

电子科技大学

计算机专业类课程

实验报告

课程名称：操作系统

学 院：计算机科学与工程学院

专 业：计算机科学与技术

学生姓名：张爽

学 号：2013060109032

指导教师：薛瑞尼

日 期：2016 年 6 月 10 日

电子科技大学

实验报告

实验一

一、实验名称：生产者消费者问题

二、实验学时：4

三、实验内容和目的：

掌握生产者消费者问题的原理、思想核心和基本场景的解决方法。编程实现以下条件两个场景：共享缓冲区中放置一个数字，取值范围为[0, 10]，初值为 0。生产者将此值加 1，消费者将此值减 1。

场景 1

- * 同一进程内启动一组生产者线程和一组消费者线程
- * 缓冲区为本进程的全局变量

场景 2

- * 启动一组生产者进程和一组消费者进程
- * 同一个数据文件为缓冲区
- * 输入
 - * `p`：生产者数量
 - * `c`：消费者数量
- * 输出

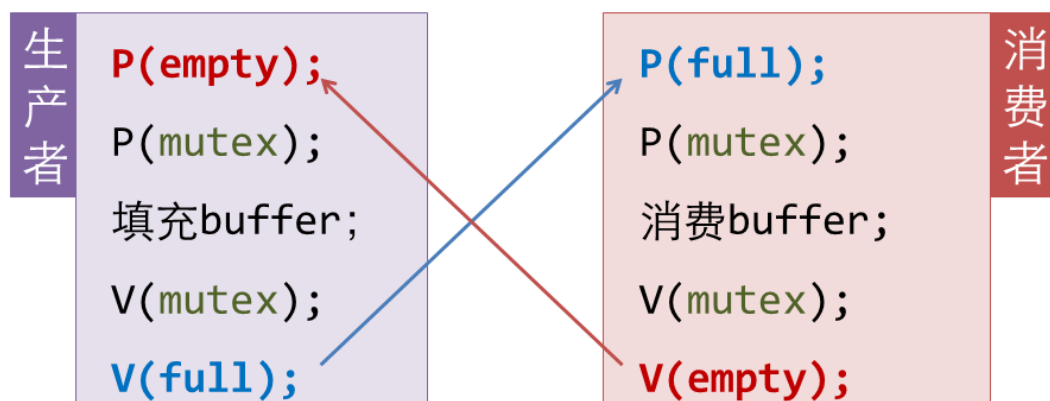
打印当前共享缓冲区中的数值，或者生产者消费者的状态。

四、实验原理：

生产者消费者问题是一个很经典的同步与互斥的问题。对于生产者：满则等待，空则填充；对于消费者：空则等待，有则获取。二者不允许同时进入缓冲区。既存在合作同步问题，也存在临界区互斥问题：1、合作同步：当缓冲池全满时，表示供过于求，生产者必须等待，同时唤醒消费者；当缓冲池全空时，表示供不应求，消费者应等待，同时唤醒生产者。2、互斥：缓冲池显然是临界资源，所在生产者与消费都要使用它，而且都要改变它的状态。

指导 PV 操作语句：

- semaphore full = 0 → “满”缓冲区数量
- semaphore empty = N → “空”缓冲区数量
- semaphore mutex = 1 → 互斥访问缓冲区



五、实验器材（设备、元器件）

本实验所需的硬件主要有：PC 计算机，具体要求如下：

软件环境

1. 操作系统：Windows 7；
2. 开发平台：VC2013；
3. 工具语言：C 语言

六、实验步骤：

场景 1 此次实验要实现同步与互斥故调用了 `pthread` 库。

- 1、 建立缓冲区，设好最小最大值，设立生产者、消费者、`mutex` 锁的线程标示符。
- 2、 两个函数：在生产者 `producer` 中， 先确立缓冲区中没填满，进行加 1 操作，否则等待；在消费者 `consumer` 中，先确定缓冲区中有数，即大于 0，进行减一操作，否则等待。
- 3、 主函数：输入生产者和消费者的数量，拉起两个线程，按照生产者和消费者的函数规定，随机的指定生产者和消费者进入缓冲区进行操作。此时是生产者消费者的是一直运行的，直到外部终止程序。

