东北大学软件学院
SOFTWARE COLLEGE OF
NORTHEASTERN UNIVERSITY

操作系统实验指导书

适用专业：软件工程

实验学时：**8**

开课单位：软件工程系

东北大学 软件学院

2025 年 4 月

目　录

前　言

一、实验目的

操作系统是计算机的核心和灵魂。操作系统软件的设计对整个计算机的功能和性能起着至关重要的作用，所以此门课也是必不可少的。本课程的目的是使学生掌握现代计算机操作系统的基本原理、基本设计方法及实现技术，具有分析现行操作系统和设计、开发实际操作系统的基本能力。

操作系统实验是操作系统课程的重要组成部分，属于学科基础实验范畴，是与相关教学内容配合的实践性教学环节，其作用是：理解操作系统的设计和实现思路，掌握典型算法。

二、实验内容

操作系统对计算机系统资源实施管理，是所有其他软件与计算机硬件的唯一接口，所有用户在使用计算机时都要得到操作系统提供的服务。本实验的任务是使学生结合具体实践更好地理解和掌握进程管理、处理机管理、存储器管理、设备管理和文件管理等主要功能的设计及实现原理。

三、实验要求

(1)预习实验指导书有关部分，认真做好实验的准备工作。

(2)实验中及时分析记录。

(3)按指导书要求书写实验报告，提交到BlackBoard平台。

(4)每名同学独立完成。实验报告需要按照报告模板撰写。

实验的验收将分为两个部分。第一部分是上机操作，包括检查程序运行和即时提问。第二部分是提交的实验报告。

实验一　进程/线程的同步与互斥

1 实验目的

（1） 加深对进程概念/线程概念的理解；
（2） 进一步认识并发的实质；
（3） 分析进程/线程竞争资源的现象，学习解决进程互斥的方法。
（4） 了解 Windows 对进程/线程管理的支持。

2 实验类型

综合型。实验环境：Windows 操作系统/Linux 操作系统。

3 复习和预习内容

复习进程同步和互斥的相关理论，预习 PThreads 的相关理论和用法，包括线程的管理，互斥锁，条件变量等。实验要求编写一个简单的生产者/消费者问题示例程序。

4 实验要求和相关知识介绍

实验要求：编写一个简单的生产者/消费者示例程序。生产者消费者问题是一个著名的线程同步问题，该问题描述如下：有一个生产者在生产产品，这些产品将提供给若干个消费者去消费，为了使生产者和消费者能并发执行，在两者之间设置一个具有多个缓冲区的缓冲池，生产者将它生产的产品放入一个缓冲区中，消费者可以从缓冲区中取走产品进行消费，显然生产者和消费者之间必须保持同步，即不允许消费者到一个空的缓冲区中取产品，也不允许生产者向一个已经放入产品的缓冲区中再次投放产品。

实验中可以使用任何一种线程库，例如 Windows Thread，POSIX Thread，Java Thread。

4.1 使用 Windows Thread

在 Windows 中创建线程可以调用两个函数_beginthreadex 和 CreateThread 两个函数。
（1）使用 CreateThread 在 Windows 下创建线程

CreateThread 函数原型:

```
HANDLE CreateThread
  (LPSECURITY_ATTRIBUTES lpThreadAttributes , //pointer to thread security
```

```
attributes
    DWORD dwStackSize，//initial thread stack size，  in bytes
    LPSECURITY_START_ROUTINE lpStartAddress，//pointer to thread function
    LPVOID lpParameter，//argument for new thread
    DWORD dwCreationFlags，//creation flags
    LPDWORD lpThreadId //pointer to returned thread identifier
)
```

其中，在本实验阶段比较重要的参数是第三和第四个:

a)第三个参数是一个指向函数的指针，所以应传入的参数应为函数的地址，如&Func 的形式.而这个传入的参数，则必须被声明成为:

```
DWORD WINAPI threadFunc(LPVOID threadArgu);
```

这个函数也就是要执行线程任务的那个函数体实体.这里应注意，传入应使用 Func 而非 &Func。

如：CreateThread(NULL，0，Func，…)

b)第四个参数应是执行线程任务的函数体实体所需要的参数，即上面所举例的函数 threadFunc 的参数 threadArgu，这在 WINDOWS 中被定义成一个 LPVOID 的类型，目前我认为，可以把它在功能上看成和 void* 类似。

参考:LPVOID 的原型:

```
typedef void far *LPVOID;
```

所以，当你有自己需要的类型的参数传入时，可以用

```
typedef struct
{
int firstArgu,
long secArgu,
…

}myType，* pMyType;
```

将你想要传入的参数装入一个结构体中。

在传入点，使用类似:

```
pMyType pMyTpeyArgu;
…
CreateThread(NULL，0，threadFunc，pMyTypeArgu，…);
…
```

在函数 threadFunc 内部的接收点，可以使用“强行转换”，如:

```
int intValue=((pMyType)lpvoid)->firstArgu;
long longValue=((pMyType)lpvoid)->secArgu;
… …
```

例如创建 N 个随机线程，所有线程的执行时间均为 T 秒，观察每个线程的运行状况:

为了使线程的运行趋于随机化，应先使用:

```
srand((unsigned int)time (NULL));
```

在每个线程的运行中，每个线程的睡眠时间为:

```
sleepTime=1000+30*(int)eRandom(50+tNo);
Sleep(sleepTime);
```

