**操作系统设计课程**

**本次实验的要求：**

1. **本课程是实验课程，请每位同学确保在规定时间内完成实验。**
2. **本课程成绩由两部分构成，验收成绩占70%，实验报告成绩占30%。实验结束时，由老师上机验收成果，同学们提交实验报告。**
3. **本课程设置了三类实验：操作系统模拟实验、操作系统内核实验和复现操作系统实验，模拟实验难度低于内核实验和复现实验，同学们可以根据自己的情况进行选择。完成模拟实验的同学最高分不超过89分，选择内核实验和复现实验的同学有机会得到>=90分的成绩。完成内核实验和复现实验的同学的得分并不一定高于完成模拟实验的同学，最终成绩取决于具体的完成情况。**

# 实验报告提交word或PDF格式文档，实验报告撰写要求参见“操作系统实验报告书写规范”文档，请保留报告封面的得分表格和文档末尾的教师评语表格。对文档编写不清楚的地方请参见：《学术论文编写规则》（GB/T 7713.2—2022）<https://mp.weixin.qq.com/s/NO6XIRAfT8eYGQ3XIvu65Q>

1. **提交方式：。**

操作系统模拟实验

**这里给大家提供了五个实习，每个实习中可能包含多个题目，请大家从每个实习中选择一道题完成，一共要完成五个题目。**

**注意：实习二的第一题和实习三中的第一题，实习二的第二题和实习三中的第二题，算法的实现思想类似，要求完成不同思想的两个算法。比如实习二选第一题，则实习三就不能再选第一题。**

**实习三选择第四题的同学可获得（<=89分）的分数。**

**实习三选择前三题之一的同学可获得（<=84分）的分数。**

**若想获得超过90分的分数，只有选择操作系统内核实验或复现一个操作系统才有机会。**

## 实习一 进程调度及状态转换

### 一、实习内容

**实现进程调度及进程状态转换**。

### 二、实习目的

**本实习模拟在单处理器环境下的进程调度及状态转换**，**加深对进程调度、进程状态及其状态转换的理解。**

### 三、实习题目

**实习题目分两部分说明，进程调度部分及进程状态转换部分。**

**进程调度部分**：**设计一个按优先数调度算法实现进程调度的程序**。

[**提示**]：

**(1) 假定系统有5个进程，每个进程用一个PCB来代表。PCB的结构为：**

* **进程名——如P1~P5。**
* **指针——按优先数的大小把5个进程连成队列，用指针指出下一个进程PCB的首地址。**
* **要求运行时间——假设进程需要运行的单位时间数。**
* **优先数——赋予进程的优先数，调度时总是选取优先数大的进程先执行。**
* **状态——至少包含三种状态：就绪，执行和阻塞，还有一个结束进程队列。进程初始状态都为就绪状态。**

**(2) 开始运行之前，为每个进程确定它的“优先数”和“要求运行时间”。通过键盘输入这些参数。**

**(3) 处理器总是选择就绪队列队首进程运行。采用动态改变优先数的办法，进程每运行1次，优先数减1，要求运行时间减1。**

**(4) 进程运行一次后，若要求运行时间不等于0，则将它加入就绪队列，否则，将状态改为“结束”，退出就绪队列，加入结束队列。**

**(5) 若就绪队列为空，结束，否则转到(3)重复。**

**要求能接受键盘输入的进程优先数及要求运行时间，能显示每次进程调度的情况，如当前运行进程的名称。**

**进程状态转换部分：模拟进程状态转换及其PCB的变化**

**提示：**

**(1) 设计并实现一个模拟进程状态转换及其原因的程序，当输入进程名及引起状态变化的原因后，进程状态发生转换。如就绪进程p，进程调度选中p，则p状态由就绪转为运行，若p在运行状态时申请输入/输出，则p的状态由运行变为阻塞，输入/输出完成则p的状态由阻塞变为就绪。**