场景 2 运用信号量

- 1、 建立缓冲区，设好最小最大值，两个信号量 `empty`, `full`，设立 `mutex` 锁。
- 2、 两个生产者消费者的缓冲区函数：在生产者 `producer` 的缓冲区中， 先确立缓冲区中没填满，进行加 1 操作，否则等待；在消费者 `consume` 缓冲区中，先确定缓冲区中有数，即大于 0，进行减一操作，否则等待。
- 3、 **PV 操作**：根据 **PV** 操作争取 `mutex` 锁，拿到才能进行加减操作。
- 4、 主函数：清空缓冲区和销毁已存在的信号量，用户输入生产者和消费者数量，初始化信号量；利用 `fork` 语句创建生产者进程，用 `pid` 判断如果是子进程，则开始生产，此时是否需要等待是由操作系统控制。消费者也类似，创建消费者进程，用 `pid` 判断如果是子进程，则开始消费。主控程序最后会退出。

七、实验数据及结果分析：

```
please input p and c: 3 1
Producer 0: 0->1
Producer 1: 1->2
Producer 2: 2->3
Consumer 0: 3->2
Producer 0: 2->3
Consumer 0: 3->2
Producer 2: 2->3
Producer 1: 3->4
Producer 1: 4->5
Producer 2: 5->6
Consumer 0: 6->5
Producer 0: 5->6
Producer 0: 6->7
Consumer 0: 7->6
Producer 1: 6->7
Producer 2: 7->8
Producer 0: 8->9
Producer 2: 9->10
Producer 1: waiting
Consumer 0: 10->9
Producer 1: resume
Producer 1: 9->10
Producer 1: waiting
Consumer 0: 10->9
Producer 0: 9->10
Producer 1: resume
Producer 1: waiting
Producer 2: waiting
```

场景一:

场景二:

```
please input p and c: 1 3
Producer 0: 0->1
Consumer 0: 1->0
Producer 0: 0->1
Consumer 1: 1->0
Producer 0: 0->1
Consumer 0: 1->0
Producer 0: 0->1
Consumer 2: 1->0
Producer 0: 0->1
Consumer 1: 1->0
Producer 0: 0->1
Consumer 2: 1->0
Producer 0: 0->1
Consumer 1: 1->0
Producer 0: 0->1
Consumer 0: 1->0
Producer 0: 0->1
Consumer 2: 1->0
Producer 0: 0->1
Consumer 1: 1->0
```

八、实验结论、心得体会和改进建议:

本次实验是关于生产者和消费者之间互斥和同步的问题。问题的实质是 P, V 操作，实验设一个共享缓冲区，生产者和消费者互斥的使用，当一个线程使用缓冲区的时候，另一个让其等待知道前一个线程释放缓冲区为止。

通过本次实验，我们对操作系统的 P, V 进一步的认识，深入的了解 P, V 操作的实质和其重要性。课本的理论知识进一步阐述了现实的实际问题。

电子科技大学

实 验 报 告

实验二

一、实验名称：银行家算法

二、实验学时：4

三、实验内容和目的：

输入

* `p`: 进程数量

* `r`: 资源数量

* 各进程的 `max`, `allocation`

* 输出

* 若产生死锁，打印提示：`死锁状态`。

* 否则，给出一种调度顺序。

四、实验原理：

银行家算法又称“资源分配拒绝”法，其基本思想是，系统中的所有进程放入进程集合，在安全状态下系统受到进程的请求后试探性的把资源分配给他，现在系统将剩下的资源和进程集合中其他进程还需要的资源数做比较，找出剩余资源能满足最大需求量的进程，从而保证进程运行完成后还回全部资源。这时系统将该进程从进程集合中将其清除。此时系统中的资源就更多了。反复执行上面的步骤，最后检查进程的集合为空时就表明本次申请可行，系统处于安全状态，可以实施本次分配，否则，只要进程集合非空，系统便处于不安全状态，本次不能分配给他。请进程等待。

