

目 录

| | |
|-----------------------|----|
| 1. 模拟进程调度算法的实现..... | 1 |
| 1.1 需求规格说明..... | 1 |
| 1.1.1 问题描述..... | 1 |
| 1.1.2 问题分析..... | 1 |
| 1.2 算法设计..... | 1 |
| 1.2.1 设计思想..... | 1 |
| 1.2.2 设计表示..... | 2 |
| 1.3 调试报告..... | 3 |
| 1.4 附录..... | 3 |
| 2. 模拟存储器管理的设计与实现..... | 7 |
| 2.1 需求规格说明..... | 7 |
| 2.1.1 问题描述..... | 7 |
| 2.1.2 问题分析..... | 7 |
| 2.2 算法设计..... | 7 |
| 2.2.1 设计思想..... | 7 |
| 2.2.2 设计表示..... | 8 |
| 2.3 调试报告..... | 10 |
| 2.4 附录..... | 10 |
| 3. 模拟文件系统的设计与实现..... | 15 |
| 3.1 需求规格说明..... | 15 |
| 3.1.1 问题描述..... | 15 |
| 3.1.2 问题分析..... | 15 |
| 3.2 算法设计..... | 16 |
| 3.2.1 设计思想..... | 16 |
| 3.2.2 设计表示..... | 16 |
| 3.3 调试报告..... | 17 |
| 3.4 附录..... | 18 |
| 4. 总 结..... | 21 |

1. 模拟进程调度算法的实现

1.1 需求规格说明

1.1.1 问题描述

(1) 设计内容

1) 创建进程：手动创建几个进程，或者随机创建几个进程，都在界面上完成；要求包括进程的名称（不能重复）、创建时间、执行时间等。某时刻仅一个进程在运行，需要申请的资源都能申请到。

2) 完成先来先服务、最短作业优先以及最高响应比优先调度算法。

(2) 要求

在屏幕上输出各进程不同调度算法的演示过程以及对周转时间以及平均周转时间和平均带权周转时间进行计算，并对比各算法的优劣。

1.1.2 问题分析

1.界面：要求能在界面上手动输入进程，可以在界面上设计三个输入框（EditControl 控件）分别输入进程名，到达时间和运行时间。三种调度方法可以对于三个按钮，最后结果用表格的形式显示出来（使用 ListControl 控件），用两个仅读的 EditControl 控件来输出平均周转时间和平均带权周转时间。

2.输入进程信息时，要能正确处理各种错误情况，包括但不限于：输入重名的进程，输入运行时间为 0 的进程，输入的开始时间或运行时间为负数。

1.2 算法设计

1.2.1 设计思想

1. 定义一个结构体 work 来表示一个进程。进程结构体应该包含有进程名、到达时间、运行时间、开始运行时间、结束运行时间、周转时间、带权周转时间等因素。

2. 定义一个类来表示多进程的调度。进程类应该包括：所有待调度进程的信息以及几种进程调度算法（先来先服务，短作业优先，响应比高者优先）的实现函数。

3. 上述三种调度算法的思想：

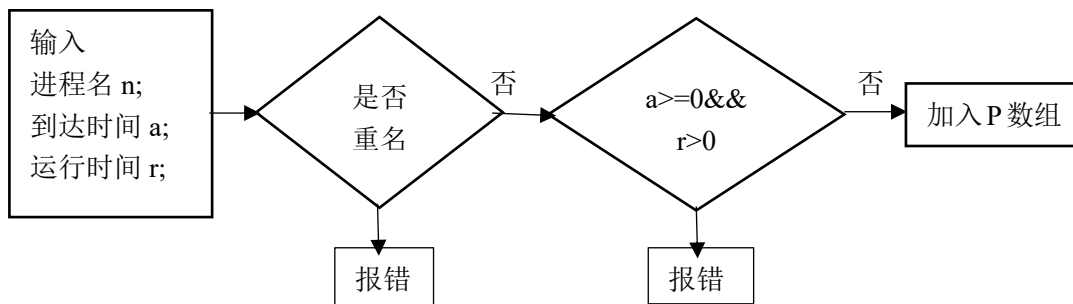
先来先服务：所有待调度进程按到达时间从小到大排序，按照排序后的顺序依次执行。若进程到达时前一进程还未结束，则该进程开始时间为前一进程的结束时间；否则该进程的开始时间即为该进程的到达时间。直至所有进程执行完毕。

短作业优先：所有待调度进程按运行时间从小到大排序，按照排序后的顺序依次执行。若轮到某一进程运行时该进程还未到达，则从就绪进程队列中查找最先到达的进程并执行该进程。直至所有进程执行完毕。

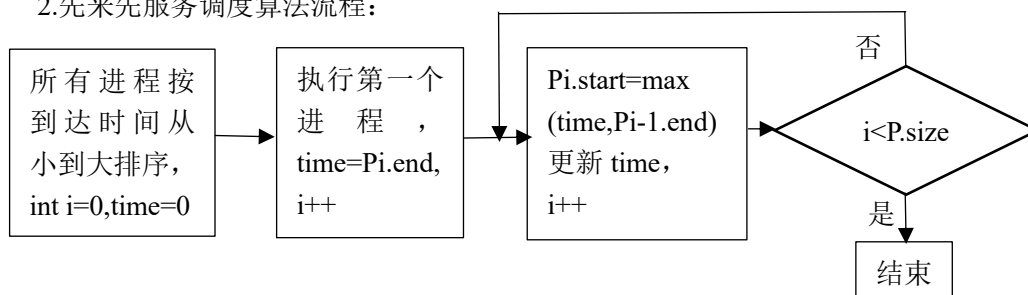
响应比高者优先：按照公式响应比 $R_p = \frac{\text{等待时间} + \text{运行时间}}{\text{运行时间}}$ 计算各进程的相应比，查找出响应比最高的进程并执行该进程，此后每执行完一个进程就重新计算一次响应比并重复上述过程，直至所有进程执行完毕。

