



**实 验 报 告**

课程名称 操作系统实验

学生学院 计算机学院

专业班级 计算机科学与技术4班

学 号\_\_\_3120005057\_

学生姓名 陈嘉浩

指导教师 林穗

22 年 6 月 12 日

**实验一 进程调度**

一、实验目的

编写并调试一个模拟的进程调度程序，以加深对进程的概念及进程调度算法的理解．

二、实验内容

1. 调试运行“时间片轮转”调度算法，给出运行结果。
2. 采用“时间片轮转”调度算法对进程进行调度。每个进程有一个进程控制块（ PCB）表示。进程控制块可以包含如下信息：进程名、到达时间、需要运行时间、已用CPU时间、进程状态等等。
3. 每个进程的状态可以是就绪 W（Wait）、运行R（Run）、或完成F（Finish）三种状态之一。 显示进程运行过程，以及进程的带权周转时间和系统的平均带权周转时间。

三、实现思路

时间片轮转：按照各进程到达就绪队列的顺序，轮流让各个进程执行一个时间片。若进程未在一个时间片内执行完，则剥夺处理机，将进程重新放到就绪队列队尾重新排队；若进程未达到一个时间片时间就完成，则立即激活调度程序，将它从就绪队列中删除，再调度就绪队列中队首的进程运行，并开启新的时间片。

短作业优先算法：每次调度时选择当前已到达且运行时间最短的作业/进程。

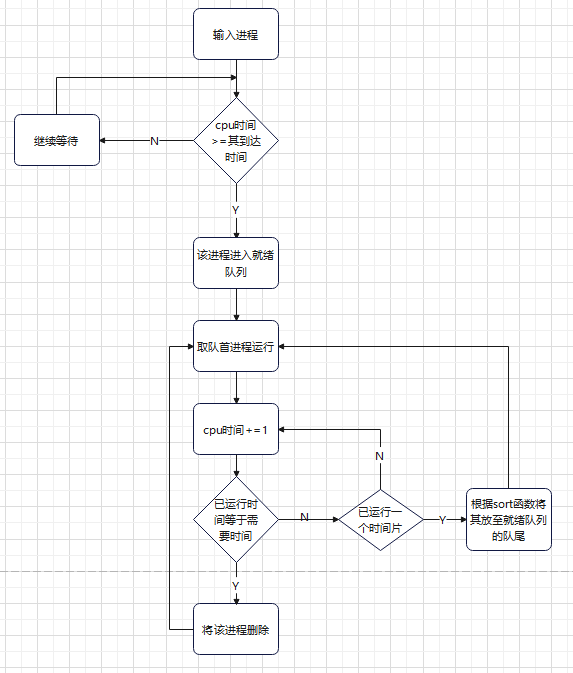
高响应比算法：每次调度时计算所有就绪进程的响应比，选响应比最高的进程。响应比=(等待时间+要求服务时间)/要求服务时间