**(2) 进程的状态为三状态或更多。**

**(3) 设计出可视性较好的界面，应能反映出进程状态变化前后的对应PCB内容、进程队列组织的变化。**

**(4) 鼓励在实验中加入新的观点或想法，并加以实现。**

## 实习二 主存空间的分配和回收

### 一、实习内容

**主存储器空间的分配和回收**。

### 二、实习目的

**通过本实习帮助理解在不同的存储管理方式下应怎样进行存储空间的分配和回收。**

### 三、实习题目

**本实习有两题，可任选一题。**

#### 第一题：可变分区管理方式下采用首次适应算法实现主存分配和回收

**[提示]：**

**(1) 可变分区方式是按作业需要的主存空间大小来分割分区的。当要装入一个作业时，根据作业需要的主存容量查看是否有足够的空闲空间，若有，则按需分配，否则，作业无法装入。假定内存大小为128K，空闲区说明表格式为：**

* **分区号——表示是第几个空闲分区；**
* **起始地址——指出空闲区的起始地址；**
* **长度——一个连续空闲区的长度；**

**(2) 采用首次适应算法分配回收内存空间。运行时，输入一系列分配请求和回收请求。**

**要求能接受来自键盘的空间申请及释放请求，能显示分区分配及回收后的内存布局情况。**

#### 第二题：在分页管理方式下采用位示图来表示主存分配情况，实现主存分配和回收

**[提示]：**

**(1) 假定系统的主存被分成大小相等的64个块，用0/1对应空闲/占用。**

**(2) 当要装入一个作业时，根据作业对主存的需求量，先查空闲块数是否能满足作业要求，若能满足，则查位示图，修改位示图和空闲块数。位置与块号的对应关系为：**

**块号=j\*8+i，其中i表示位，j表示字节。**

**根据分配的块号建立页表。页表包括两项：页号和块号。**

**(3) 回收时，修改位示图和空闲块数。**

**要求能接受来自键盘的空间申请及释放请求，能显示位示图和空闲块数的变化，能显示进程的页表。**

## 实习三 磁盘存储空间管理及文件系统

### 一、实习内容

**模拟磁盘空闲空间的表示方法，以及模拟实现磁盘空间的分配和回收。**

### 二、实习目的

**磁盘初始化时把磁盘存储空间分成许多块（扇区），这些空间可以被多个用户共享。用户作业在执行期间常常要在磁盘上建立文件或把已经建立在磁盘上的文件删去，这就涉及到磁盘存储空间的分配和回收。一个文件存放到磁盘上，可以组织成顺序文件（连续文件）、链接文件（串联文件）、索引文件等，因此，磁盘存储空间的分配有两种方式，一种是分配连续的存储空间，另一种是可以分配不连续的存储空间。怎样有效地管理磁盘存储空间是操作系统应解决的一个重要问题，通过本实习使学生掌握磁盘存储空间的分配和回收算法。**

### 三、实习题目

**本实习有四个题目，可以任选一个，但不能与内存管理的题目类似。**

#### 第一题：连续磁盘存储空间的分配和回收

**[提示]：**

**(1) 要在磁盘上建立顺序文件时，必须把按序排列的逻辑记录依次存放在磁盘的连续存储空间中。可假定磁盘初始化时，已把磁盘存储空间划分成若干等长的块（扇区），按柱面号和盘面号的顺序给每一块确定一个编号。随着文件的建立、删除、磁盘存储空间被分成许多区（每一区包含若干块），有的区存放着文件，而有的区是空闲的。当要建立顺序文件时必须找到一个合适的空闲区来存放文件记录，当一个文件被删除时，则该文件占用的区应成为空闲区。为此可用一张空闲区表来记录磁盘存储空间中尚未占用的部分，格式如下：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序 号** | **起始空闲块号** | **空闲块个数** | **状 态** |
| **1** | **5** | **6** | **未 分 配** |
| **2** | **14** | **3** | **未 分 配** |
| **3** | **21** | **30** | **未 分 配** |
| **4** |  |  |  |
| **。。。** |  |  |  |

**(2) 建立文件时，先查找空闲区表，从空闲区表中找出一个块数能满足要求的区，由起始空闲块号能依次推得可使用的其它块号。若不需要占用该区的所有块时，则剩余的块仍应为未分配的空闲块，这时要修改起始空闲块号和空闲块数。若占用了该区的所有块，则删去该空闲区。删除一个文件时，需要考虑空闲块的合并情况。**

**磁盘存储空间的分配和回收算法类似于主存储器的可变分区方式的分配和回收。同学们可参考实习二的第一题。**

**(3) 当找到空闲块后，必须启动磁盘把信息存放到指定的块中，启动磁盘必须给出由三个参数组成的物理地址：盘面号、柱面号和物理记录号（即扇区号）。故必须把找到的空闲块号换算成磁盘的物理地址。**

**为了减少移臂次数，磁盘上的信息按柱面上各磁道顺序存放。现假定一个盘组共有200个柱面，（编号0-199）每个柱面有20个磁道（编号0-19，同一柱面上的各磁道分布在各盘面上，故磁道号即盘面号。），每个磁道被分成等长的6个物理记录（编号0-5，每个盘面被分成若干个扇区，故每个磁道上的物理记录号即为对应的扇区号）。那么，空闲块号与磁盘物理地址的对应关系如下：**