- 设系统中有 m 种不同资源， n 个进程
 - $\text{Resource}[j]$, j 资源的数量
- Available 向量: 系统中尚未分配的每种资源的总量
 - $\text{Available}[j]$: 尚未分配的资源 j 的数量
- Max 矩阵: 各个进程对每种资源的最大需求量(进程事先声明)
 - $\text{Max}[i, j](\text{Claim}[i, j])$: 进程 i 对资源 j 的最大需求量
- Allocation 矩阵: 当前资源分配状况
 - $\text{Allocation}[i, j]$: 进程 i 获得的资源 j 的数量
- Need 矩阵: 每个进程还需要的剩余资源的数量
 - $\text{Need}[i, j]$: 进程 i 尚需的资源 j 的数量

五、实验器材（设备、元器件）

本实验所需的硬件主要有：PC 计算机，具体要求如下：

软件环境

1. 操作系统：Windows 7;
2. 开发平台：VC2013;
3. 工具语言：C++语言

六、实验步骤:

当进程 p_i 提出资源申请时，系统执行下列步骤: (1) 若 $\text{Request}[i] \leq \text{Need}[i]$, 转 (2); 否则错误返回 (2) 若 $\text{Request}[i] \leq \text{Available}$, 转 (3); 否则进程等待(3) 假设系统分配了资源，则有：

$\text{Available} := \text{Available} - \text{Request}[i];$

$Allocation[i] := Allocation[i] + Request[i];$

$Need[i] := Need[i] - Request[i]$

若系统新状态是安全的，则分配完成若系统新状态是不安全的，则恢复原状态，进程等待。

1、在程序中，首先实现将序列扫描，如果 $Request \leq Need$ ，则转向 2；否则，出错；如果 $Request \leq Available$ ，则转向 3，否则等待；系统试探分配请求的资源给进程；系统执行安全性算法

2、判断序列是否安全：在此处设置了两个向量：Work=Available 和 Finish：初始化为 False 若 $Finish[i]=False \& \& Need \leq Work$ ，则执行 3；否则执行 4 (I 为资源类别) 进程 P 获得第 i 类资源，则顺利执行直至完成。

释放资源： $Work = Work + Allocation$ ； $Finish[i] = true$ ；若所有进程的 $Finish[i] = true$ ，则表示系统安全；否则，不安全。