四、主要的数据结构

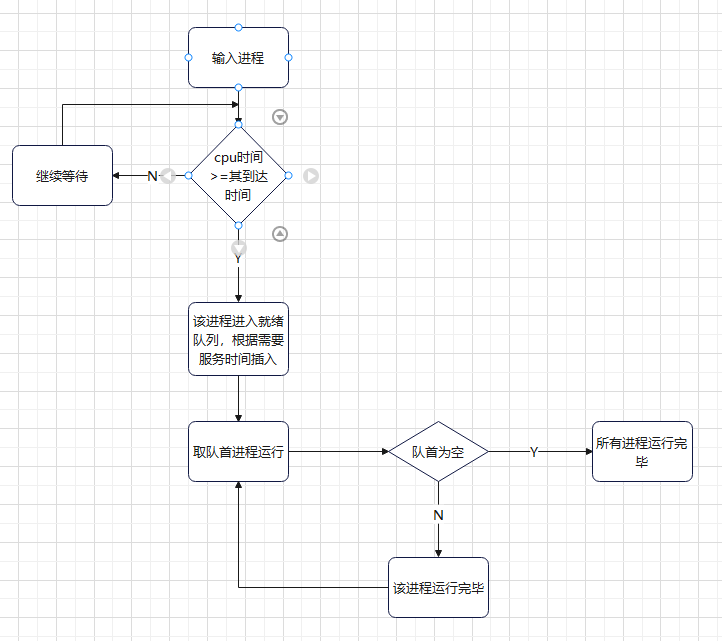
PCB，定义了进程名，其运行状态，到达时间，需要时间，已运行时间，(优先级)。

五、算法流程图

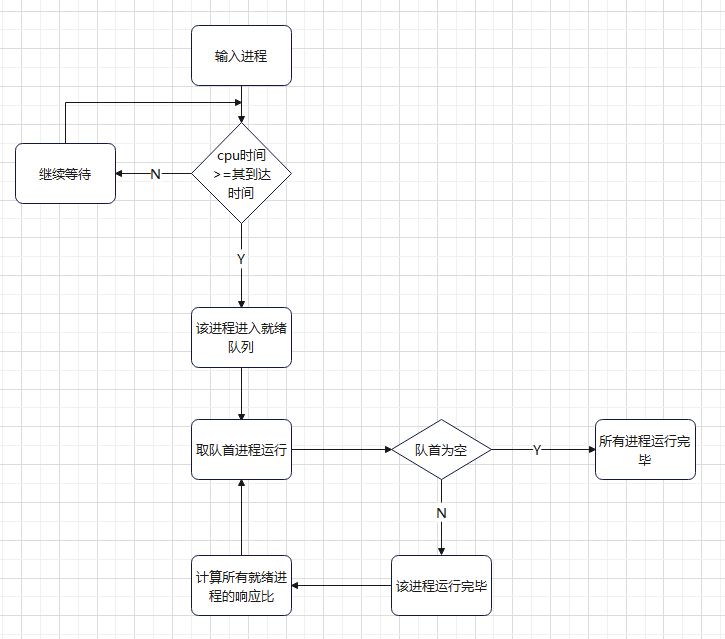
时间片轮转：



短时间优先：

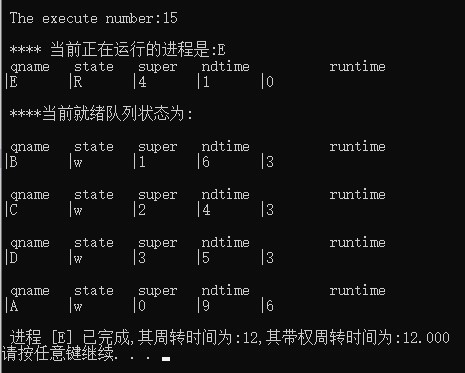


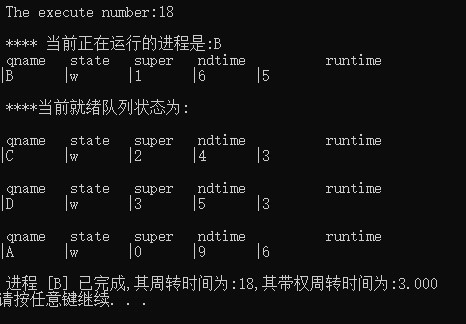
高响应比：



六、运行与测试*（系统运行截图）*

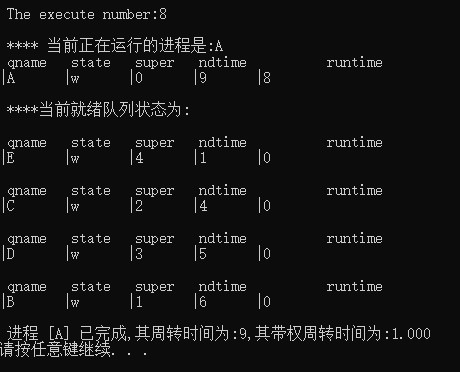
时间片轮转：

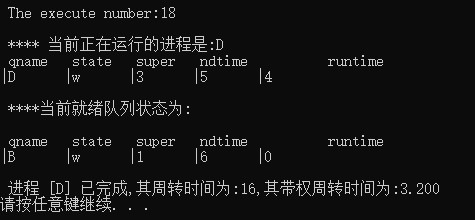


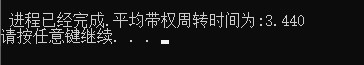




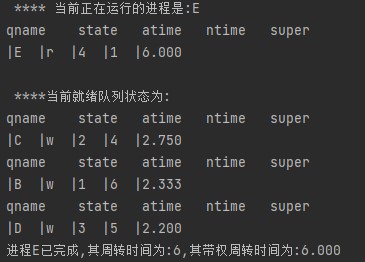
短时间优先：

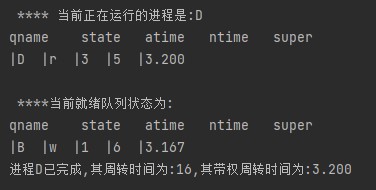


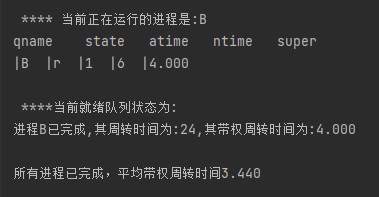




高响应比：







七、总结

时间片轮转：相比短作业优先算法，时间片轮转算法有效地缓解了长作业的饥饿现象，但其做了一个隐含的假设，即所有系统中的所有进程的紧迫性是相同的，但实际情况并非如此。