**则 物理记录号 = 空闲块号 % 6**

**磁道号=（空闲块号 / 6 ）% 20**

**柱面号=（空闲块号 / 6）/20**

**(4) 删除一个文件时，从文件目录表中可得到该文件在磁盘上的起始地址和逻辑记录个数，假定每个逻辑记录占磁盘上的一块，则可推算出归还后的起始空闲块号和块数，登记到空闲区表中。换算关系如下：**

**起始空闲块号=（柱面号20+磁道号）6+物理记录号**

**空闲块数=逻辑记录数**

**(5) 请设计磁盘存储空间的分配和回收程序，要求把分配到的空闲块转换成磁盘物理地址，把归还的磁盘空间转换成空闲块号。**

**要求能接受来自键盘的空间申请及释放请求，能显示或打印分配及回收后的空闲区表以及分配到的磁盘空间的起始物理地址：包括柱面号、磁道号、物理记录号（扇区号）。**

#### 第二题：用位示图管理磁盘存储空间

**[提示]：**

**(1) 为了提高磁盘存储空间的利用率，可在磁盘上组织成链接文件、索引文件，这类文件可以把逻辑记录存放在不连续的存储空间。为了表示哪些磁盘空间已被占用，哪些磁盘空间是空闲的，可用位示图来指出。位示图由若干字节构成，每一位与磁盘上的一块对应，“1”状态表示相应块已占用，“0”状态表示该块为空闲。位示图的形式与实习二中的位示图一样，但要注意，对于主存储空间和磁盘存储空间应该用不同的位示图来管理，绝不可混用。**

**(2) 申请一块磁盘空间时，由分配程序查位示图，找出一个为“0”的位，计算出这一位对应块的磁盘物理地址，且把该位置成占用状态“1”。假设现在有一个盘组共8个柱面，每个柱面有2个磁道（盘面），每个磁道分成4个物理记录。那么，当在位示图中找到某一字节的某一位为“0”时，这个空闲块对应的磁盘物理地址为：**

**柱面号=字节号**

**磁道号= 位数 / 4**

**物理记录号= 位数 % 4**

**(3) 归还一块磁盘空间时，由回收程序根据归还的磁盘物理地址计算出归还块在位示图中的对应位，把该位置成“0”。按照（2）中假设的盘组，归还块在位示图中的位置计算如下：**

**字节号=柱面号**

**位数=磁道号4+物理记录号**

**(4) 设计申请磁盘空间和归还磁盘空间的程序。**

**要求能接受来自键盘的空间申请及释放请求，要求能显示或打印程序运行前和运行后的位示图；分配时把分配到的磁盘空间的物理地址显示或打印出来，归还时把归还块对应于位示图的字节号和位数显示或打印出来。**

#### 第三题：模拟UNIX系统的空闲块成组链接法，实现磁盘存储空间的管理

**[提示]：**

**(1) 假定磁盘存储空间已被划分成长度为n的等长块，共有M块可供使用。UNIX系统中采用空闲块成组链接的方法来管理磁盘存储空间，将磁盘中的每N个空闲块（N<M）分成一组，最后一组可以不足N块，每组的第一块中登记了下一组空闲块的块数和块号，第一组的块数和块号登记在专用块中，登记的格式如下：**

|  |  |
| --- | --- |
| **0** | **闲块数k** |
| **1** | **空闲块号1** |
| **2** | **空闲块号2** |
| **…** | **…** |
| **K** | **空闲块号k** |
|  |  |

**当第一项内容为“0”时，则第二项起指出的空闲块是最后一组。**

**(2) 可用二维数组A [0…M-1] [0…n-1]来模拟管理磁盘空间，用A[i]表示第I块，第0块A[0]作为专用块。**

**(3) 成组链接的分组情况记录在磁盘物理块中，为了查找链接情况，必须把它们读入主存，故当磁盘初始化后，系统先将专用块内容复制到主存中。定义一个数组MA存放专用块内容，即MA: =A[0]。申请一块磁盘空间时，查MA，从中找出空闲块号，当一组的空闲块只剩第一块时，则应把该块中指出的下一组的空闲块数和块号复制到专用块中，然后把该块分配给申请者。当一组的空闲块分配完后则把专用块内容（下一组链接情况）复制到主存，再为申请者分配。**

**(4) 归还一块时给出归还的块号，若当前组不满规定块数时，将归还块登记入该组；若当前组已满，则另建一新组，这时归还块作为新一组的第一块，应把主存中登记的一组链接情况MA复制到归还块中，然后在MA重新登记一个新组。**

**(5) 设计分配和归还磁盘空间的程序。**

**要求能接受来自键盘的空间申请及释放请求，能显示或打印分配的磁盘空间的块号，在完成一次分配或归还后能显示或打印各空闲块组的情况（各组的空闲块数和块号）。本实习省去了块号与物理地址之间的转换工作，而在实际的系统中必须进行块号与物理地址的转换工作。**