这样，可以使进程的运行趋于随机化。

4.2 使用 POSIX Thread

（1）什么是 POSIX Thread

POSIX 线程（英语：POSIX Threads，常被缩写为 Pthreads）是 POSIX 的线程标准，定义了创建和操纵线程的一套 API。

实现 POSIX 线程标准的库常被称作 Pthreads，一般用于 Unix-likePOSIX 系统，如 Linux、Solaris。但是 Microsoft Windows 上的实现也存在，例如直接使用 Windows API 实现的第三方库 pthreads-w32；而利用 Windows 的 SFU/SUA 子系统，则可以使用微软提供的一部分原生 POSIX API。

Pthreads 定义了一套 C 语言的类型、函数与常量，它以 pthread.h 头文件和一个线程库实现。

Pthreads API 中大致共有 100 个函数调用，全都以"pthread_"开头，并可以分为四类：

（a）线程管理，例如创建线程，等待(join)线程，查询线程状态等。

（b）互斥锁（Mutex）：创建、摧毁、锁定、解锁、设置属性等操作。

（c）条件变量（Condition Variable）：创建、摧毁、等待、通知、设置查询属性等操作。

（d）使用了互斥锁的线程间的同步管理。

（2）PThread 常用函数：

pthread_create()：创建一个线程

pthread_exit()：终止当前线程

pthread_cancel()：请求中断另外一个线程的运行。被请求中断的线程会继续运行，直至到达某个取消点(Cancellation Point)。取消点是线程检查是否被取消并按照请求进行动作的一个位置。POSIX 的取消类型（Cancellation Type）有两种，一种是延迟取消(PTHREAD_CANCEL_DEFERRED)，这是系统默认的取消类型，即在线程到达取消点之前，不会出现真正的取消；另外一种是异步取消(PHREAD_CANCEL_ASYNCHRONOUS)，使用异步取消时，线程可以在任意时间取消。系统调用的取消点实际上是函数中取消类型被修改为异步取消至修改回延迟取消的时间段。几乎可以使线程挂起的库函数都会响应 CANCEL 信号，终止线程，包括 sleep、delay 等延时函数。

pthread_join()：阻塞当前的线程，直到另外一个线程运行结束。

pthread_kill()：向指定 ID 的线程发送一个信号，如果线程不处理该信号，则按照信号默认的行为作用于整个进程。信号值 0 为保留信号，作用是根据函数的返回值判断线程是不是还活着。

pthread_cleanup_push()：线程可以安排异常退出时需要调用的函数，这样的函数称为线程清理程序，线程可以建立多个清理程序。线程清理程序的入口地址使用栈保存，实行先进后处理原则。由 pthread_cancel 或 pthread_exit 引起的线程结束，会次序执行由 pthread_cleanup_push 压入的函数。线程函数执行 return 语句返回不会引起线程清理程序被执行。

pthread_cleanup_pop()：以非 0 参数调用时，引起当前被弹出的线程清理程序执行。

pthread_setcancelstate()：允许或禁止取消另外一个线程的运行。

pthread_setcanceltype()：设置线程的取消类型为延迟取消或异步取消。

（3）在 Windows 下配置 PThread

若平台为 Windows 10 64 位系统，编译环境为 Microsoft Visual Studio，则可以按照以下步骤进行设置。

第一步：下载 pthreads

下载地址为： ftp://sourceware.org/pub/pthreads-win32/pthreads-w32-2-9-1-release.zip ，下载后解压可以得到 3 个文件夹，分别为“Pre built.2”，“pthreads.2”,“QueueUserAPCEX”。打开“Pre-built.2”可以得到新的 3 个子文件夹：“dll”，“include”，“lib”，分别是动态链接库，头文件和静态链接库。

第二步：配置头文件

将上面“include”文件夹中的 3 个文件拷贝到 Visual Studio 的安装目录中 include 目录下：C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.25.28610\include （不同计算机目录位置不一样）

第三步：配置静态链接库

将第一步中“lib”文件夹中的 x86 文件夹中的 5 个文件，复制到：C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.25.28610\lib\x86 中。将第一步中“lib”文件夹中的 x64 文件夹中的 2 个文件，复制到：C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.25.28610\lib\x64 中。（不同计算机目录位置不一样）

第四步：配置动态链接库

将第一步中“dll”文件夹中的“x64”文件夹中的两个文件拷贝到 C:\Windows\SysWOW64 目录下。将第一步中“dll”文件夹中的“x86"文件夹中的 5 个文件拷贝到 C:\Windows\System32目录下。

5 思考题

（1） 如何控制进程间的相互通信？

（2） 什么事进程同步？什么是进程互斥？分别有哪些实现方式？

实验二　处理机调度(2 学时)