1.2.2 设计表示

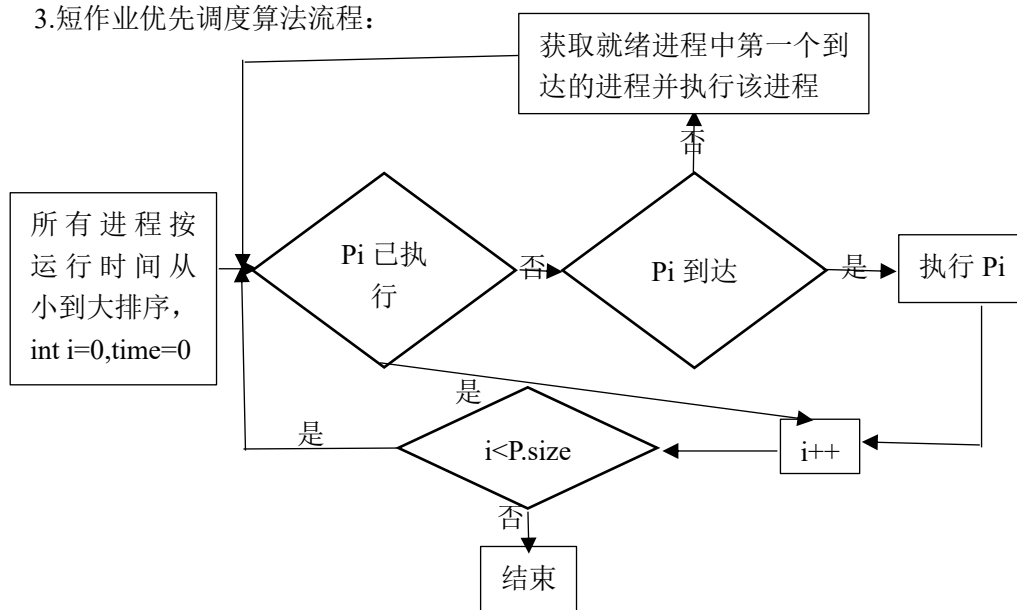
1. 添加进程：输入进程名，到达时间，运行时间以添加进程。流程如下：



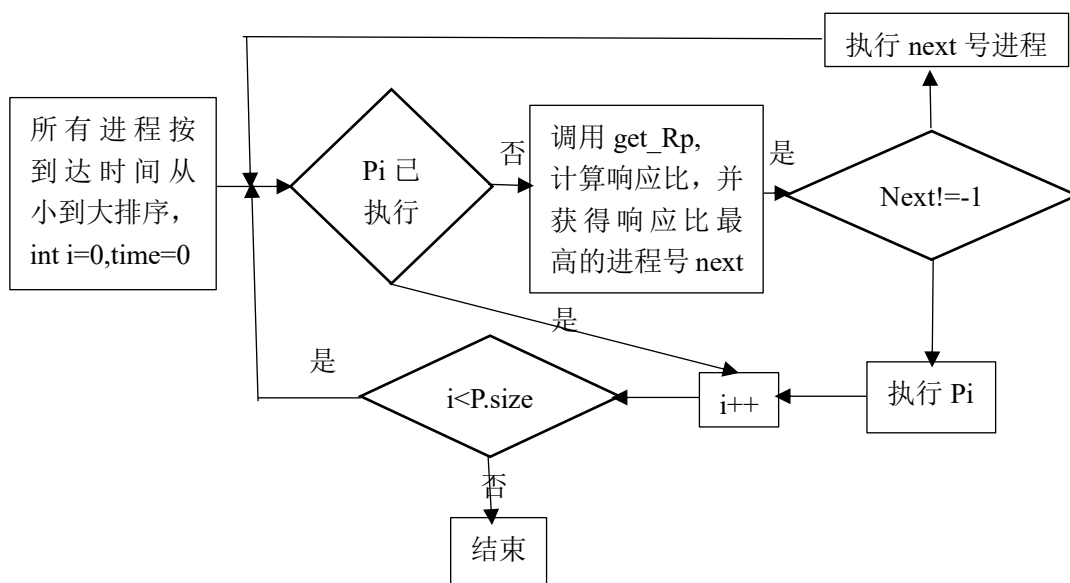
2. 先来先服务调度算法流程：



3. 短作业优先调度算法流程：



4. 响应比高者优先调度算法：



1.3 调试报告

遇到的问题：抢占式的短作业优先算法的实现以及显示问题

解决方案：未解决。

改进思想：实现抢占式的短作业优先算法，改用更直观的显示方式（类似进度条那样的时间流显示效果）。

1.4 附录

源代码清单如右图：

主要代码都位于 process.h 、 process.cpp 以及 ScheduleDlg.h 和 ScheduleDlg.cpp 中。

Work 结构体和 process 类的定义和实现都写在 process.h 和 process.cpp 中。

ScheduleDlg.h 和 ScheduleDlg.cpp 中写的代码都是与界面显示内容相关的。

其他代码文件是 MFC 对话框项目自带的。

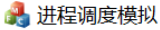
The screenshot shows a file explorer window with the following files listed:

- res
- framework.h
- pch.cpp
- pch.h
- process.cpp
- process.h
- resource.h
- Scheduler.apx
- Scheduler.cpp
- Scheduler.h
- Scheduler.rc
- Scheduler.vcxproj
- Scheduler.vcxproj.filters
- Scheduler.vcxproj.user
- ScheduleDlg.cpp
- ScheduleDlg.h
- targetver.h

测试数据：

| 进程 | A | B | C | D | E | F | G | H | I | J |
|----|---|----|----|----|----|----|---|----|----|----|
| 到达 | 0 | 2 | 5 | 7 | 12 | 15 | 4 | 6 | 8 | 10 |
| 运行 | 7 | 10 | 20 | 30 | 40 | 8 | 8 | 20 | 10 | 12 |

测试结果——先来先服务：


×

创建进程

进程名

到达时间

运行时间

确定

调度算法

先来先服务算法

短作业优先算法

高响应比优先算法

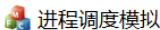
| 进程名 | 到达时间 | 运行时间 | 开始时间 | 结束时间 | 周转时间 | 带权周转时间 |
|-----|------|------|------|------|------|--------|
| A | 0 | 7 | 0 | 7 | 7 | 1.000 |
| B | 2 | 10 | 7 | 17 | 15 | 1.500 |
| G | 4 | 8 | 17 | 25 | 21 | 2.625 |
| C | 5 | 20 | 25 | 45 | 40 | 2.000 |
| H | 6 | 20 | 45 | 65 | 59 | 2.950 |
| D | 7 | 30 | 65 | 95 | 88 | 2.933 |
| I | 8 | 10 | 95 | 105 | 97 | 9.700 |
| J | 10 | 12 | 105 | 117 | 107 | 8.917 |
| E | 12 | 40 | 117 | 157 | 145 | 3.625 |
| F | 15 | 8 | 157 | 165 | 150 | 18.750 |
| | | | | | | |
| | | | | | | |
| | | | | | | |

平均周转时间

平均带权周转时间

关闭

测试结果——短作业优先：


×

创建进程

进程名

到达时间

运行时间

确定

调度算法

先来先服务算法

短作业优先算法

高响应比优先算法

| 进程名 | 到达时间 | 运行时间 | 开始时间 | 结束时间 | 周转时间 | 带权周转时间 |
|-----|------|------|------|------|------|--------|
| A | 0 | 7 | 0 | 7 | 7 | 1.000 |
| G | 4 | 8 | 7 | 15 | 11 | 1.375 |
| F | 15 | 8 | 15 | 23 | 8 | 1.000 |
| B | 2 | 10 | 23 | 33 | 31 | 3.100 |
| I | 8 | 10 | 33 | 43 | 35 | 3.500 |
| J | 10 | 12 | 43 | 55 | 45 | 3.750 |
| C | 5 | 20 | 55 | 75 | 70 | 3.500 |
| H | 6 | 20 | 75 | 95 | 89 | 4.450 |
| D | 7 | 30 | 95 | 125 | 118 | 3.933 |
| E | 12 | 40 | 125 | 165 | 153 | 3.825 |
| | | | | | | |
| | | | | | | |
| | | | | | | |