短时间优先：对长作业不利，可能导致产生饥饿现象。未考虑作业的紧迫程度。

高响应比：综合考虑进程的等待时间以及需要服务时间。故算法实现了较好的折中。但是，该算法在每次进行调度之前都要先做响应比的计算，显然会增加系统开销。

实验二 动态分区分配方式的模拟

一、实验目的

了解动态分区分配方式中的数据结构和分配算法，并进一步加深对动态分区存储管理方式及其实现过程的理解

二、实验内容

1. 用C语言分别实现采用首次适应算法和最佳适应算法的动态分区分配过程和回收过程。其中，空闲分区通过空闲分区链（表）来管理；在进行内存分配时，系统优先使用空闲区低端的空间。
2. 假设初始状态下，可用的内存空间为640KB，并有下列的请求序列：

•作业1申请130KB

•作业2申请60KB

•作业3申请100KB

•作业2释放60KB

•作业4申请200KB

•作业3释放100KB

•作业1释放130KB

•作业5申请140KB

•作业6申请60KB

•作业7申请50KB

•作业8申请60KB

请分别采用首次适应算法和最佳适应算法进行内存的分配和回收，要求每次分配和回收后显示出空闲内存分区链的情况。

三、实现思路

首次适应算法：空闲分区以地址递增的次序排列。每次分配内存时顺序查找空闲分区表，找到大小能满足要求的第一个空闲分区。

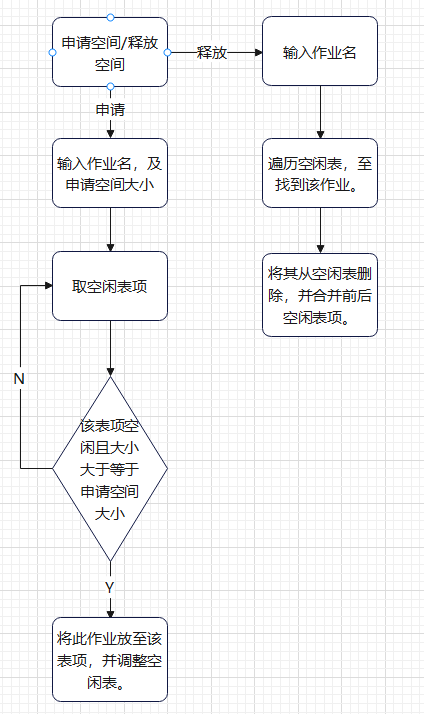
最佳适应算法：空闲分区按容量递增的次序排列。每次分配内存时顺序查找空先分区表，找到大小能满足要求的第一个空闲分区。

