr Hd7000, LastAccessed 2023/12/01 21:23:34
Page 9, isIn False, isModified False, BlockNo 81, HdAddr Hd8000, LastAccessed None
Page 10, isIn False, isModified False, BlockNo 91, HdAddr Hd9000, LastAccessed None
Page 11, isIn False, isModified False, BlockNo 101, HdAddr Hd10000, LastAccessed None
Page 12, isIn False, isModified False, BlockNo 111, HdAddr Hd11000, LastAccessed None
Page 13, isIn False, isModified False, BlockNo 121, HdAddr Hd12000, LastAccessed None
Page 14, isIn False, isModified False, BlockNo 131, HdAddr Hd13000, LastAccessed None
Page 15, isIn False, isModified False, BlockNo 141, HdAddr Hd14000, LastAccessed None
Page 16, isIn False, isModified False, BlockNo 151, HdAddr Hd15000, LastAccessed None
Page 17, isIn False, isModified False, BlockNo 161, HdAddr Hd16000, LastAccessed None
Page 18, isIn False, isModified False, BlockNo 171, HdAddr Hd17000, LastAccessed None
Page 19, isIn False, isModified False, BlockNo 181, HdAddr Hd18000, LastAccessed None
Page 20, isIn False, isModified False, BlockNo 191, HdAddr Hd19000, LastAccessed None
-----
按回车模拟下一步页面操作...
Page 19 in 181 block is write
发生缺页中断
在内存的页面为: PageNo: 8 | PageNo: 19 |
Page 1, isIn False, isModified False, BlockNo 1, HdAddr Hd0, LastAccessed None
```

4 实验总结

通过本实验，可以观察到不同页面置换算法对系统性能的影响。FIFO 算法简单高效，但可能不总是最优的选择，特别是在频繁访问的页面被置换时。LRU 算法更加智能，能够有效地利用历史访问信息来预测未来的页面访问模式，但其实现相对复杂。在实际应用中，需要根据系统的具体情况，选择合适的页面置换算法。