#### 第四题：设计一个简单的两级目录文件系统

**通过简单文件系统的设计实现，加深对文件系统的内部数据结构、功能及实现过程的理解。**

**提示：**

**(1) 假设有8个用户，每个用户保存文件数不超过10个，所有用户可同时打开文件数不超过16。**

**(2) 二级目录结构包含主目录及用户文件目录。主目录应包含用户名、用户文件目录指针等字段，用户文件目录应包含文件名、保护字段（包含三个权限读、写、执行，分别用R、W、X表示）、文件长度、文件存放地址等字段，系统打开文件目录结构包含用户文件目录结构的相关字段并增加文件读写指针字段。文件的物理结构采用链接结构（也可以使用其他物理结构）。**

**(3) 文件可以用内存映射文件方式实现，即通过将磁盘块映射到内存中的一页，允许文件 I/O被处理为常规内存访问。在退出系统时，相关文件内容及目录内容应以文件形式存放在磁盘上，下次启动文件系统时再装入内存。**

**(4) 实现下列文件操作命令功能：dir 列文件目录、open 打开文件、close 关闭文件、create 创建文件、delete 删除文件、read 读文件、write 写文件。**

**1）dir 用户名，算法思路如下：**

**① 判断用户是否存在，若不存在显示错误信息并返回；**

**② 显示该用户文件目录的内容，包含文件名、文件长度，保护字段。**

**2）open 用户名/文件名，算法思路如下：**

**①判断用户是否存在，若不存在显示错误信息并返回；**

**②系统查找相应用户的用户文件目录，若没找到指定文件则显示文件不存在并返回；**

**③申请一个空闲的系统打开文件表项，若没有空闲表项则显示打开文件过多并返回；**

**④将文件目录复制到系统打开文件表项中并初始化文件读写指针，返回打开文件在系统打开文件表的表项下标（后面称为文件描述符）供读写操作使用，系统打开文件表设计为结构数组（也可以设计为其他形式，此时返回的是能找到打开文件属性信息数据结构的指针，也称为文件句柄）。**

**3）close 用户文件描述符，算法思路如下：**

**①判断用户文件描述符是否存在，若不存在显示错误信息并返回；**

**②将文件描述符所指系统打开文件表项释放，使其成为空闲表项。**

**4）create 用户名/文件名/保护字段，算法思路如下：**

**①判断用户是否存在，若不存在显示错误信息并返回；**

**②系统查找相应用户的用户文件目录，检查用户已有文件数是否小于10，若有10个文件且新建文件名不存在于用户文件目录中则显示创建文件已达最大数目并返回；**

**③在用户文件目录表中查找指定文件，若找到指定文件则将该文件长度清零，申请一个空闲的系统打开文件表项，将文件目录复制到系统打开文件表项中并初始化文件读写指针，返回该文件的文件描述符。**

**④若没有找到指定文件，则申请一个空闲文件目录项，填写文件名及保护字段信息，初始化其他字段；再申请一个空闲的系统打开文件表项，将文件目录复制到系统打开文件表项中并初始化文件读写指针，返回文件描述符。**

**5）delete 用户名/文件名，算法思路如下：**

**①判断用户是否存在，若不存在显示错误信息并返回；**

**②系统查找相应用户的用户文件目录，检查文件是否存在，若文件不存在则显示错误信息并返回；**

**③在用户文件目录表中查找指定文件，释放文件占用的空间，释放目录项。**

**6）read 文件描述符/存放读数据的缓冲区/本次读字符数，算法思路如下：**

**①判断用户文件描述符是否存在，若不存在显示错误信息并返回；**

**②根据文件描述符找到相应的系统打开文件表项，判断该文件是否允许读操作，若不允许则显示错误信息并返回；**

**③按照本次读字符数从当前文件指针开始读数据到缓冲区中。**

**7）write 文件描述符/存放写数据的缓冲区/本次写字符数，算法思路如下：**

**①判断用户文件描述符是否存在，若不存在显示错误信息并返回；**

**②根据文件描述符找到相应的打开文件表项，判断该文件是否允许写操作，若不允许则显示错误信息并返回；**

**③按照本次写字符数从文件尾开始将缓冲区中的数据写到文件中，并修改目录中的文件大小**

**(5) 程序启动后，应先将磁盘上的两级目录结构及文件内容读入内存数据结构中并初始化系统打开文件表（可以事先确定文件系统的数据结构初值，包含主目录的8个用户信息及指向各自用户文件目录的指针，每用户可以有初始文件也可以没有，若有初始文件则还需为这些已存在文件赋值，这些初值以文件形式存放在磁盘上）。然后显示接收用户命令输入的界面，用户输入命令，系统执行命令并显示命令执行结果，还应显示与命令相关的目录及文件内容变化前后情况。**