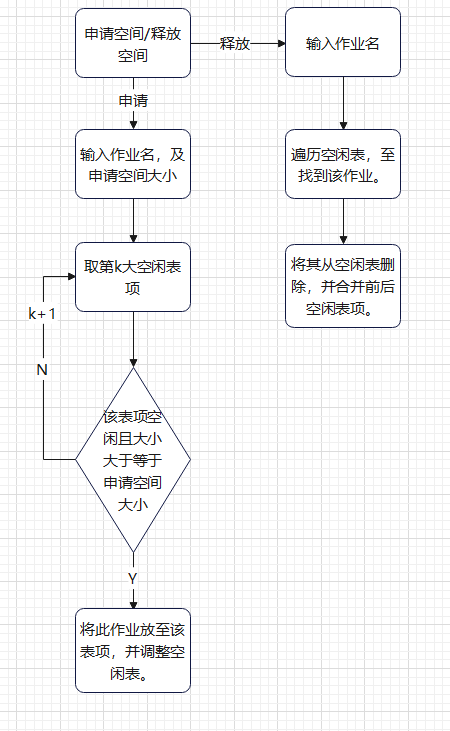
四、主要的数据结构

空闲表项， 定义了分区的大小、分区的起始地址以及分区的状态。

五、算法流程图

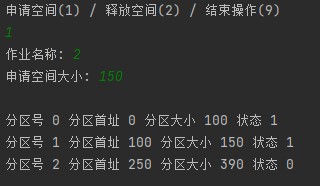
首次适应算法：

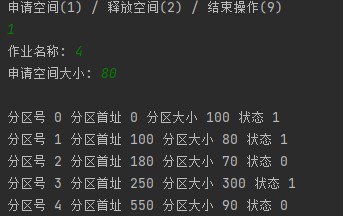
  
最佳适应算法：



六、运行与测试

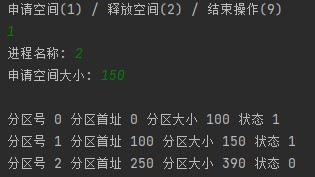
首次适应算法：

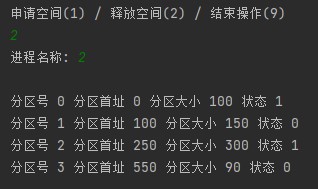


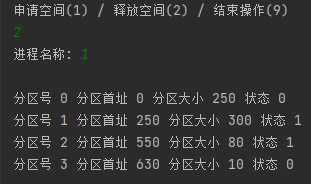




最佳适应算法：







七、总结

首次适应算法：该算法倾向于优先利用内存中低址部分的空闲分区，从而保留了高址部分的大空闲区。这为以后到达的大作业分配大的内存空间创造了条件。其缺点是低址部分不断被划分，会留下许多难以利用的、很小的空闲分区，称为碎片。而每次查找又都是从低址部分开始的，这无疑又会增加查找可用空闲分区时的开销。

最佳适应算法：狐立地看，最佳适应算法似乎是最佳的，然而在宏观上却不一定。因为每次分配后所切割下来的剩余部分总是最小的，这样，在存储器中会留下许多难以利用的碎片。

**实验三 请求调页存储管理方式的模拟**

**一、实验目的**

**通过对页面、页表、地址转换和页面置换过程的模拟，加深对请求调页系统的原理和实现过程的理解。**

**二、实验内容**

**（1）假设每个页面中可存放10条指令，分配给作业的内存块数为4。**

**（2）用c语言模拟一个作业的执行过程，该作业共有320条指令，即它的地址空间为32页，目前它的所有页都还未调入内存。在模拟过程中，如果所访问的指令已在内存，则显示其物理地址，并转下一条指令。如果所访问的指令还未装入内存，则发生缺页，此时需记录缺页的次数，并将相应页调入内存。如果4个内存块均已装入该作业，则需进行页面置换，最后显示其物理地址，并转下一条指令。在所有320指令执行完毕后，请计算并显示作业运行过程中发生的缺页率。**