平均周转时间

平均带权周转时间

关闭

测试结果——响应比高者优先：

进程调度模拟

创建进程

进程名

到达时间

运行时间

确定

调度算法

先来先服务算法

短作业优先算法

高响应比优先算法

| 进程名 | 到达时间 | 运行时间 | 开始时间 | 结束时间 | 周转时间 | 带权周转时间 |
|-----|------|------|------|------|------|--------|
| A | 0 | 7 | 0 | 7 | 7 | 1.000 |
| B | 2 | 10 | 7 | 17 | 15 | 1.500 |
| G | 4 | 8 | 17 | 25 | 21 | 2.625 |
| I | 8 | 10 | 25 | 35 | 27 | 2.700 |
| F | 15 | 8 | 35 | 43 | 28 | 3.500 |
| J | 10 | 12 | 43 | 55 | 45 | 3.750 |
| C | 5 | 20 | 55 | 75 | 70 | 3.500 |
| H | 6 | 20 | 75 | 95 | 89 | 4.450 |
| D | 7 | 30 | 95 | 125 | 118 | 3.933 |
| E | 12 | 40 | 125 | 165 | 153 | 3.825 |

平均周转时间

57.3

平均带权周转时间

3.078

关闭

其他错误处理：

运行时间为 0：

进程调度模拟

创建进程

进程名

到达时间

运行时间

确定

调度算法

先来先服务算法

短作业优先算法

高响应比优先算法

| 进程名 | 到达时间 | 运行时间 | 开始时间 | 结束时间 | 周转时间 | 带权周... |
|-----|------|------|------|------|------|--------|
|-----|------|------|------|------|------|--------|

平均周转时间

0

平均带权周转时间

0

关闭

错误

!

进程运行时间不能为0!

确定

输入负数:

The screenshot shows the "进程调度模拟" (Process Scheduling Simulation) application window. On the left, under the "创建进程" (Create Process) section, the process name is set to "A", arrival time to "-2", and execution time to "-3". The "调度算法" (Scheduling Algorithm) section has three options: "先来先服务算法" (First-Come-First-Served), "短作业优先算法" (Shortest Job First), and "高响应比优先算法" (Highest Response Ratio First). In the center, an error dialog box titled "错误" (Error) displays a yellow warning triangle icon and the message "进程到达时间必须不能为负数!" (Process arrival time must not be negative!). A "确定" (OK) button is at the bottom right of the dialog. In the background, a table lists scheduling metrics: "进程名", "到达时间", "运行时间", "开始时间", "结束时间", "周转时间", and "带权周...". At the bottom, input fields for "平均周转时间" (Average Turnaround Time) and "平均带权周转时间" (Average Weighted Turnaround Time) both show the value "0". A "关闭" (Close) button is located at the bottom right corner.

重名：

[illegible]

2. 模拟存储器管理的设计与实现

2.1 需求规格说明

2.1.1 问题描述

(1) 设计内容

- 1) 在内存开辟两块存储空间，分别模拟内存和外存，大小分别是 8M，25M。
- 2) 给定三个程序 A，B，C，大小分别是 4.2M，8M 和 9.8M。模拟其并发执行过程。
- 3) 每个进程在内存中固定分配 4 个页面，缺页时分别采用四种置换算法（FIFO, LRU, NUR, OPT）进行置换。
- 4) 假定页面大小为 512K，进程执行时页面调度顺序要求手动输入。

(2) 要求

在屏幕上输出各进程页面置换过程（以图表形式描述）以及每种页面置换算法的缺页率和置换率，并对比各算法的优劣。

2.1.2 问题分析

1.界面设计：界面上设计三个 EditControl 控件来输入各进程的页面调度顺序。设计一个下拉框来选择页面置换算法，用一个总表（ListControl）来显示页面置换过程。

2.外存大小为 25M，每页大小为 512K，所以虚拟页最多 50 页，大于 50 的请求都不能执行。每个进程有四个页面，在页面置换过程中各进程的这 4 个页面互相不尝试影响。

2.2 算法设计

2.2.1 设计思想

1.定义两个结构体 Ppage 和 Vpage 分别代表物理页和虚拟页，物理页有页号、下一物理页指针、对应的虚拟页号、访问位、修改位等内容。虚拟页有页号、下一虚拟页指针、缺页标识等内容。

2.定义一个 process 类来表示进程。process 类应该包括有进程名、进程大小、页框（物理页）、页面请求（虚拟页）等基本信息。为了方便记录缺页率和置换率，还要再加入缺页率和置换率两个信息到 process 类。除此之外，将四种页面置换算法（FIFO, LRU, NUR, OPT）也加入到 process 类中，因为题目中要求实现三个进程的并发执行，所以就必须使用到信号量，所以在实现上述四种页面置换算法是要将信号量作为参数传入函数中。

3.定义一个 multiprocess 类来实现多个进程的并发执行（题目中为 A，B，C 三个进程）。multiprocess 就相当于模拟了总的内存。所以 multiprocess 包括有所有的虚拟页信息。

4.四种置换算法的设计思想:

FIFO: 先进先出页面置换算法,即当发生缺页中断时,优先置换掉最早进入的页面。可用一个指针来指向最早进入的页面,当该页面被置换后,指针后移一个即可。

LRU: 最近最久未使用页面置换算法,即优先置换最久未使用的页面。可以用一个数组记录下各页面上次被访问的时间,当缺页中断时,查找该数组获得时间最小的页面并将其置换。