**(6) 鼓励在实验中加入外存空间管理等功能，以使本文件系统功能更完善。**

## 实习四 进程创建

### 一、实习内容

**利用Linux的消息通信机制实现父子进程的消息通信。**

### 二、实习目的

**了解进程的创建及进程通信过程，进一步理解进程的概念，明确进程和程序的区别。**

### 三、实习题目

### 编制一段程序，利用fork( )及消息通信系统调用实现父子进程间的通信。

**[提示]：**

**(1) 可用fork()系统调用来创建一个新进程。**

**系统调用格式：pid=fork()**

**fork()返回值意义如下：**

* **=0：若返回值为0，表示当前进程是子进程。**
* **>0：若返回值大于0，表示当前进程是父进程，返回值为子进程的pid值。**
* **-1：若返回值小于0，表示进程创建失败。**

**如果fork()调用成功，它向父进程返回子进程的pid，并向子进程返回0，即fork()被调用了一次，但返回了两次。此时OS在内存中建立一个新进程，所建的新进程是调用fork()父进程的副本，称为子进程。子进程继承了父进程的许多特性，并具有与父进程完全相同的用户级上下文。父进程与子进程并发执行。**

**(2) 编译和执行的方法：**

**gcc 源文件名 -o 执行文件名**

**最后，在shell提示符下输入： ./执行文件名**

**就可执行该文件。**

**(3) Linux的进程通信机制有信号、管道、共享内存、消息传递及信号量。消息是一个格式化的可变长的信息单元。消息机制允许进程向其他进程发送消息。当一个进程收到多个消息时，可将这些消息排成一个队列。**

**在Linux中，消息机制向用户提供了四个系统调用，分别用于建立、发送、接收和管理消息。**

**1）系统调用msgget用来建立消息队列或者获取消息队列的描述符。该系统调用的语法格式如下：**

**int msgget(key,msgflg)；**

**key\_t key；**

**int msgflg；**

**其中，key是用户指定的消息队列的名字；msgflg是用户设置的标志和访问方式。如IPC\_CREAT表示系统中若无以key命名的消息队列，则建立消息队列；若该队列已存在，则返回该消息队列的描述符。**

**2）系统调用msgsnd（ ）向指定的消息队列发送一个消息，并将发送消息链接到该消息队列的尾部。系统调用的语法格式为：**

**int msgsnd(msgid,msgp,msgsz,msgflg)；**

**int msgid，msgsz，msgflg；**

**struct msgbuf \*msgp；**

**其中，msgid是由msgget返回的消息队列描述符；msgp是指向用户消息缓冲区的指针。msgsz是消息的长度，msgflg规定了当无内存空间存储消息时，进程是等待还是立即返回。**

**3）系统调用msgrcv( ) 从指定消息队列中接收指定类型的消息，其语法格式如下：**

**int msgrcv (msgid, msgp, msgsz, msgtyp, msgflg)；**

**int msgid,msgsz,msgflg；**

**struct msgbuf \* msgp；**

**long msgtyp；**

**其中，msgid、msgp、msgsz、msgflg与msgsnd中的对应参数相似，msgtyp是接收消息的类型。**

**msgtyp的取值有：**

**msgtyp＝0时，核心寻找消息队列中的第一个消息，并将它返回给调用者；**

**msgtyp为正整数时，核心返回给定消息类型的第一个消息；**

**msgtyp为负整数时，核心应在其类型值小于或等于msgtyp绝对值的所有消息中，选出其类型值最低的一个消息返回。**

**4）用户在建立了消息队列后，可以利用msgctl系统调用来读取它的状态信息并进行修改，如查询消息队列描述符、修改消息队列的许可权等。msgctl的语法格式为：**

**int msgctl(msgid,cmd,buf)；**

**int msgid,cmd；**

**struct msgid\_ds \*buf；**

**其中，buf是用户缓冲区地址，供用户存放控制参数和查询结果；cmd是规定的命令。**

## 实习五 进程同步

### 一、实习内容

**模拟实现同步机构，以避免发生进程执行时可能出现的与时间有关的错误。**

### 二、实习目的

**当进程并发执行时，如果对进程访问的共享变量不加限制，就会产生“与时间有关”的错误。为了防止这类错误，系统必须用同步机构来控制进程对共享变量的访问。**

**一般说，同步机构是由若干条同步原语所组成。本实验要求学生模拟P、V操作同步机构的实现，模拟进程的并发执行，了解进程并发执行时同步机构的作用。**

### 三、实习题目

#### 模拟P、V操作实现同步机构，且用P、V操作解决生产者—消费者问题。

**[提示]：**

**(1) P、V操作同步机构，由P操作原语和V操作原语组成，它们的定义如下：**

* **P操作原语P (s)：将信号量s减去1，若结果小于0，则执行原语的进程被置成等待信号量s的状态。**