七、实验结论、心得体会和改进建议：

安全序列下：

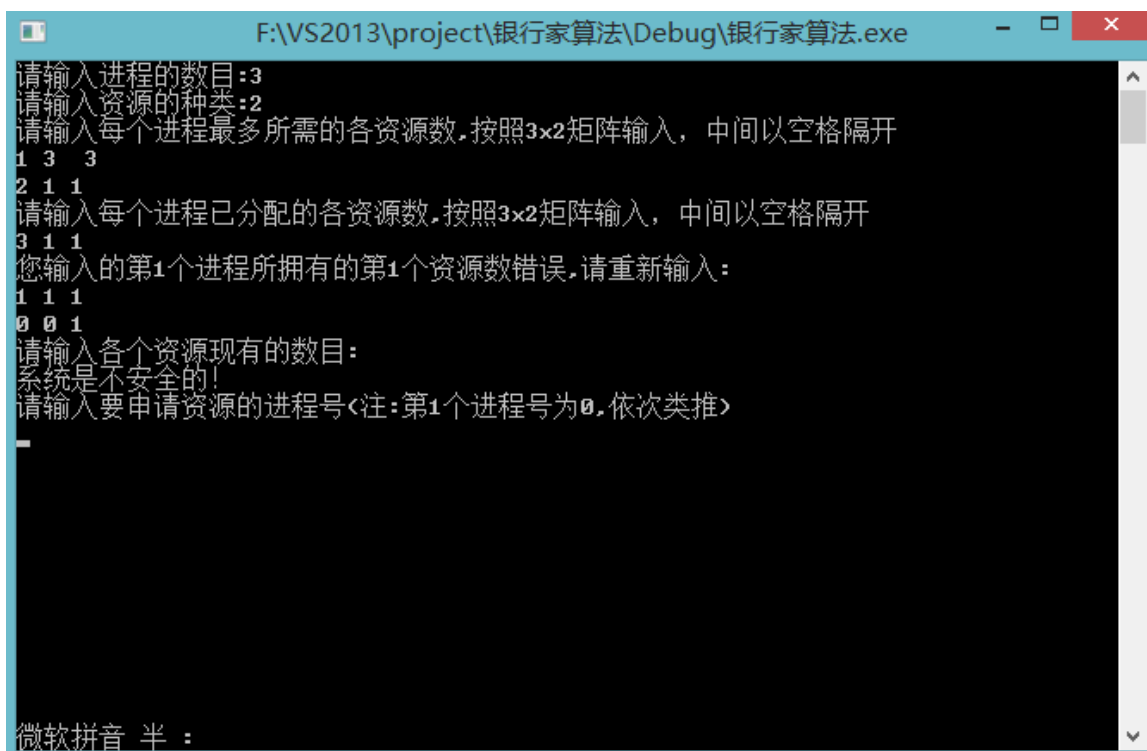
```
C:\WINDOWS\system32\cmd.exe

请输入进程的数目:3
请输入资源的种类:2
请输入每个进程最多所需的各资源数,按照3x2矩阵输入,中间以空格隔开
3 2 2
4 2 3
请输入每个进程已分配的各资源数,按照3x2矩阵输入,中间以空格隔开
0 0 1
1 0 0
请输入各个资源现有的数目:
10
5
系统安全!
安全序列:
0-->1-->2
请输入要申请资源的进程号<注:第1个进程号为0,依次类推>
0
请输入进程所请求的各资源的数量
2
2
系统安全!
安全序列:
0-->1-->2
同意分配请求!
若想继续请求分配,请按y
微软拼音 半 :
```

```
C:\WINDOWS\system32\cmd.exe

y
请输入要申请资源的进程号<注:第1个进程号为0,依次类推>
1
请输入进程所请求的各资源的数量
1
1
系统安全!
安全序列:
0-->1-->2
同意分配请求!
若想继续请求分配,请按y
y
请输入要申请资源的进程号<注:第1个进程号为0,依次类推>
3
请输入进程所请求的各资源的数量
0
1
您输入的请求数超过进程的需求量!请重新输入!
系统安全!
安全序列:
0-->1-->2
同意分配请求!
若想继续请求分配,请按y
-
```

不安全的序列:



```
F:\VS2013\project\银行家算法\Debug\银行家算法.exe
请输入进程的数目:3
请输入资源的种类:2
请输入每个进程最多所需的各资源数.按照3x2矩阵输入,中间以空格隔开
1 3 3
2 1 1
请输入每个进程已分配的各资源数.按照3x2矩阵输入,中间以空格隔开
3 1 1
您输入的第1个进程所拥有的第1个资源数错误.请重新输入:
1 1 1
0 0 1
请输入各个资源现有的数目:
系统是不安全的!
请输入要申请资源的进程号<注:第1个进程号为0.依次类推>
微软拼音 半:
```

银行家算法是避免死锁的一种重要方法，通过编写一个简单的银行家算法程序，加深了解有关资源申请、避免死锁等概念，并体会和了解死锁和避免死锁的具体实施方法。死锁的产生，必须同时满足四个条件，即一个资源每次只能由一个进程；第二个为等待条件，即一个进程请求资源不能满足时，它必须等待，但它仍继续保持已得到的所有其他资源；第三个为非剥夺条件，即在出现死锁的系统中一定有不可剥夺使用的资源；第四个为循环等待条件，系统中存在若干个循环等待的进程，即其中每一个进程分别等待它前一个进程所持有的资源。防止死锁的机构只能确保上述四个条件之一不出现，则系统就不会发生死锁。