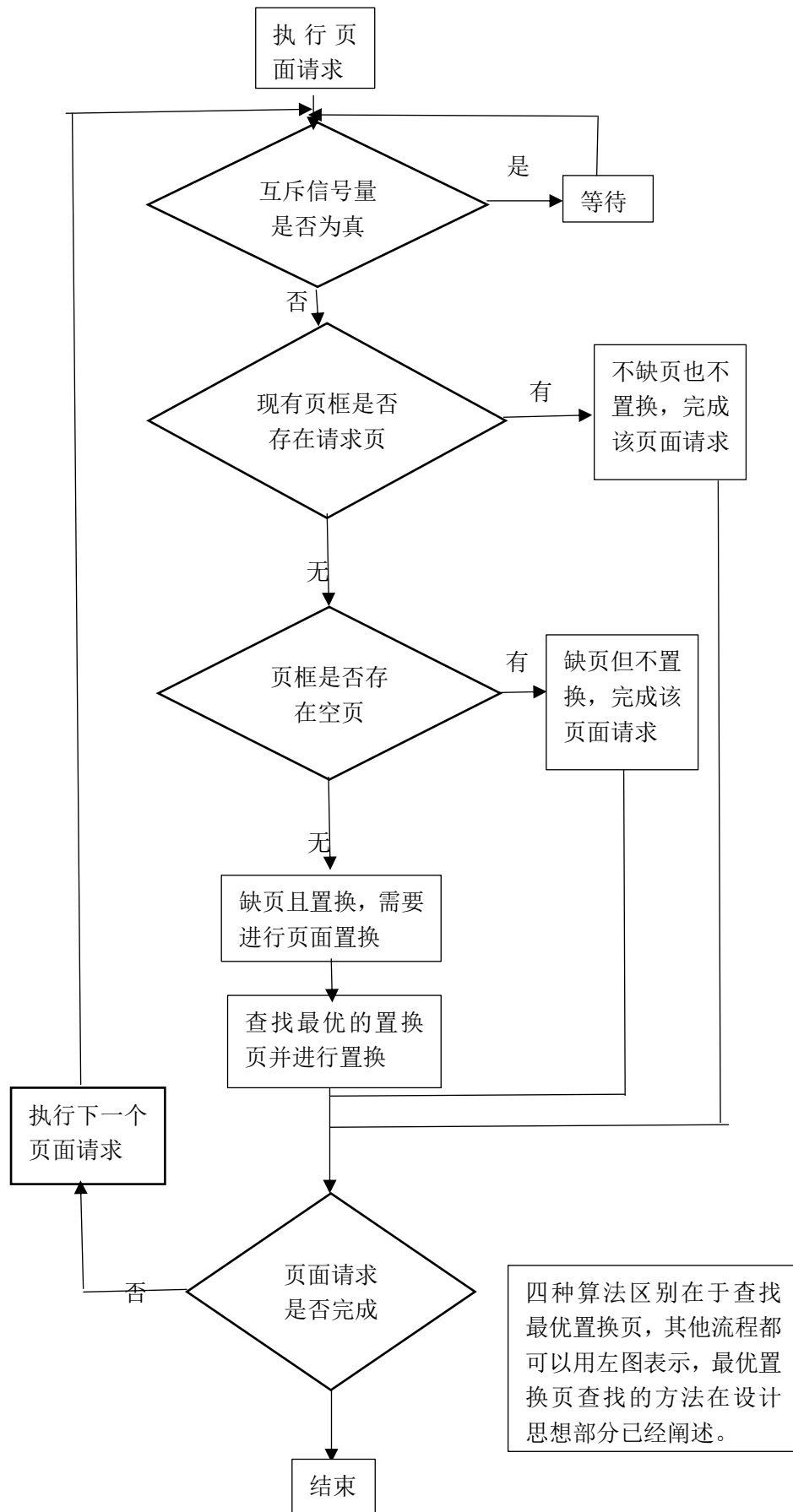
NRU: 最近未使用页面置换算法,即优先置换最近未使用的页面。这是就需要用到 **Vpage** 中的访问位 **R** 和修改位 **V**,页面被访问(修改)后就将其访问位(修改位)置为 1,然后如果一定时间内(我设计的是每进行 5 次页面访问)页面都未被访问,则将该页面访问位重新置为 0。当发生缺页时,优先选择 **R** 和 **V** 都是 0 的页进行置换,其次是 **R** 为 0, **V** 为 1。

OPT: 最优页面置换算法,即优先置换掉接下来不会被访问或者最晚被访问的页面。由于这个题我们是手动一次性输入所有的页面调度请求,因此这种方法可以实现。用一个数组保存各页面下次被访问的时间,缺页时置换出时间最长的那一页即可。

2.2.2 设计表示

上述四种页面置换算法的流程都是:首先检查页框内有没有当前页面,如果有,直接进行访问即可(不缺页也不置换);如果没有,则再检查有没有空页页框,如果存在空页,则调度当前请求页面(缺页但不置换),如果不存在空页,那就需要进行页面置换(缺页且中断)。

流程都可以用下一页的流程图来表示。



2.3 调试报告

遇到的问题 1: 各进程的置换率与缺页率返回不到页面上, 在页面上显示不出来。

解决方法: 改成数组, 用数组来传递就能传递出来。

遇到的问题 2: 互斥信号量无处可写。

解决方法: 将互斥信号量作为参数传递到页面置换算法的函数中。

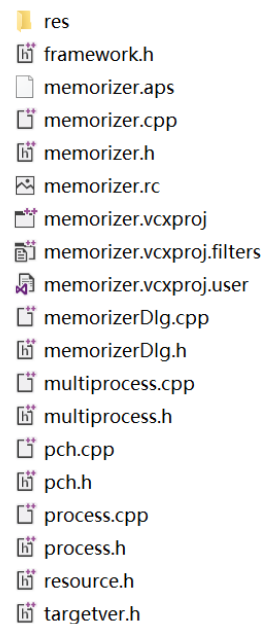
改进思想: 在界面上表格中显示页面置换过程时能够按调度顺序依次显示, 这样更能体现出并发执行的特点。现在做的是先执行页面调度, 最后一起显示在表格上。

2.4 附录

源代码清单如右图所示。

其中 process.h 和 process.cpp 内容为: 物理页(Ppage), 虚拟页(Vpage), 进程类(process)的定义与实现
四个页面置换算法(FIFO,LRU,NRU,OPT)的实现也在 Process 类中。

Multiprocess.h 和 multiprocess.cpp 内容为: 多进程类 (multiprocess)的定义与实现。主要内容是实现 A, B, C 三个进程的并发执行。包括有 A, B, C 三个由 Process 实例化的对象。



- res
 - framework.h
 - memorizer.aps
 - memorizer.cpp
 - memorizer.h
 - memorizer.rc
 - memorizer.vcxproj
 - memorizer.vcxproj.filters
 - memorizer.vcxproj.user
 - memorizerDlg.cpp
 - memorizerDlg.h
 - multiprocess.cpp
 - multiprocess.h
 - pch.cpp
 - pch.h
 - process.cpp
 - process.h
 - resource.h
 - targetver.h

测试数据:

A 进程调度: 1 2 3 4 1 2 5 1 2 3 4 5

B 进程调度: 12 4 3 20 3 1 20 12

C 进程调度: 2 3 5 3 6 8 4 3

测试结果——FIFO:

模拟内存管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序: 1 2 3 4 1 2 5 1 2 3 4 5