* **V操作原语V (s)：将信号量s加1，若结果不大于0，则释放一个等待信号量s的进程。**

**这两条原语是如下的两个过程：**

**procedure p (var s: semaphore);**

**begin s: = s-1;**

**if s<0 then W (s)**

**end {p}**

**procedure v (var s: semaphore);**

**begin s: = s+1;**

**if s£0 then R (s)**

**end {v}**

**其中W（s）表示将调用过程的进程置为等待信号量s的状态；R（s）表示释放一个等待信号量s的进程。**

**在系统初始化时应把semaphore定义为某个类型，为简单起见，在模拟实验中可把上述的semaphore直接改成integer。**

**(2) 生产者—消费者问题。**

**假定有一个生产者和一个消费者，生产者每次生产一件产品，并把生产的产品存入共享缓冲器以供消费者取走使用。消费者每次从缓冲器内取出一件产品去消费。禁止生产者将产品放入已满的缓冲器内，禁止消费者从空缓冲器内取产品。假定缓冲器内可同时存放10件产品，用P、V操作来实现生产者和消费者之间的同步，生产者和消费者两个进程的程序如下：**

**B: array [0..9] of products;**

**s1, s2; semaphore;**

**s1: =10, s2: =0;**

**IN, out: integer;**

**IN: =0; out: =0;**

**cobegin**

**procedure producer;**

**c: products;**

**begin**

**L1:**

**Produce (c);**

**P (s1);**

**B[IN]: =C;**

**IN: =(IN+1)mod 10;**

**V (s2);**

**goto L1**

**end;**

**procedure consumer;**

**x: products;**

**begin**

**L2: p (s2);**

**x: =B[out];**

**out: =(out+1) mod10;**

**v (s1);**

**consume (x);**

**goto L2**

**end;**

**coend.**

**其中的semaphore和products是预先定义的两个类型，在模拟实现中semaphore用integer代替，products可用integer或char等代替。**

**(3) 进程控制块PCB。**

**为了记录进程执行时的情况，以及进程让出处理器后的状态，断点等信息，每个进程都有一个进程控制块PCB。在模拟实验中，假设进程控制块的结构如图1。其中进程的状态有：运行态、就绪态、等待态和完成态。当进程处于等待态时，在进程控制块PCB中要说明进程等待原因（在模拟实验中进程等待原因是为等待信号量s1或s2）；当进程处于等待态或就绪态时，PCB中保留了断点信息，一旦进程再度占有处理器则从断点位置继续运行；当进程处于完成状态，表示进程执行结束。**

|  |
| --- |
| **进程名** |
| **状态** |
| **等待原因** |
| **断点** |

**图1 进程控制块结构**

**(4) 处理器的模拟。**

**计算机硬件提供了一组机器指令，处理器的主要职责是解释执行机器指令。为了模拟生产者和消费者进程的并发执行，我们必须模拟一组指令和处理职能。**

**模拟的一组指令见图2，其中每条指令的功能由一个过程来实现。用变量PC来模拟“指令计数器”，假设模拟的指令长度为1，每执行一条模拟指令后，PC加1，提取出下一条指令地址。使用模拟的指令，可把生产者和消费者进程的程序表示为图3的形式。**

**定义两个一维数组PA[0..4]和SA[0..4]，每一个PA[i]存放生产者程序中的一条模拟指令执行的入口地址；每个SA[i]存放消费者程序中的一条模拟指令执行的入口地址。于是模拟处理器执行一条指令的过程为：取出PC之值，按PA[PC]或SA[PC]得模拟指令执行的入口地址，将PC之值加1，转向由入口地址确定的相应的过程执行。**