**（3）置换算法：采用先进先出（FIFO）、最近最久未使用（LRU）和最佳置换（OPT）算法置换算法。**

三、实现思路

最佳置换算法：每次选择淘汰的页面将是以后永不使用，或者在最长时间内不再被访问的页面，这样可以保证最低的缺页率。

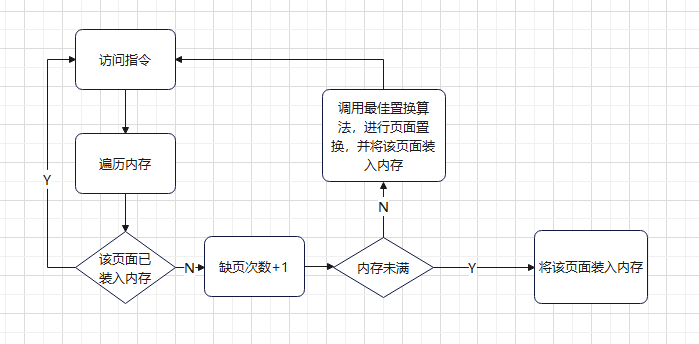
LRU：每次淘汰的页面是最近最久未使用的页面。

四、主要的数据结构

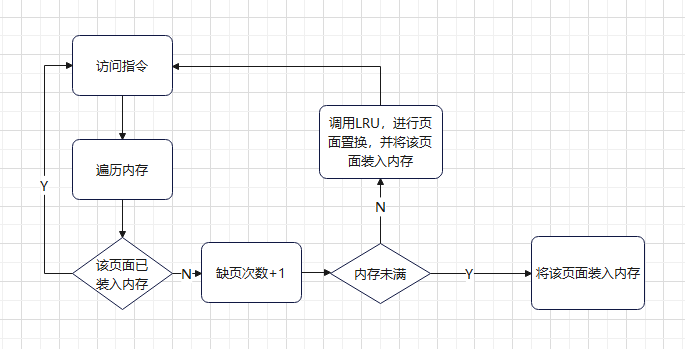
内存卡Block，存储指令对应的页面。

五、算法流程图

最佳置换算法：

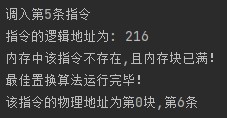


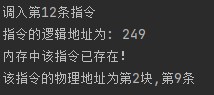
LRU:



六、运行与测试

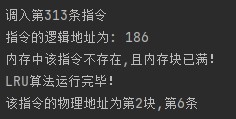
最佳置换算法

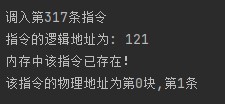






LRU：







七、总结

最佳置换算法可以保证最低的缺页率，但是操作系统并不能提前预判页面访问序列，故最佳置换算法是无法实现的。LRU算法性能虽然好，但是需要专门的硬件支持，实现困难，开销大。

拓展实验

银行家算法

1. 实验目的：分析操作系统的核心功能模块，理解相关功能模块实现的数据结构和算法，并加以实现，加深对操作系统原理和实现过程的理解。
2. 实验内容：

银行家算法：在进程提出资源申请时，先预判此次分配是否会导致系统进入不安全状态，如果会进入不安全状态，就暂时不答应这次请求，让该进程先阻塞等待。

三、实现思路

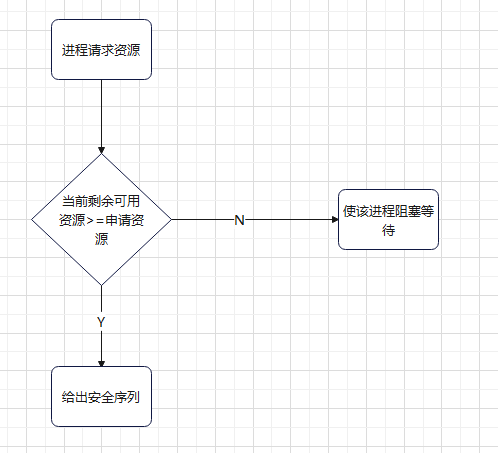
银行家算法：进程提出资源申请后，与当前剩余可用资源对比，若不安全则不答应。否则，找出一个安全序列。

四、主要的数据结构

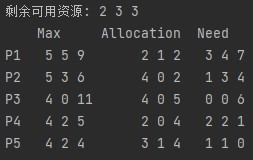
银行家算法：Max数组表示进程的最大需求，Allocation表示系统对各个进程已分配的资源，Need表示各个进程还需要多少资源。