一、实验目的

（1）加深对处理机调度的作用和工作原理的理解。

（2）进一步认识并发执行的实质。

二、实验类型

设计型。

三、预习内容

预习课本处理机调度有关内容，包括进程占用处理机的策略方法。

四、实验要求与提示

本实验有两个题，针对这两个问题编写代码并撰写实验报告：

第一题：设计一个按优先权调度算法实现处理器调度的程序。

【提示】

（1）假定系统有五个进程，每一个进程用一个进程控制块 PCB 来代表，进程控制块的格式为：

进程名
指针
要求运行时间
优先数
状态

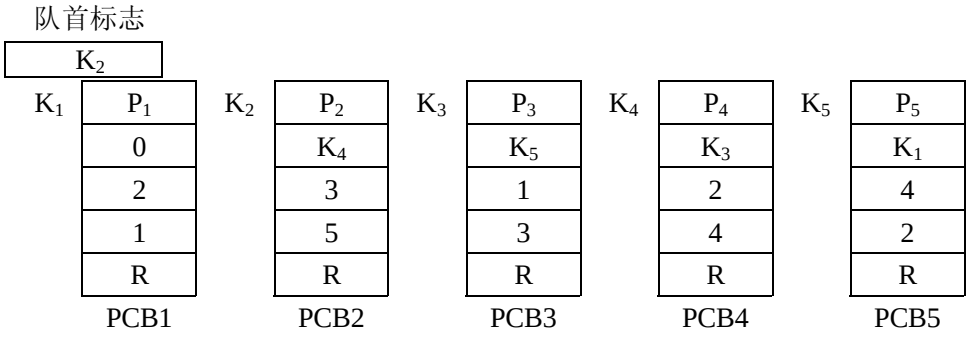
其中，

- 进程名——作为进程的标识，假设五个进程的进程名分别为 P_1，P_2，P_3，P_4，P_5。
- 指针——按优先数的大小把五个进程连成队列，用指针指出下一个进程的进程控制块的首地址，最后一个进程中的指针为“0”。
- 要求运行时间——假设进程需要运行的单位时间数。
- 优先数——赋予进程的优先权，调度时总是选取优先数大的进程先执行。
- 状态——可假设有三种状态，“就绪”状态（ready）、“运行”状态（working）和“结束”状态（finish）。五个进程的初始状态都为“就绪”，用“R”表示；当进程

运行结束后，它的状态为“结束”，用“E”表示；当进程被选中开始运行但尚未结束时，它的状态为“运行”，用“W”表示。

（2） 在每次运行你所设计的处理器调度程序之前，为每个进程任意确定它的“优先数”和“要求运行时间”。

（3）为了调度方便，把五个进程按给定的优先数从大到小连成队列。用一单元指出队首进程，用指针指出队列的连接情况。例：



（4）处理器调度总是选队首进程运行。采用动态改变优先数的办法，进程每运行一次优先数就减“1”。由于本实验是模拟处理器调度，所以，对被选中的进程并不实际的启动运行，而是执行：

优先数-1

要求运行时间-1

来模拟进程的一次运行。

提醒注意的是：在实际的系统中，当一个进程被选中运行时，必须恢复进程的现场，让它占有处理器运行，直到出现等待事件或运行结束。在这里省去了这些工作。

（5）进程运行一次后，若要求运行时间≠0，则再将它加入队列（按优先数大小插入，且置队首标志）；若要求运行时间=0，则把它的状态修改成“结束”（E），且退出队列。

（6）若“就绪”状态的进程队列不为空，则重复上面（4）和（5）的步骤，直到所有进程都成为“结束”状态。

（7）在所设计的程序中应有显示或打印语句，能显示或打印每次被选中进程的进程名以及运行一次后进程队列的变化。

（8）为五个进程任意确定一组“优先数”和“要求运行时间”，启动所设计的处理器调度程序，显示或打印逐次被选中进程的进程名以及进程控制块的动态变化过程。例如，下面的运行结果示意：

INPUT NAME , NEEDTIME AND PRIORITY

P1　　2　　1

P2　　3　　5

P3　　1　　3

P4　　2　　4

P5　　4　　2

OUTPUT　OF　PRIORITY：

CPUTIME:0

NAME	CPUTIME	NEEDTIME	PRIORITY	STATE
P1	0	2	1	ready
P2	0	3	5	ready
P3	0	1	3	ready
P4	0	2	4	ready
P5	0	4	2	ready

CPUTIME:1

P1	0	2	1	ready
P2	1	2	4	working
P3	0	1	3	ready
P4	0	2	4	ready
P5	0	4	2	ready

CPUTIME:2

P1	0	2	1	ready
P2	2	2	4	ready
P3	0	1	3	ready
P4	1	1	3	working
P5	0	4	2	ready

CPUTIME:3

P1	0	2	1	ready
P2	3	1	3	working
P3	0	1	3	ready
P4	2	1	3	ready
P5	0	4	2	ready

CPUTIME:4

P1	0	2	1	ready
P2	4	1	3	ready
P3	1	0	2	finish
P4	3	1	3	ready
P5	0	4	2	ready

CPUTIME:5

P1	0	2	1	ready
P2	5	1	3	ready
P3	1	0	2	finish
P4	4	0	2	finish
P5	0	4	2	ready

CPUTIME:6

P1	0	2	1	ready
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	0	4	2	ready

CPUTIME:7

P1	0	2	1	ready
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	1	3	1	working

CPUTIME:8

P1	1	1	0	working
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	2	3	1	ready

CPUTIME:8

P1	2	1	0	ready
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	3	2	0	working

CPUTIME:9

P1	3	0	-1	finish
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	4	2	0	ready

CPUTIME:10

P1	3	0	-1	finish
P2	6	0	2	finish
P3	1	0	2	finish
P4	4	0	2	finish
P5	5	1	-1	working