**(5) 程序设计**

**本实验中的程序由三部分组成：初始化程序、处理器调度程序、模拟处理器指令执行程序。各部分程序的功能及相互间的关系由图4至图7给出。**

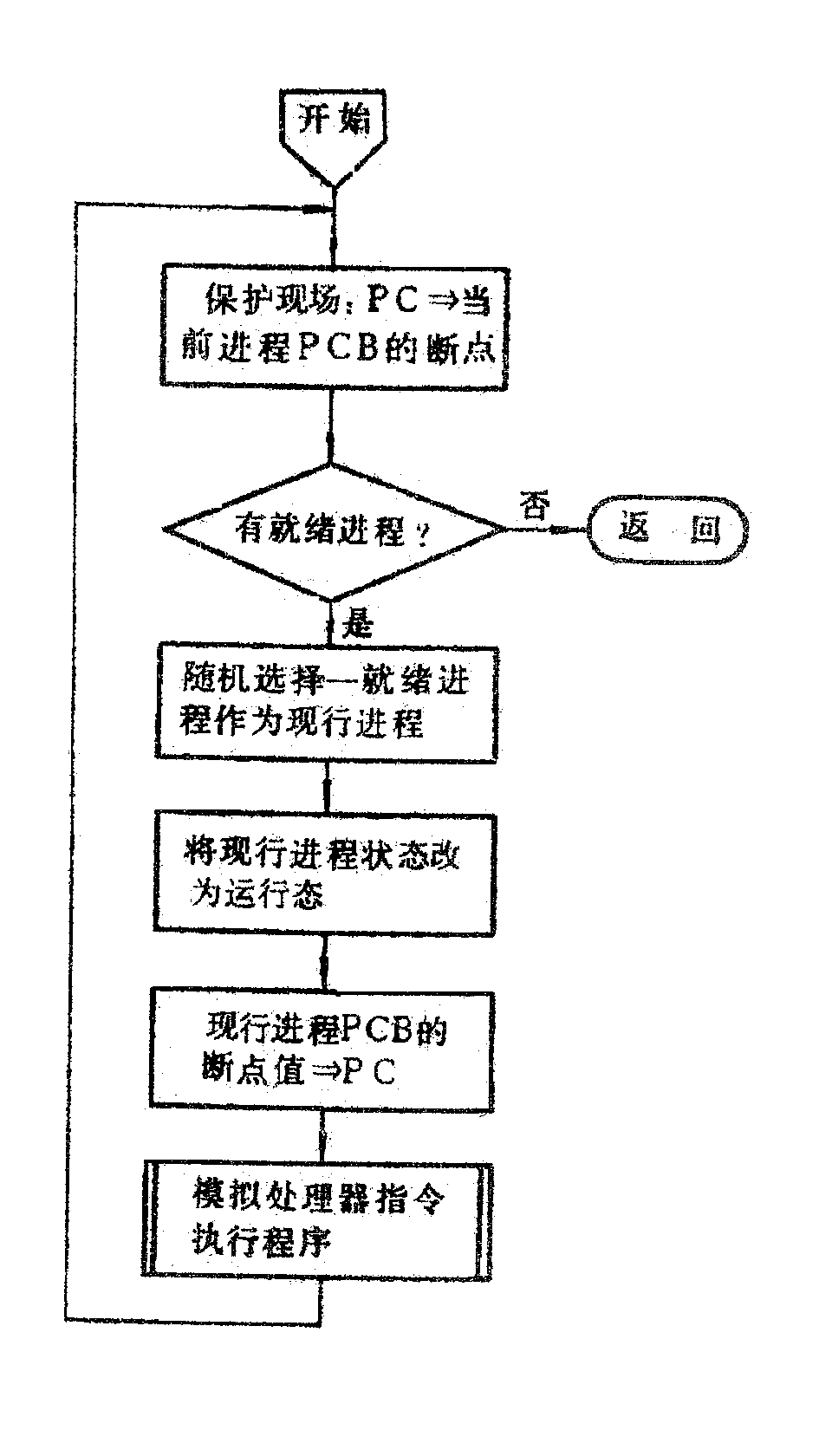
|  |  |
| --- | --- |
| **模拟的指令** | **功 能** |
| **p (s)** | **执行P操作原语** |
| **v (s)** | **执行V操作原语** |
| **Put** | **B[IN]: =product; IN: = (IN+1) mod 10** |
| **GET** | **x:=B[out]; out: =(out+1) mod 10** |
| **produce** | **输入一个字符放入C中** |
| **consume** | **打印或显示x中的字符** |
| **GOTO L** | **PC: =L** |
| **NOP** | **空操作** |