缺页率: 0.8333 置换率: 0.5

进程B, 大小:8M

页面调度顺序: 12 4 3 20 3 1 20 12

缺页率: 0.75 置换率: 0.25

进程C, 大小:9.8M

页面调度顺序: 2 3 5 3 6 8 4 3

缺页率: 0.875 置换率: 0.375

置换算法: FIFO 运行

| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
|-------|---|---|---|---|---|---|---|
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |

| 页面走向 | 12 | 4 | 3 | 20 | 3 | 1 | 20 |
|------|----|----|----|----|----|----|----|
| 物理块1 | 12 | 12 | 12 | 12 | 12 | 1 | 1 |
| 物理块2 | | 4 | 4 | 4 | 4 | 4 | 4 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 20 | 20 | 20 | 20 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | 缺 | |

| 页面走向 | 2 | 3 | 5 | 3 | 6 | 8 | 4 |
|------|---|---|---|---|---|---|---|
| 物理块1 | 2 | 2 | 2 | 2 | 2 | 8 | 8 |
| 物理块2 | | 3 | 3 | 3 | 3 | 3 | 4 |
| 物理块3 | | | 5 | 5 | 5 | 5 | 5 |
| 物理块4 | | | | | 6 | 6 | 6 |
| 缺页 | 缺 | 缺 | 缺 | | 缺 | 缺 | 缺 |

关闭

LRU:

模拟内存管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序: 1 2 3 4 1 2 5 1 2 3 4 5

缺页率: 0.6666 置换率: 0.3333

进程B, 大小:8M

页面调度顺序: 12 4 3 20 3 1 20 12

缺页率: 0.75 置换率: 0.25

进程C, 大小:9.8M

页面调度顺序: 2 3 5 3 6 8 4 3

缺页率: 0.75 置换率: 0.25

置换算法: LRU 运行

| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
|-------|---|---|---|---|---|---|---|
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 5 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |

| 页面走向 | 12 | 4 | 3 | 20 | 3 | 1 | 20 |
|------|----|----|----|----|----|----|----|
| 物理块1 | 12 | 12 | 12 | 12 | 12 | 1 | 1 |
| 物理块2 | | 4 | 4 | 4 | 4 | 4 | 4 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 20 | 20 | 20 | 20 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | 缺 | |

| 页面走向 | 2 | 3 | 5 | 3 | 6 | 8 | 4 |
|------|---|---|---|---|---|---|---|
| 物理块1 | 2 | 2 | 2 | 2 | 2 | 8 | 8 |
| 物理块2 | | 3 | 3 | 3 | 3 | 3 | 3 |
| 物理块3 | | | 5 | 5 | 5 | 5 | 4 |
| 物理块4 | | | | | 6 | 6 | 6 |
| 缺页 | 缺 | 缺 | 缺 | | 缺 | 缺 | 缺 |

关闭

NUR:

模拟内存器管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序 1 2 3 4 1 2 5 1 2 3 4 5

缺页率 0.6666 置换率 0.3333

进程B, 大小:8M

页面调度顺序 12 4 3 20 3 1 20 12

缺页率 0.75 置换率 0.25

进程C, 大小:9.8M

页面调度顺序 2 3 5 3 6 8 4 3

缺页率 0.75 置换率 0.25

置换算法 NUR 运行

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 5 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |
| 页面走向 | 12 | 4 | 3 | 20 | 3 | 1 | 20 |
| 物理块1 | 12 | 12 | 12 | 12 | 12 | 1 | 1 |
| 物理块2 | | 4 | 4 | 4 | 4 | 4 | 4 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 20 | 20 | 20 | 20 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | 缺 | |
| 页面走向 | 2 | 3 | 5 | 3 | 6 | 8 | 4 |
| 物理块1 | 2 | 2 | 2 | 2 | 2 | 8 | 8 |
| 物理块2 | | 3 | 3 | 3 | 3 | 3 | 3 |
| 物理块3 | | | 5 | 5 | 5 | 5 | 4 |
| 物理块4 | | | | | 6 | 6 | 6 |
| 缺页 | 缺 | 缺 | 缺 | | 缺 | 缺 | 缺 |

< 关闭

OPT:

模拟内存器管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序 1 2 3 4 1 2 5 1 2 3 4 5

缺页率 0.3333 置换率 0

进程B, 大小:8M

页面调度顺序 12 4 3 20 3 1 20 12

缺页率 0.625 置换率 0.125

进程C, 大小:9.8M

页面调度顺序 2 3 5 3 6 8 4 3

缺页率 0.75 置换率 0.25

置换算法 OPT 运行

| | | | | | | | |
|-------|----|----|----|----|----|----|----|
| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |
| 页面走向 | 12 | 4 | 3 | 20 | 3 | 1 | 20 |
| 物理块1 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 物理块2 | | 4 | 4 | 4 | 4 | 1 | 1 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 20 | 20 | 20 | 20 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | 缺 | |
| 页面走向 | 2 | 3 | 5 | 3 | 6 | 8 | 4 |
| 物理块1 | 2 | 2 | 2 | 2 | 2 | 8 | 4 |
| 物理块2 | | 3 | 3 | 3 | 3 | 3 | 3 |
| 物理块3 | | | 5 | 5 | 5 | 5 | 5 |
| 物理块4 | | | | | 6 | 6 | 6 |
| 缺页 | 缺 | 缺 | 缺 | | 缺 | 缺 | 缺 |

< 关闭

其他错误处理：
输入数字与空格之外的其他字符：

模拟内存器管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序 1 3

缺页率 0.3333 置换率 0

进程B, 大小:8M

页面调度顺序 12 4 3 20 3 1 20 12

缺页率 0.625 置换率 0.125

进程C, 大小:9.8M

页面调度顺序 2 3 5 3 6 8 4 3

缺页率 0.75 置换率 0.25

置换算法 FIFO 运行

| | | | | | | | |
|-------|---|---|---|---|---|---|---|
| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |

错误

! 页面调度顺序输入错误!
只能输入正整数和空格!

确定

| | | | | |
|--|----|----|----|----|
| | 20 | 3 | 1 | 20 |
| | 12 | 12 | 12 | 12 |
| | 4 | 4 | 1 | 1 |
| | 3 | 3 | 3 | 3 |
| | 20 | 20 | 20 | 20 |
| | 缺 | | 缺 | |

| | | | | | | | |
|------|---|---|---|---|---|---|---|
| 物理块1 | 2 | 2 | 2 | 3 | 6 | 8 | 4 |
| 物理块2 | | 3 | 3 | 2 | 2 | 8 | 4 |
| 物理块3 | | | 5 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 5 | 5 | 5 | 5 |
| 缺页 | 缺 | 缺 | 缺 | | 6 | 6 | 6 |

< >

关闭

模拟内存器管理

内存:8M 外存:25M 页面大小:512K

进程A, 大小:4.2M

页面调度顺序 1 -3

缺页率 0.3333 置换率 0

进程B, 大小:8M

页面调度顺序 12 4 3 20 3 1 20 12

缺页率 0.625 置换率 0.125

进程C, 大小:9.8M

页面调度顺序 2 3 5 3 6 8 4 3

缺页率 0.75 置换率 0.25

置换算法 FIFO 运行

| | | | | | | | |
|-------|---|---|---|---|---|---|---|
| 页面... | 1 | 2 | 3 | 4 | 1 | 2 | 5 |
| 物理块1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 物理块2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| 物理块3 | | | 3 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 4 | 4 | 4 | 4 |
| 缺页 | 缺 | 缺 | 缺 | 缺 | | | 缺 |

错误

! 页面调度顺序输入错误!
只能输入正整数和空格!