CPUTIME:11

P1	3	0	-1	finish
P2	6	0	2	finish

P3	1	0	2	finish
P4	4	0	2	finish
P5	6	0	-2	finish

NAME	RoundTime	WaitingTime
P1	9	7
P2	6	3
P3	4	3
P4	5	3
P5	11	7

第二题：设计一个按时间片轮转法实现处理器调度的程序。

【提示】

（1）假定系统有五个进程，每一个进程用一个进程控制块 PCB 来代表。进程控制块的格式为：

进程名
指针
要求运行时间
已运行时间
状态

其中，

- 进程名——作为进程的标识，假设五个进程的进程名分别为 Q_1，Q_2，Q_3，Q_4，Q_5。
- 指针——进程按顺序排成循环队列，用指针指出下一个进程的进程控制块的首地址，最后一个进程的指针指出第一个进程的进程控制块首地址。
- 要求运行时间——假设进程需要运行的单位时间数。
- 已运行时间——假设进程已经运行的单位时间数，初始值为“0”。
- 状态——有两种状态，“就绪”和“结束”，初始状态都为“就绪”，用“R”表示。当一个进程运行结束后，它的状态为“结束”，用“E”表示。

（2）每次运行所设计的处理器调度程序前，为每个进程任意确定它的“要求运行时间”。

（3）把五个进程按顺序排成循环队列，用指针指出队列连接情况。另用一标志单元记录轮到运行的进程。例如，当前轮到 P_2 执行，则有：

标志单元

K_2

K_1	K_2	K_3	K_4	K_5
Q_1	Q_2	Q_3	Q_4	Q_5
K_2	K_3	K_4	K_5	K_1
2	3	1	2	4
1	0	0	0	0
R	R	R	R	R
PCB1	PCB2	PCB3	PCB4	PCB5

（4）处理器调度总是选择标志单元指示的进程运行。由于本实验是模拟处理器调度的功能，所以，对被选中的进程并不实际的启动运行，而是执行：

已运行时间+1

来模拟进程的一次运行，表示进程已经运行过一个单位的时间。

请同学注意：在实际的系统中，当一个进程被选中运行时，必须置上该进程可以运行的时间片值，以及恢复进程的现场，让它占有处理器运行，直到出现等待事件或运行满一个时间片。在这时省去了这些工作，仅用“已运行时间+1”来表示进程已经运行满一个时间片。

（5）进程运行一次后，应把该进程的进程控制块中的指针值送到标志单元，以指示下一个轮到运行的进程。同时，应判断该进程的要求运行时间与已运行时间，若该进程的要求运行时间≠已运行时间，则表示它尚未执行结束，应待到下一轮时再运行。若该进程的要求运行时间=已运行时间，则表示它已经执行结束，应指导它的状态修改成“结束”（E）且退出队列。此时，应把该进程的进程控制块中的指针值送到前面一个进程的指针位置。

（6）若“就绪”状态的进程队列不为空，则重复上面的（4）和（5）的步骤，直到所有的进程都成为“结束”状态。

（7）在所设计的程序中应有显示或打印语句，能显示或打印每次选中进程的进程名以及运行一次后进程队列的变化。

（8）为五个进程任意确定一组“要求运行时间”，启动所设计的处理器调度程序，显示或打印逐次被选中的进程名以及进程控制块的动态变化过程。输出要求与上述优先权优先调度输出形式类似。

五、思考题

（1）处理机调度的目的？

（2）你实现优先权调度算法的思想？

（3）你采用时间片轮转法实现处理机调度的思想？

（4）比较效率如何？

六、实验报告

（1）实验题目。

（2）程序中使用的数据结构及符号说明。

（3）流程图。

（4）实现的核心代码。

（5）按题目要求格式得到的程序运行时的初值和运行结果。

实验三　存储管理(4 学时)

一、实验目的

（1）加深对存储管理的作用和工作原理的理解。

（2）进一步认识主存空间的分配和回收方法。

（3）进一步认识虚拟存储器的工作原理。

（4）进一步认识文件系统的内部功能及内部实现。

二、实验类型

综合型。

三、预习内容

预习课本存储管理有关内容，包括各种内存分配方法和利用分页式存储管理实现虚拟存储器策略方法。

预习课本文件系统有关内容，包括文件和目录的创建和删除等基本原理。

四、实验要求与提示

本实验有四个题，其中第一、第四题必做，第二、第三题中任选一个。

第一题：模拟分页式存储管理中硬件的地址转换和产生缺页中断。

【提示】

（1）分页式虚拟存储系统是把作业信息的副本存放在磁盘上，当作业被选中时，可把作业的开始几页先装入主存且启动执行。为此，在为作业建立页表时，应说明哪些页已在主存，哪些页尚未装入主存，页表的格式为：

页号	标志	主存块号	在磁盘上的位置

其中，

- 标志——用来表示对应页是否已经装入主存，标志位=1，则表示该页已经在主存，标志位=0，则表示该页尚未装入主存。
- 主存块号——用来表示已经装入主存的页所占的块号。

l 在磁盘上的位置——用来指出作业副本的每一页被存放在磁盘上的位置。

（2）作业执行时，指令中的逻辑地址指出了参加运算的操作数存放的页号和单元号，硬件的地址转换机构按页号查页表，若该页对应标志为“1”，则表示该页已在主存，这时根据关系式：

绝对地址=块号×块长+单元号

计算出欲访问的主存单元地址。如果块长为 2 的幂次，则可把块号作为高地址部分，把单元号作为低地址部分，两者拼接而成绝对地址。按计算出的绝对地址可以取到操作数，完成一条指令的执行。若访问的页对应标志为“0”，则表示该页不在主存，这时硬件发“缺页中断”信号，由操作系统按该页在磁盘上的位置，把该页信息从磁盘读出装入主存后再重新执行这条指令。