电 子 科 技 大 学

实 验 报 告

实验三

一、 实验名称：页式存储逻辑地址到物理地址映射

二、 实验学时：4

三、 实验内容和目的：

条件：64 位地址空间

* 输入：

* 页记录大小（如 4Byte）

* 页表级数（如，2 表示 2 级页表，n 表示 n 级页表）

* 逻辑地址（十六进制）

* 输出：物理地址（物理块号，块内偏移）

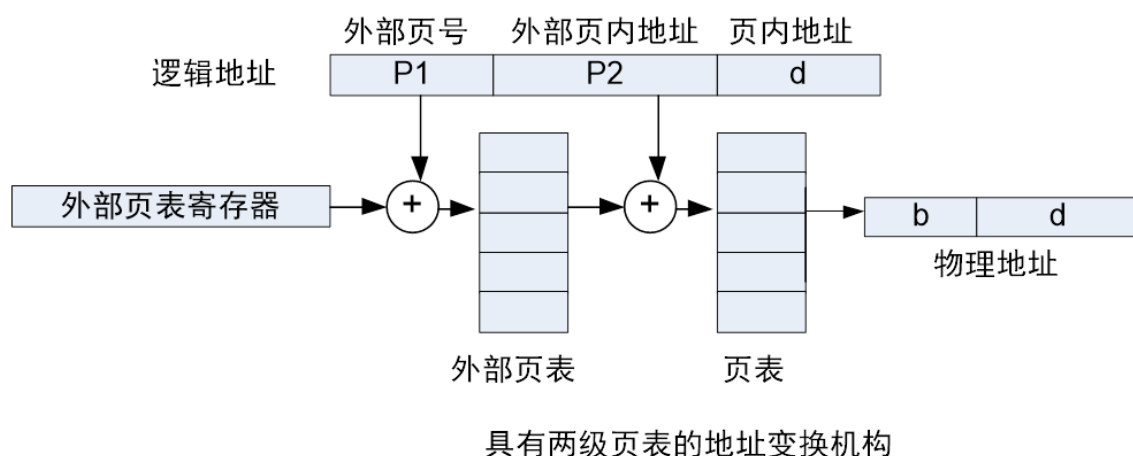
说明：页表随机产生，为便于验证可令逻辑页号 `n` 的物理块号为 `n`。

四、 实验原理：

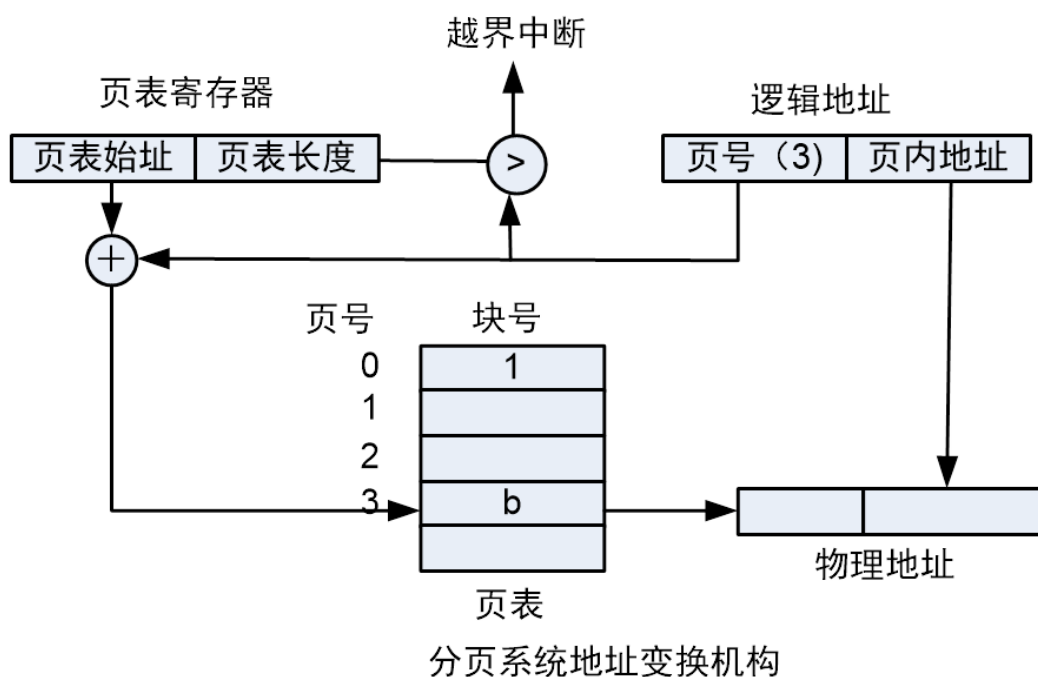
页内地址：长度由页大小决定，是低位部分

页号：除去页内地址所占的高位部分

页表 page table：逻辑页号→物理块号的映射，页表放在内存，属于进程的现场信息。页表也分为两级页表和多级页表，将页表分级是因为若不分级，则会导致页表过大无法连续存储，故要分级，将其离散存放在不同页块中，则能按需调用。两级页表查找过程如下图：