确定

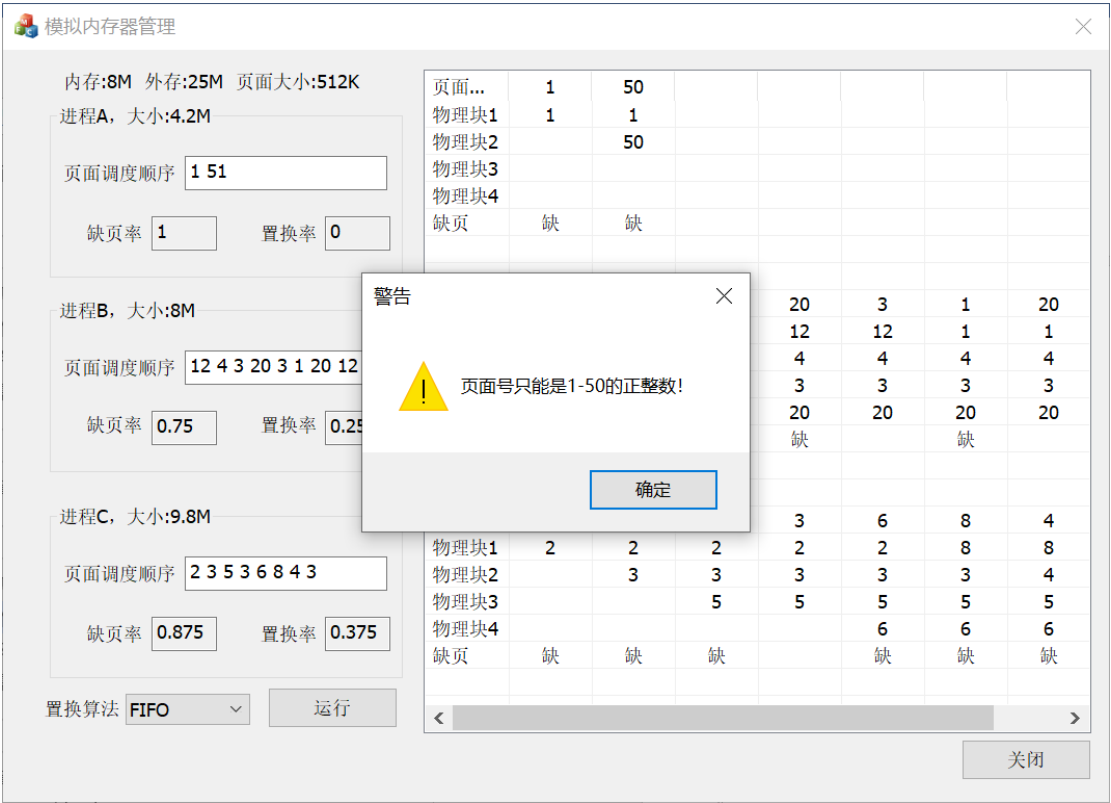
| | | | | |
|--|----|----|----|----|
| | 20 | 3 | 1 | 20 |
| | 12 | 12 | 12 | 12 |
| | 4 | 4 | 1 | 1 |
| | 3 | 3 | 3 | 3 |
| | 20 | 20 | 20 | 20 |
| | 缺 | | 缺 | |

| | | | | | | | |
|------|---|---|---|---|---|---|---|
| 物理块1 | 2 | 2 | 2 | 3 | 6 | 8 | 4 |
| 物理块2 | | 3 | 3 | 2 | 2 | 8 | 4 |
| 物理块3 | | | 5 | 3 | 3 | 3 | 3 |
| 物理块4 | | | | 5 | 5 | 5 | 5 |
| 缺页 | 缺 | 缺 | 缺 | | 6 | 6 | 6 |

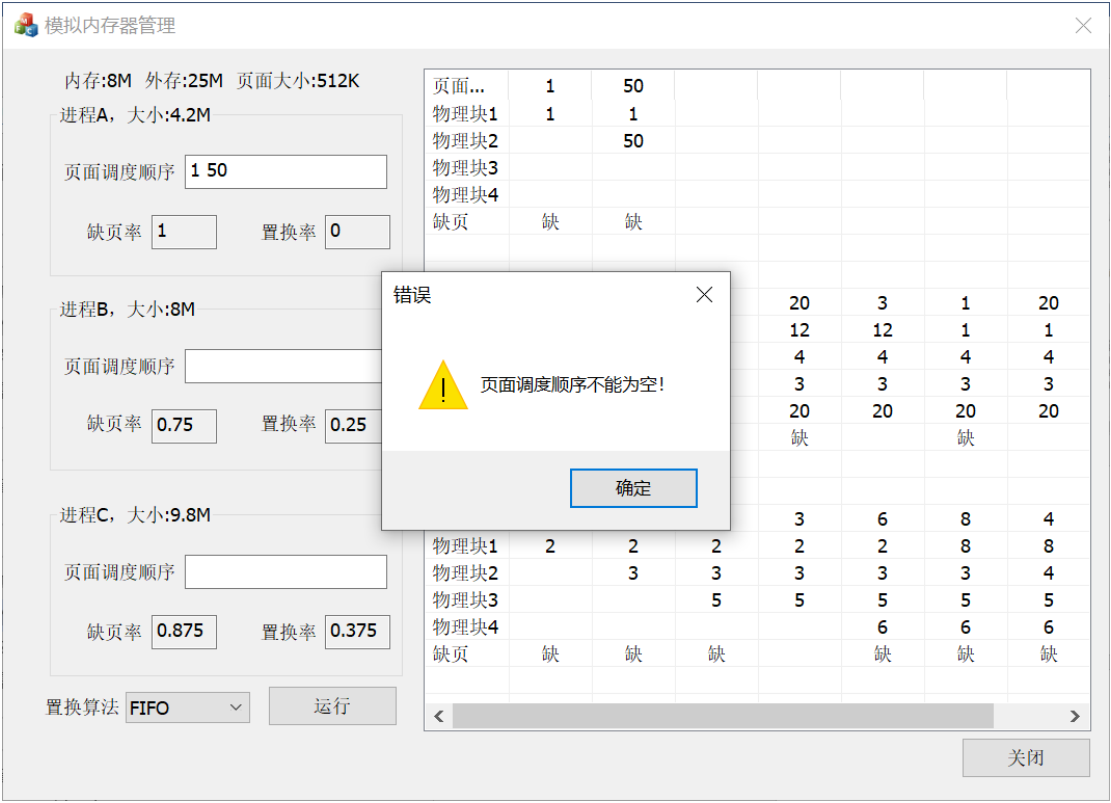
< >

关闭

超出页面大小:



页面调度顺序为空:



3. 模拟文件系统的设计与实现

3.1 需求规格说明

3.1.1 问题描述

(1) 设计内容

1) 设计一个 10 个用户的文件系统。每个用户最多可以保存 10 个文件，一次运行用户可打开多个文件。

2) 程序采用二级文件目录。(即设置主目录(MFD)和用户文件目录(UFD))。另外，可打开文件设置指针。

3) 为了方便实现，对文件的读写作了简化。在执行读写命令时，只需改读写指针。并不进行实际的读写操作。

4) 实现的基本功能主要包括：改变目录(CD)，创建目录(MD)，显示目录(DIR)，删除目录(RD)，打开全部文件(openall)，打开单个文件(open)，建立一个文件(create)，删除一个文件(delete)，写文件(write)，读文件(read)，修改文件的保护码(change)，退出(exit)等。

(2) 要求

考虑特殊情况如：各个命令对全路径和相对路径的支持、目录不存在时，给出错误信息、不能用 cd 进入文件、命令之中不能有空格(如 ex it，给出错误提示)、相对路径的解析、路径中的空格剔除、新建目录或文件时的问题、重名问题、目录或文件的名字长度限制、目录或文件的名字中包含不合法字符(注意空格)、删除目录或文件时的问题、删除不存在的文件或目录给出错误提示、删除目录时目录不为空(如果该目录为空，则可删除，否则给出是否做删除提示，删除操作将该目录下的全部文件和子目录都删除)、进入到某个目录下，却要删除本目录或上级目录、不能用 delete 删除目录、不能用 RD 删除文件等都要考虑在内。

3.1.2 问题分析

采用二级目录，且有十个用户，则有一个主目录和最多十个用户目录，用户目录下不能再创建目录。根目录下不能创建文件。主目录下的用户目录数不能多于 10 个，用户目录下的文件数不能多于 10 个。

对文件进行写操作时必须要先打开文件，不打开就不能写，实际情况也是如此。

3.2 算法设计

3.2.1 设计思想

1. 定义一个结构体 `file` 表示文件，包括有文件名，文件内容等属性。为了方便进行文件的增删查改，我在这题中使用到了链表的思想，因此我在文件结构体中还加入了一个下一文件指针。

2. 定义一个结构体 `directory` 来表示目录，包括有目录名，下一目录指针，上一级目录指针，目录下的文件指针，目录下的目录指针等属性。

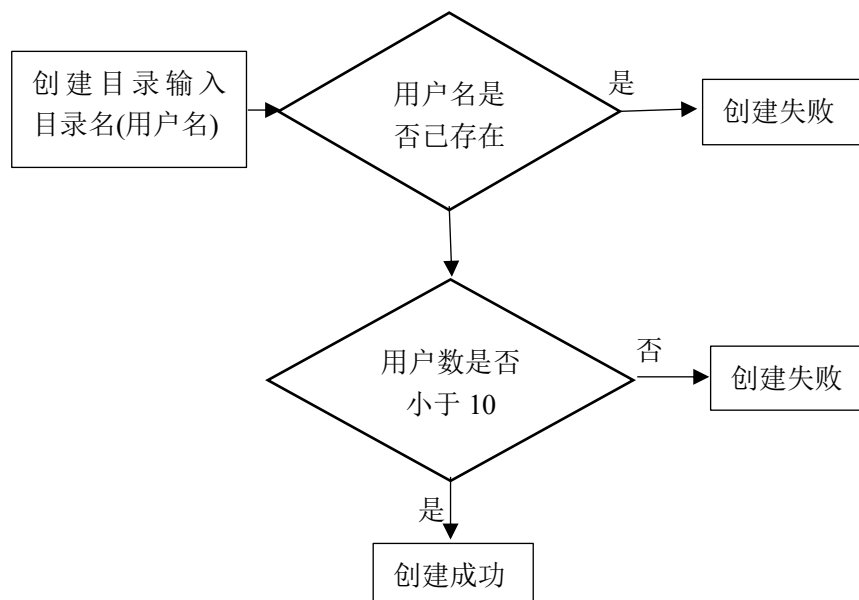
3. 定义一个 `filesystem` 类来表示文件系统，包括有当前目录，根目录以及文件系统的各种操作函数（改变/创建/删除目录，创建/删除/读/写/打开文件等），还包括一个 `run` 函数，表示运行文件系统，在 `run` 中根据输入的指令来调用不同的操作函数以模拟文件系统的操作。

4. 题目中要求二级目录，且有十个用户，所以文件系统包括一个根目录以及根目录下有若干个用户目录（最多不能超过十个）。而在用户目录下可以创建文件，并进行文件的操作。

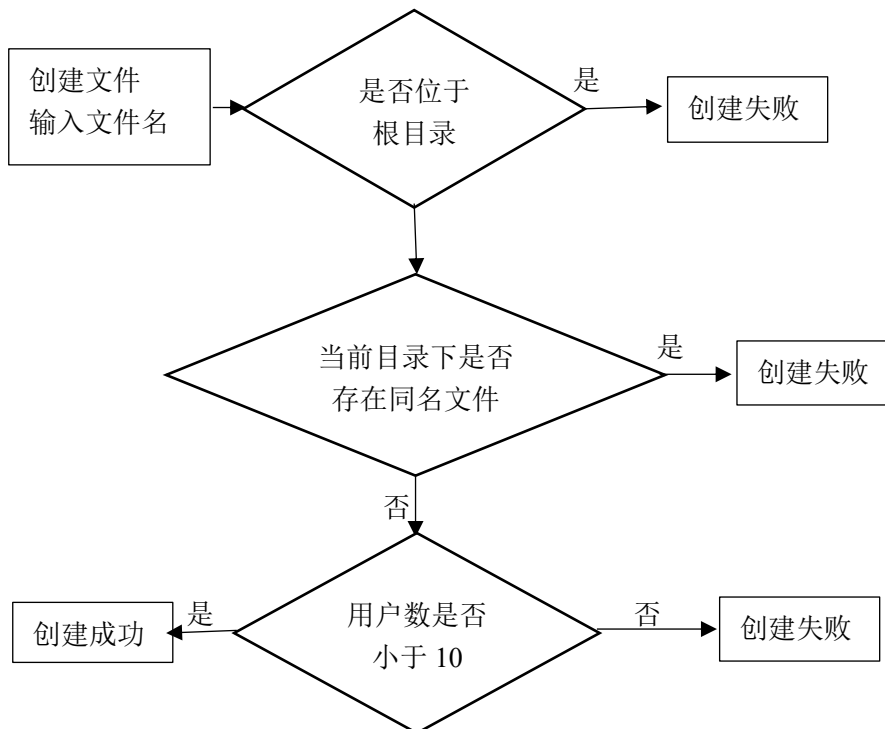
5. 创建文件或创建目录时，如果当前文件数或目录数没有超过可允许的最大值且不存在同名的文件或目录，则将该文件或目录添加到相应的链表下。