五、算法流程图

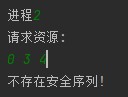
银行家算法：



六、运行与测试







1. 总结

银行家算法：使用该算法，系统可用检测此次资源分配后，系统是否处于安全状态，有效避免的死锁。

磁盘调度算法

一、实验目的：分析操作系统的核心功能模块，理解相关功能模块实现的数据结构和算法，并加以实现，加深对操作系统原理和实现过程的理解。

1. 实验内容：

通过给出系统当前I/O请求队列，计算出所使用算法的磁头寻道顺序及平均寻道长度。

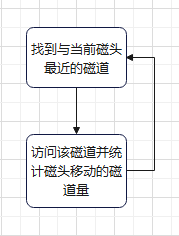
1. 实现思路

SSTF:优先处理的磁道是当前最近的磁道。

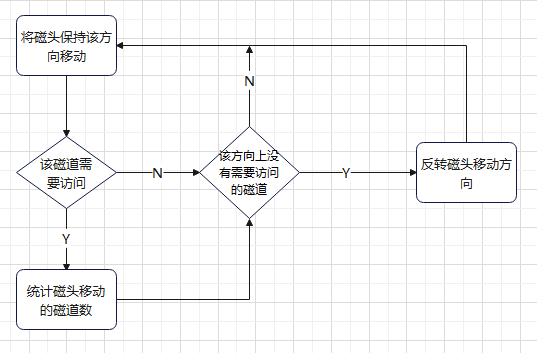
SCAN:磁头根据当前磁头移动方向继续移动，直至该方向上没有磁道需要访问，则朝相反的方向继续移动，直至该方向上没有磁道需要访问。

1. 主要的数据结构
2. 算法流程图

SSTF：

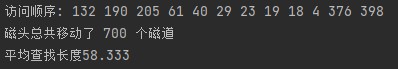


SCAN:



1. 运行与测试

SSTF：



SCAN：



七、总结

SSTF性能较好，平均寻道时间短，但是可能产生饥饿现象。

SCAN不会产生饥饿现象，但是对于各个位置磁道的响应频率不平均。

多级反馈队列调度算法

一、实验目的：分析操作系统的核心功能模块，理解相关功能模块实现的数据结构和算法，并加以实现，加深对操作系统原理和实现过程的理解。

1. 实验内容：

实现多级反馈队列调度算法对进程实现调度。

1. 实现思路

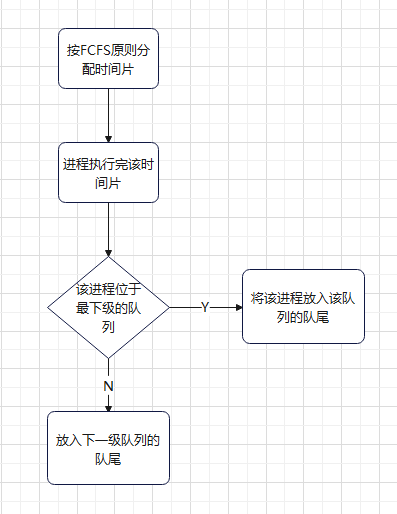
设置多级就绪队列，各级队列优先级从高到低，时间片设置为1， 2， 4， …新进程到达时先进入第一级队列，按FCFS原则排队等待分配时间片。只有当第k级队列为空时，才会为第k+1级队头的进程分配时间片。

1. 主要的数据结构

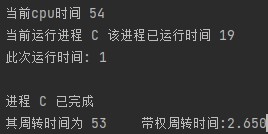
队列，存储PCB

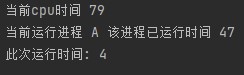
PCB，包含进程的相关信息

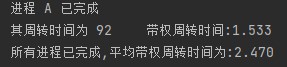
1. 算法流程图



1. 运行与测试







七、总结

多级反馈队列调度算法对各类型进程相对公平。

位示图

一、实验目的：分析操作系统的核心功能模块，理解相关功能模块实现的数据结构和算法，并加以实现，加深对操作系统原理和实现过程的理解。

1. 实验内容：实现位示图对磁盘块进行分配以及回收。
2. 实现思路

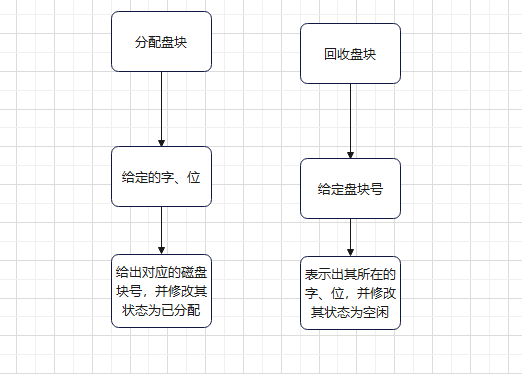
每个二进制位对应一个盘块，1代表空闲，0代表盘块已分配。b = n \* i + j+ 1