（3）设计一个“地址转换”程序来模拟硬件的地址转换工作。当访问的页在主存时，则形成绝对地址，但不去模拟指令的执行，而用输出转换后的地址来代替一条指令的执行。当访问的页不在主存时，则输出“*该页页号”，表示产生了一次缺页中断。该模拟程序的算法如图 3-1。

（4）假定主存的每块长度为 128 个字节；现有一个共七页的作业，其中第 0 页至第 3 页已经装入主存，其余三页尚未装入主存；该作业的页表为：

0	1	5	011
1	1	8	012
2	1	9	013
3	1	1	021
4	0		022
5	0		023
6	0		121

如果作业依次执行的指令序列为：

操作	页号	单元号	操作	页号	单元号
+	0	070	移位	4	053
+	1	050	+	5	023
×	2	015	存	1	037
存	3	021	取	2	078
取	0	056	+	4	001
-	6	040	存	6	084

运行设计的地址转换程序，显示或打印运行结果。因仅模拟地址转换，并不模拟指令的执行，故可不考虑上述指令序列中的操作。

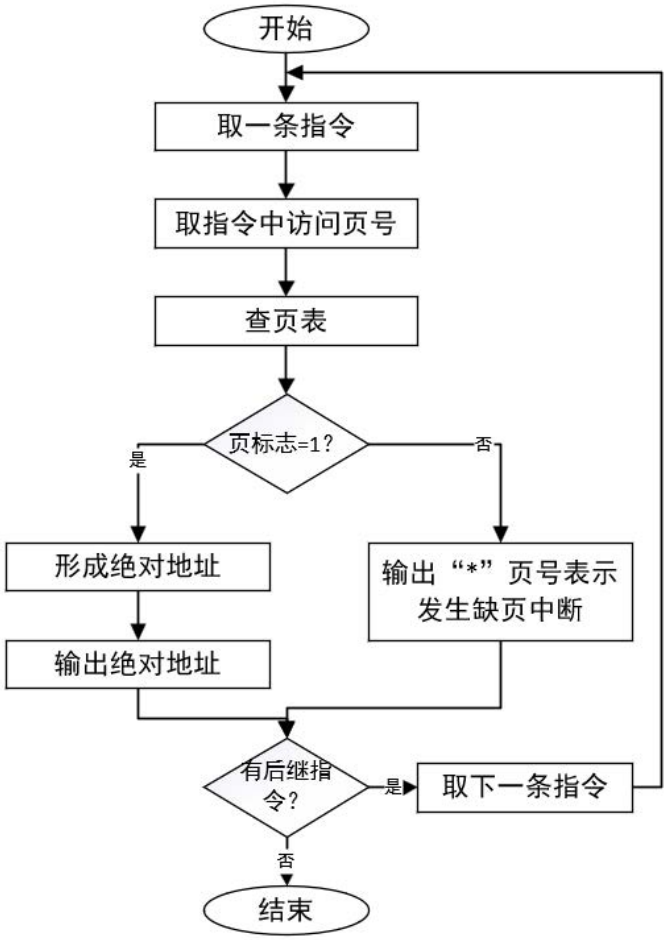


图 3-1　地址转换模拟算法

第二题：用先进先出（FIFO）页面调度算法处理缺页中断。

【提示】

（1）在分页式虚拟存储系统中，当硬件发出“缺页中断”后，引出操作系统来处理这个中断事件。如果主存中已经没有空闲块，则可用 FIFO 页面调度算法把该作业中最先进入主存的一页调出，存放到磁盘上。然后再把当前要访问的页装入该块。调出和装入后都要修改页表中对应页的标志。

（2）FIFO 页面调度算法总是淘汰该作业中最先进入主存的那一页，因此可以用一个数组来表示该作业已在主存的页面。假定作业被选中时，把开始的 m 个页面装入主存，则数组的元素可定为 m 个。例如：

P[0]，P[1]…，P[m-1]

其中，每一个 P[i] (I=0，1，…，m-1) 表示一个在主存中的页面号。它们的初值为：

P[0]：=0，P[1]：=1，…，P[m-1]：=m-1

用一指针 K 指示当要装入新页时，应淘汰的页在数组中的位置，K 的初值为“0”。

当产生缺页中断后，操作系统选择 P[k]所指出的页面调出，然后执行：

P[k]：=要装入页的页号

k：= (k+1) mod m

再由装入程序把要访问的一页信息装入到主存中。重新启动刚才那条指令执行。

（3）编制一个 FIFO 页面调度程序，为了提高系统效率，如果应淘汰的页在执行中没有

修改过，则可不必把该页调出（因在磁盘上已有副本）而直接装入一个新页将其覆盖。因此在页表中增加是否修改过的标志，为“1”表示修改过，为“0”表示未修改过，格式为：

页号	标志	主存块号	修改标志	在磁盘上的位置

由于是模拟调度算法，所以，不实际地启动调出一页和装入一页的程序，而用输出调出的页号和装入的页号来代替一次调出和装入的过程。

把第一题中程序稍作改动，与本题结合起来，FIFO 页面调度模拟算法如图 3-2。

（4）如果一个作业的副本已在磁盘上，在磁盘上的存放地址以及已装入主存的页和作业依次执行的指令序列都同第一题中（4）所示。于是增加了“修改标志”后的初始页表为：

页号	标志	主存块号	修改标志	在磁盘上的位置
0	1	5	0	011
1	1	8	0	012
2	1	9	0	013
3	1	1	0	021
4	0		0	022
5	0		0	023
6	0		0	121