地址变换过程如下图：



五、 实验器材（设备、元器件）

本实验所需的硬件主要有：PC 计算机，具体要求如下：

软件环境

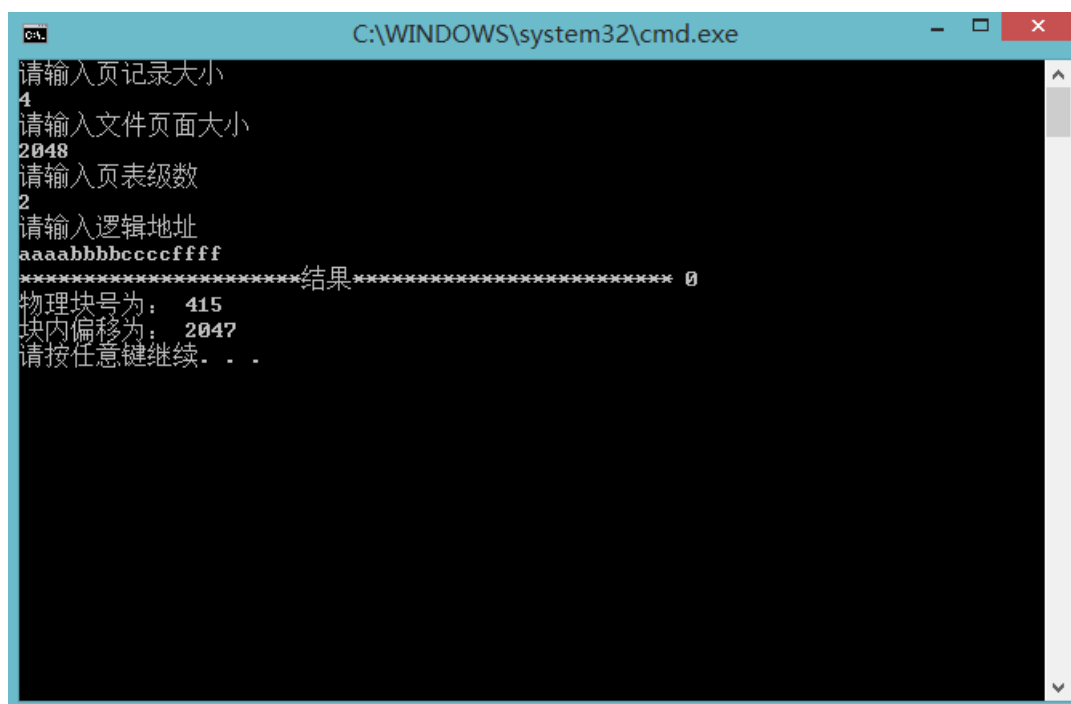
1. 操作系统： Windows 7；

2. 开发平台：VC2013;
3. 工具语言：C++语言

六、实验步骤:

1. 打开 VS2013，新建项目;
2. 实施地址转换的函数（核心函数）：在 64 位地址空间下，要求用户输入后，首先需要做参数检查，判断其是否是 2 的倍数和是否整除；此时我们先不考虑最外层页表，将输入做 2 的开方计算出页内地址位数，再得到每页页表项个数和其在虚拟地址中占的位数。之后再计算最外层页表，在逻辑地址中的位数和其页号。注意：此时为了方便验证，这里假设逻辑页号 n 的物理块号也为 n ，依次计算后续页表的页号，
3. 主函数：要求用户输入页记录大小、页表级数、十六进制的逻辑地址，输出物理地址，即物理块号和块内偏移。

七、实验结论、心得体会和改进建议:



```
C:\WINDOWS\system32\cmd.exe
请输入页记录大小
4
请输入文件页面大小
2048
请输入页表级数
2
请输入逻辑地址
aaaabbbbccccffff
*****结果*****
物理块号为: 415
块内偏移为: 2047
请按任意键继续...
```