i = (b-1) // n ，j = (b-1) % n

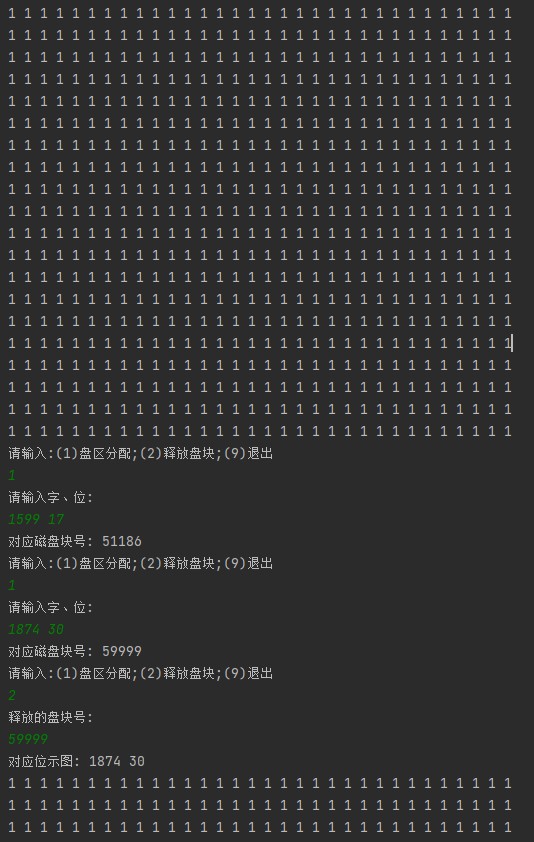
1. 主要的数据结构

位示图：存储各个磁盘块的使用情况

1. 算法流程图



1. 运行与测试



七、总结

注意盘块号，字，位是从1开始，还是从0开始。

记录型信号量解决生产者-消费者问题

一、实验目的：分析操作系统的核心功能模块，理解相关功能模块实现的数据结构和算法，并加以实现，加深对操作系统原理和实现过程的理解。

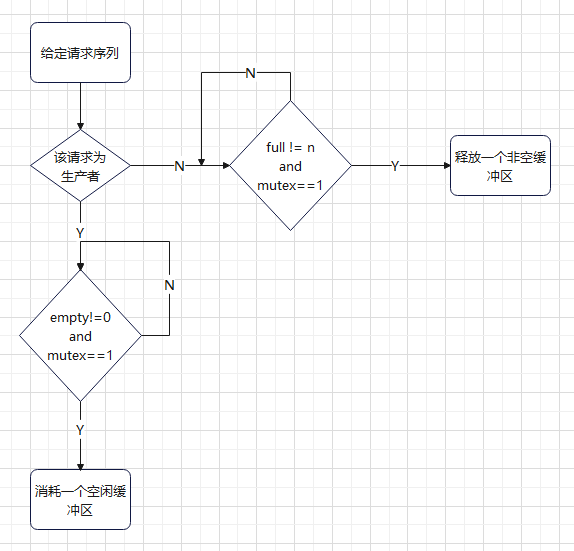
二、实验内容：实现一次只有一个生产者能使用缓冲区，缓冲区满时，生产者必须等待，缓冲区空时，消费者必须等待。

三、实现思路：设置互斥信号量mutex实现对缓冲区的互斥访问，设置empty、full两个信号量分别表示空闲缓冲区的数量、非空缓冲区的数量。

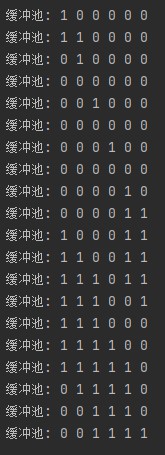
四、主要的数据结构

Buffer数组，表示缓冲池。

1. 算法流程图



1. 运行与测试



七、总结

实现各进程对临界资源互斥访问。