按依次执行的指令序列，运行你所设计的程序，显示或打印每次调出和装入的页号，以及执行了最后一条指令后的数组 P 的值。

（5）为了检查程序的正确性，可再任意确定一组指令序列，运行设计的程序，核对执行的结果。

第三题：用最近最少用（LRU）页面调度算法处理缺页中断。

【提示】

（1）在分页式虚拟存储系统中，当硬件发出“缺页中断”后，引出操作系统来处理这个中断事件。如果主存中已经没有空闲块，则可用 LRU 页面调度算法把该作业中距现在最久没有被访问过的一页调出，存放到磁盘上。然后再把当前要访问的页装入该块。调出和装入后都要修改页表中对应页的标志。

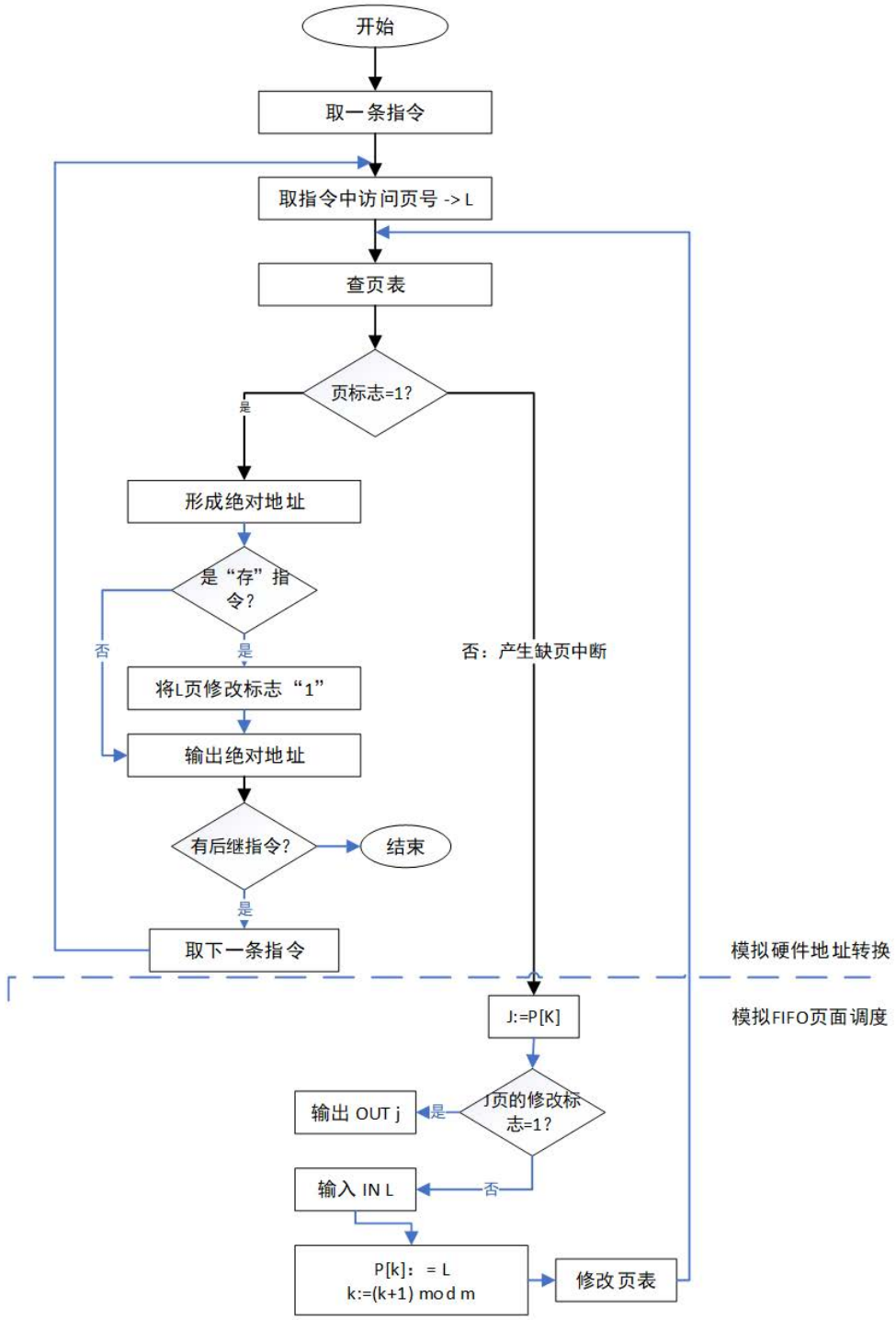


图 3-2　FIFO 页面调度模拟算法

（2）LRU 页面调度算法总是淘汰该作业中距现在最久没被访问过的那页，因此可以用一个数组来表示该作业已在主存的页面。数组中的第一个元素总是指出当前刚访问的页号，因此最久没被访问过的页总是由最后一个元素指出。如果主存只有四块空闲块且执行第一题中提示（4）假设的指令序列，采用 LRU 页面调度算法，那么在主存中的页面变化情况如下：

3	0	6	4	5	1	2	4	6
2	3	0	6	4	5	1	2	4
1	2	3	0	6	4	5	1	2
0	1	2	3	0	6	4	5	1

当产生缺页中断后，操作系统总是淘汰由最后一个元素所指示的页，再把要访问的页装入淘汰页所占的主存块中，页号登记到数组的第一个元素中，重新启动刚才那条指令执行。

（3）编制一个 LRU 页面调度程序，为了提高系统效率，如果淘汰的页在执行中没有修改过，则可不必把该页调出。参看第二题中提示（3）。模拟调度算法不实际地启动调出一页和装入一页的程序而用输出调出的页号和装入的页号来代替。把第一题中程序稍作改动，与本题结合起来，LRU 页面调度模拟算法如图 3-3。