通过本次实验我了解了页式存储的逻辑地址到物理地址映射的基本思想和计算方法，页表的概念，两级页表、多级页表的工作方式以及如何进行逻辑地址和物理地址的转换。同时编程实现了，提高了我的动手能力。

电子科技大学

实验报告

实验四

一、 实验名称：混合索引逻辑地址到物理地址映射

二、 实验学时：4

三、 实验内容和目的：

条件：自定义混合索引 `inode` 结构

- * 必须包括一次，二次，和三次间接块

- * 逻辑块 `n` 对应物理块 `n`

- * 输入：文件逻辑地址

- * 输出

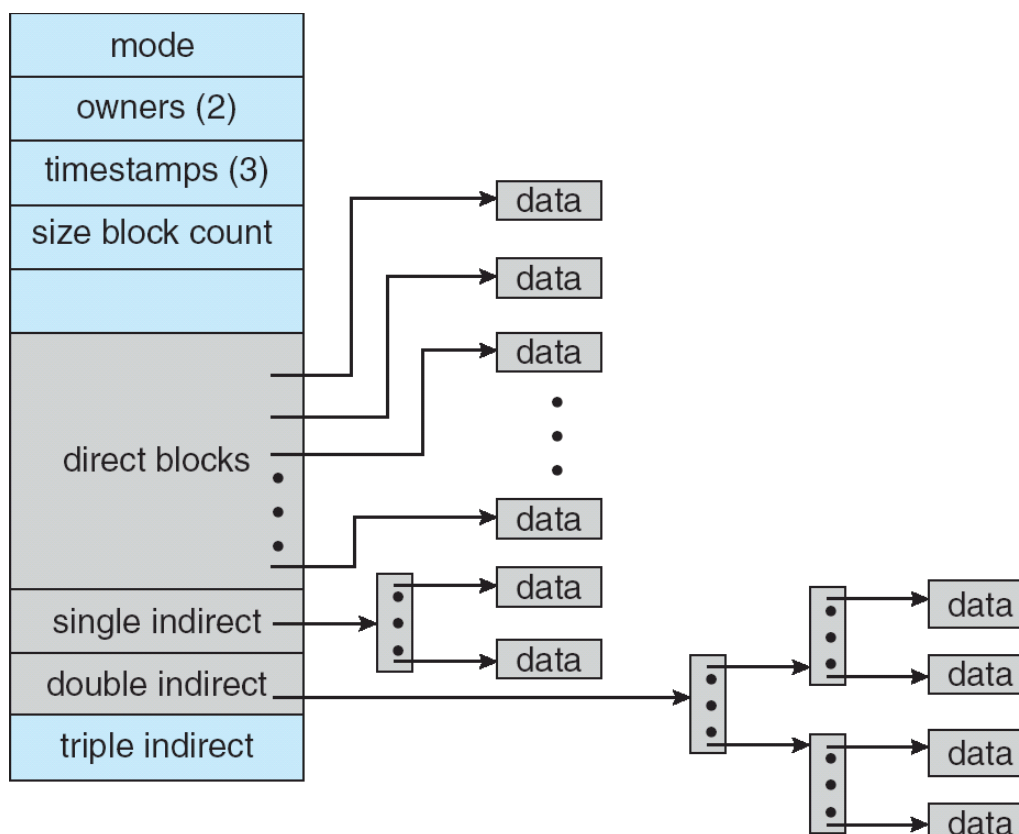
1. 输出 `inode` 详细信息（间接块不展开）

2. 物理地址（物理块号，块内偏移）

四、 实验原理：

混合索引方式是将多种索引分配方式相结合而形成的一种分配方式。系统既采用了直接地址，又采用了一级索引分配方式，两级索引分配方式，还采用了三级索引分配方式。

在本次实验中，我们设置了一级、二级、三级间接块作为混合索引的结构，每个间接块包含十个元素，逻辑块 n 对应物理块 n ，在设置好 `inode` 的结构后，文件按照顺序依次填充占用物理块。