3.2.2 设计表示

1. 创建目录：输入 `md+要创建的目录名`。在本题中，由于二级目录和多用户的限制，创建目录也就是创建用户目录。



2.创建文件：输入 `create`+要创建的文件名。创建文件要转到用户目录下。



3.改变目录：输入 `cd`+要改变到的目录名。如果在当前目录链表中查不到该目录，则改变目录失败；否则当前目录就变成要改变的目录。

也可以输入 `cd ..`返回上一级目录。

4.删除目录：输入 `rd`+要删除的目录名。然后在当前目录下查找该目录名，如果查找不到则删除失败；如果查找到了，若目录下不为空，则需要用户二次确认才会删除。

5.读文件：输入 `read`+文件名，在当前目录下查找该文件，若查找成功则输出该文件内容；否则读文件失败。

6.打开文件：输入 `open`+文件名，在当前目录下查找该文件，若查找成功则打开该文件并输出该文件内容。也可直接输入 `openall` 打开所有文件。

7.写文件：输入 `write`+文件名，在当前目录下查找该文件，若查找成功且该文件以及打开则用户可以输入要写入文件的内容。

8.删除文件：输入 `delete`+文件名，在当前目录下查找该文件，若查找成功则删除该文件。

9.显示目录：输入 `dir`，则会显示出当前目录下的所有文件与目录。

10.退出：输入 `exit` 退出系统。

3.3 调试报告

遇到的问题 1：无法向文件中写入多行数据

解决方法： 写文件时每次回车让用户选择是否结束输入。

遇到的问题 2：打开文件与读文件的区别？

解决方法：打开文件才能对文件进行写操作。读文件不需要对文件进行打开操作。

改进思想：

1.创建的信息能保存下来中，每次打开系统之前的东西都能保存下来，而不是每次打开都是一个新的文件系统。创建的文件也能实实在在的保存在本地。

2.为文件添加保护码。

3.4 附录

源代码清单：

代码主要在 filesystem.h 和 filesystem.cpp 中
内容为文件结构体(file)，目录结构体(directory)，
以及文件系统类(filesystem)的实现。

(resource.h 等文件是之前尝试做界面的时候添加
的，实际上后来没有把界面做出来)

名称

filesystem.aps
filesystem.cpp
filesystem.h
filesystem.rc
filesystem.vcxproj
filesystem.vcxproj.filters
filesystem.vcxproj.user
main.cpp
resource.h

一个完整的测试用例:

```
C:\Users\yangt\source\repos\filesystem\Debug\filesystem.exe
1.create 文件名----创建文件
2.write 文件名----写文件
3.read 文件名-----读文件
4.delete 文件名----删除文件
5.open 文件名-----打开文件
6.openall-----打开所有文件
7.md 目录名-----创建目录(用户目录)
8.rd 目录名-----删除目录(用户目录)
9.dir-----显示目录
10.cd 目录名-----更改目录
11.cd ..-----返回上一级目录
12.exit-----退出
root/>md ul
创建成功!
root/>cd ul
更改目录成功!
root/ul/>create file1
创建成功!
root/ul/>open file1
打开成功!
file1:
root/ul/>write file1
123
是否要结束输入? y/n n
456
是否要结束输入? y/n n
789
是否要结束输入? y/n y
写入成功!
root/ul/>read file1
读取成功!
文件内容为:
123
456
789
root/ul/>create file2
创建成功!
root/ul/>dir
ul:
    文件:
    file2
    file1

    目录:
root/ul/>delete file2
删除成功
root/ul/>dir
ul:
    文件:
    file1

    目录:
root/ul/>cd ..
更改目录成功
root/>rd ul
确定删除该目录下的所有文件和目录? y/n
y
删除成功!
root/>dir
root:
    文件:

    目录:
root/>exit
请按任意键继续. . .
```

其他错误处理:

根目录下创建文件:

```
root/>create file
创建失败! 请转到用户目录下创建文件!
root/>_
```

在未打开文件的情况下写文件:

```
root/user1/>create file1
创建成功!
root/user1/>write file1
文件未打开! 请先打开该文件再进行写操作!
root/user1/>
```

创建文件的文件名过长:

```
root/user1/>create abcdefghijklmnopqrstuvwxyz
文件名过长!
```

创建文件数达上限:

```
root/user1/>dir
user1:
    文件:
    k
    j
    i
    h
    g
    f
    d
    c
    b
    a
    目录:
root/user1/>create 1
创建失败! 文件数达上限!
```

创建用户数达上限:

```
root/>dir
root:
    文件:
    目录:
    user10
    user9
    user8
    user7
    user6
    user5
    user4
    user3
    user2
    user1
root/>md user11
创建错误! 用户目录已达上限
```

命令错误:

```
root/>ex it
无法识别的语句
```

4. 总 结

这次课程设计三个题目涉及进程调度，页面置换，文件系统三个部分，这三部分也是这学期操作系统课程学习的几大重点内容。第一题之前课内实习的时候做过类似的，甚至还比课内实习的那题简单一点。所以第一题并没有花费太多的时间，主要就是解决界面的问题。第二题其实那几个页面置换并没有花费太多的时间，大多数时间都花费在总表的显示以及进程的并发执行上。并发执行的实现我觉得对我来说是一大难点，但是通过第二题我也学会了不少相关的方法。第三题由于对这部分内容相对没有前面那么熟练，同样也因为时间问题，所以做得也比较简单。但是在这过程中，我对文件系统有了更深的了解，现在对这一部分内容也更加熟练了。

通过这次课程设计，我对这学期操作系统原理课程学习的内容有了一个更深的了解，同时也在界面设计方面学习到了许多。虽然这次实习的题目量不是很多，但课程设计的那两周也花费了很多的时间在这上面，虽然还有一些内容没有实现（第一题的抢占式短作业优先算法，第三题的界面以及保存下文件系统的信息），但我已经是尽最大努力在做了。

对本课程设计的建议与意见：希望老师能提供一些相关的资料以及示例程序，就比如说第三题这样的题目如果能有示例程序的话就不至于在理解题目意思上耗费太多的时间。