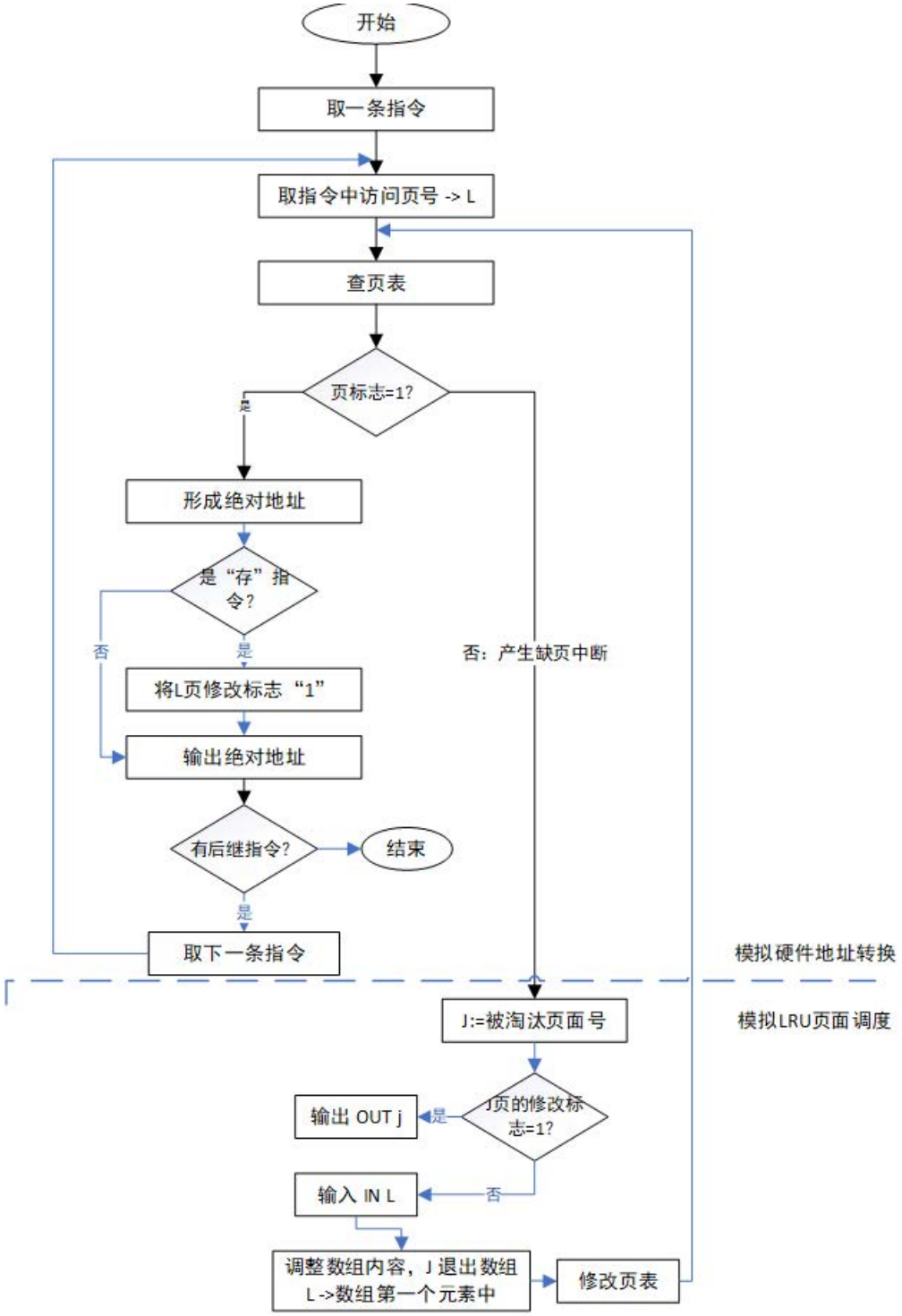


图 3-3　LRU 页面调度模拟算法

（4）按第一题中提示（4）的要求，建立一张初始页表，页表中为每一页增加“修改标志”位（参考第二题中提示（4））。然后按依次执行的指令序列，运行设计的程序，显示或打印每次调出和装入的页号，以及执行了最后一条指令后数组中的值。

（5）为了检查程序的正确性，可再任意确定一组指令序列，运行设计的程序，核对执行的结果。

第四题：用高级语言编写和调试一个简单的文件系统，模拟文件管理的工作过程。

要求设计一个 n 个用户的简单二级文件系统，每次用户可保存 m 个文件，用户在一次运行中只能打开一个文件，对文件必须设置保护措施。要求做到以下几点：

1.可以实现下列几条命令（至少 4 条）：login 用户登录；dir 列文件目录；create 创建文件；delete 删除文件；open 打开文件；close 关闭文件；read 读文件；write 写文件。

2.列目录时要列出文件名、物理地址、保护码和文件长度。

3.源文件可以进行读写保护。

【提示】

（1）设计一个 10 个用户的文件系统，每次用户可保存 10 个文件，一次运行用户可以打开 5 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（UED）。另外，为打开文件设置了运行文件目录（AFD）。