五、 实验器材（设备、元器件）

本实验所需的硬件主要有：PC 计算机，具体要求如下：

软件环境

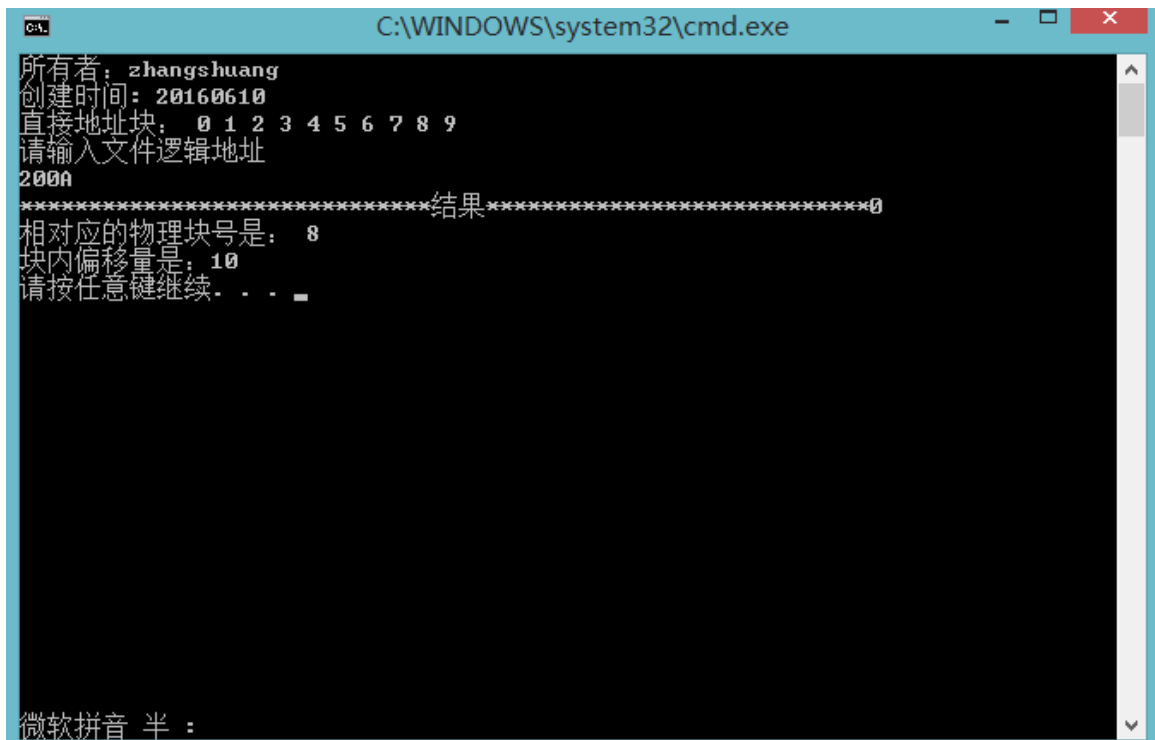
1. 操作系统： Windows 7；
2. 开发平台： VC2013；
3. 工具语言： C++语言

六、 实验步骤：

1. 打开 VS2013，新建项目；
2. 设置盘块大小、包含元素数量：假设虚拟地址为 32 位，一个磁盘块的大小为 1KB，逻辑块 n 对应物理块 n，每种间接块包含 10 个元素，文件按顺序占用物理块. 注：这里定义的 inode 以及间接块主要是为了表明概念上的结构，与真实 inode 不完全相同

3. 创建一级二级三级间接块的结构函数，inode 的结构函数
4. 实施地址转换的函数（核心函数）：初始化索引和各级间接块，按照输入，由于一个间接块包含 10 个元素，根据各级索引的计算方法计算，分到各逻辑地址所对应的物理地址。
5. 主函数：输入文件的逻辑地址，输出相应物理块号和偏移量。

七、实验结论、心得体会和改进建议：



```
C:\WINDOWS\system32\cmd.exe
所有者: zhangshuang
创建时间: 20160610
直接地址块: 0 1 2 3 4 5 6 7 8 9
请输入文件逻辑地址
2000
*****结果*****
相对应的物理块号是: 8
块内偏移量是: 10
请按任意键继续. . .
```

通过本次实验我了解了混合索引的逻辑地址到物理地址映射的基本思想和计算方法，直接索引、多级索引的概念和如何进行逻辑地址和物理地址的转换。同时编程实现了，提高了我的动手能力。