（3）为了便于实现，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）算法与框图

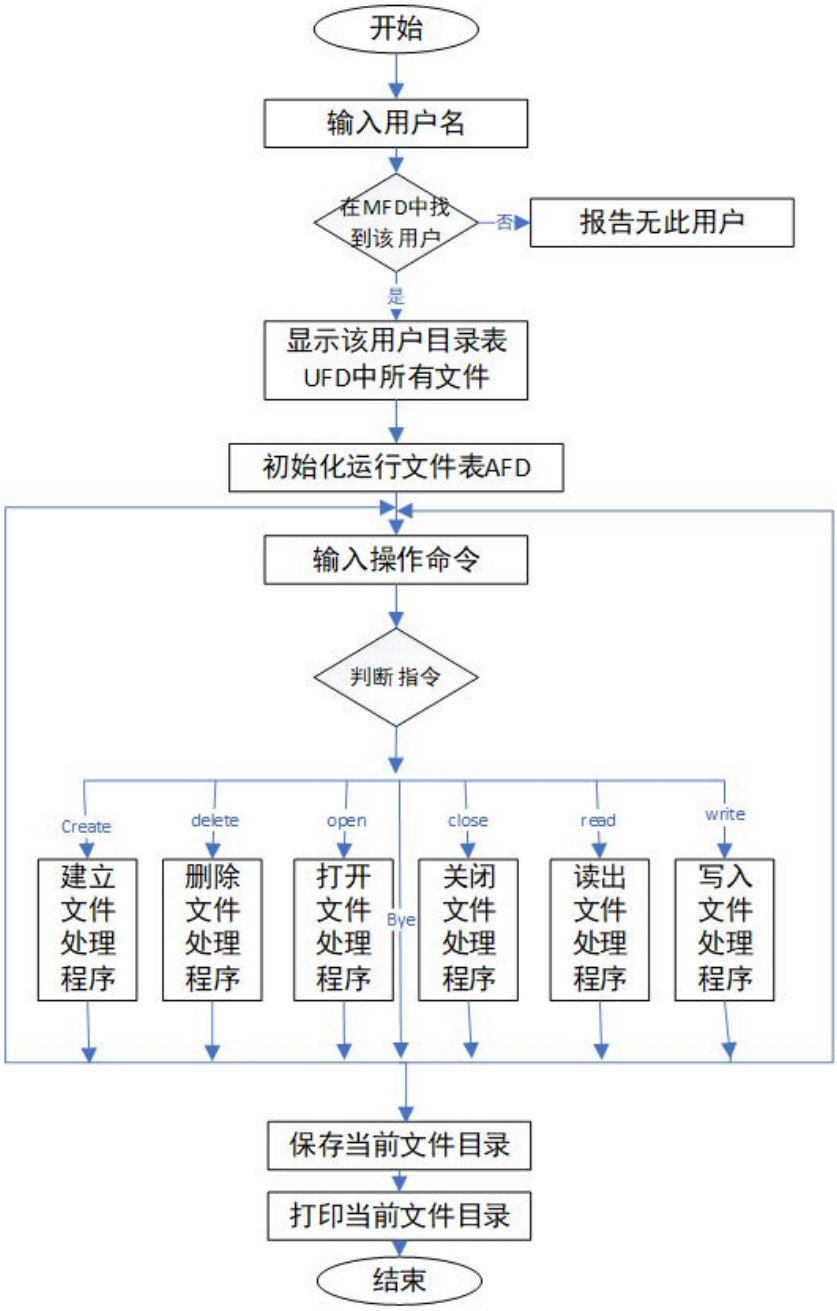
- 因系统小，文件目录的检索使用了简单的线性搜索。
- 文件保护简单使用了三位保护码：允许读写执行、对应位为 1，对应位为 0，则表示不允许读写、执行。
- 程序中使用的主要设计结构如下：
  - 主文件目录和用户文件目录（ MFD、UFD）
  - 打开文件目录（ AFD）（即运行文件目录）

MDF
用户名
文件目录指针
用户名
文件目录指针
……

UFD
文件名
保护码
文件长度
文件名
……

AFD
打开文件名
打开保护码
读写指针
……

文件系统算法的流程图如下：



（5）编一个通过屏幕选择命令的文件管理系统，每屏要为用户提供足够的选择信息，不需要打入冗长的命令。

（6）设计一个树型目录结构的文件系统，其根目录为 root，各分支可以是目录，也可以是文件，最后的叶子都是文件。

五、思考题

（1）先进先出页面调度算法的思想？

（2）最近最少用（LRU）页面调度算法思想？

（3）比较两种调度算法的效率（哪种调度算法使产生缺页中断的次数少）？

（4）分析在什么情况下采用哪种调度算法更有利？

（5）文件管理和目录管理的思想？

六、实验报告

（1）实验题目（第二题或第三题）。

（2）程序中使用的数据结构及符号说明。

（3）实现的核心代码。

（4）打印初始页表、每次调出（要调出一页时）和装入的页号、执行最后一条指令后在主存中的页面号（即数组的值）。

（5）文件管理的基本操作界面。

参考资料

（1）《操作系统概念》（第六版），Abraham Silberschatz 著，高等教育出版社，2004-1。
（2）《计算机操作系统》（修订版），汤子瀛等编，西安电子科技大学出版社，2001-8。
（3）《<计算机操作系统>学习指导与题解》,汤子瀛等编，西安电子科技大学出版社,2003-3。