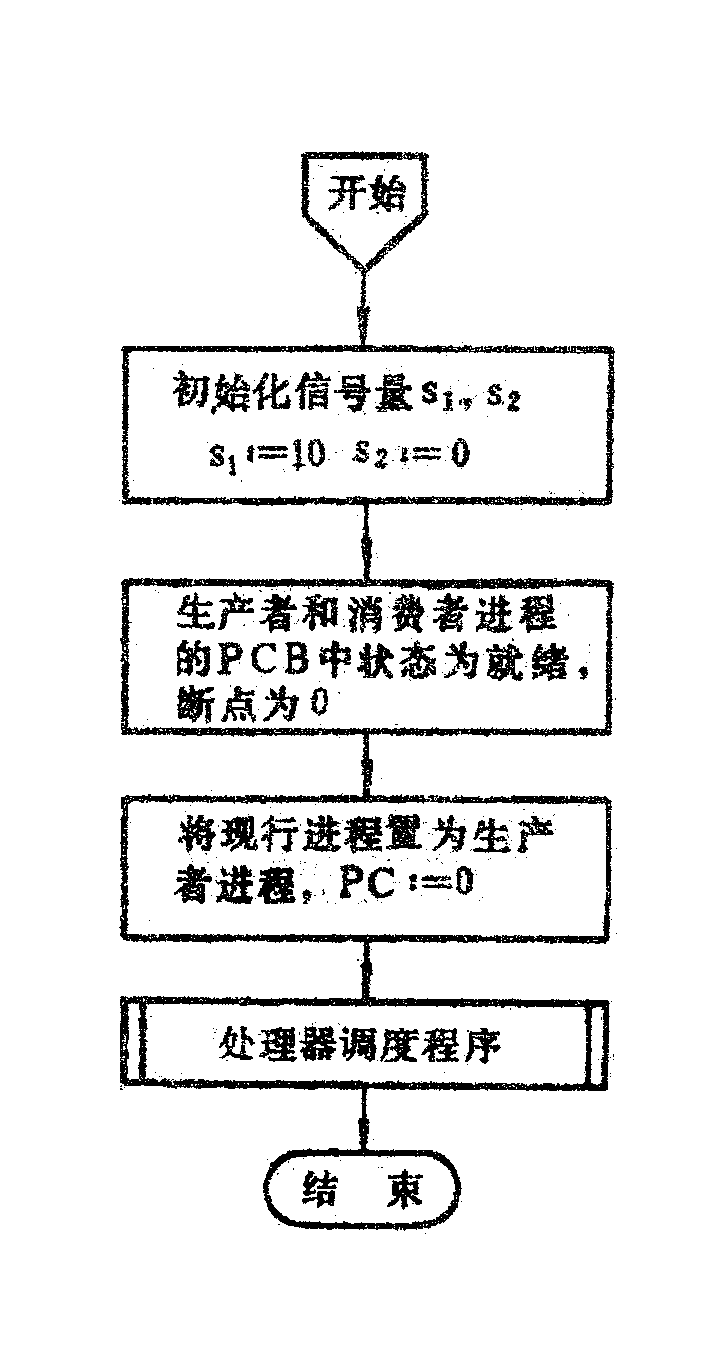
**图2 模拟的处理器指令**

|  |  |  |
| --- | --- | --- |
| **序号** | **生产者程序** | **消费者程序** |
| **0** | **produce** | **p (s2)** |
| **1** | **p (s1)** | **GET** |
| **2** | **PUT** | **v (s1)** |
| **3** | **v (s2)** | **consume** |
| **4** | **goto 0** | **goto 0** |

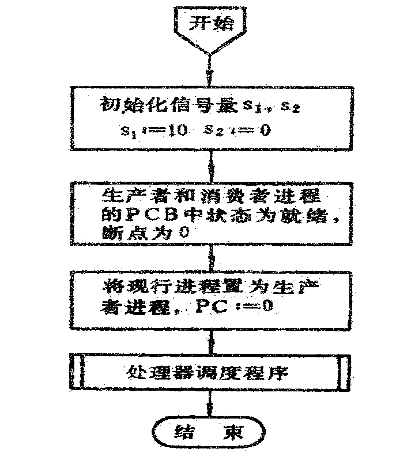
**图3 生产者和消费者程序**

* **初始化程序：模拟实验的程序从初始化程序入口启动，初始化工作包括对信号量s1、s2赋初值，对生产者、消费者进程的PCB初始化。初始化后转向处理调度程序，其流程如图4。**
* **处理器调度程序：在计算机系统中，进程并发执行时，任一进程占用处理器执行完一条指令后就有可能被打断而让出处理器由其它进程运行。故在模拟系统中也类似处理，每当执行一条模拟的指令后，保护当前进程的现场，让它成为非运行态，由处理器调度程序按随机数再选择一个就绪进程占用处理器运行。处理器调度程序流程见图5。**

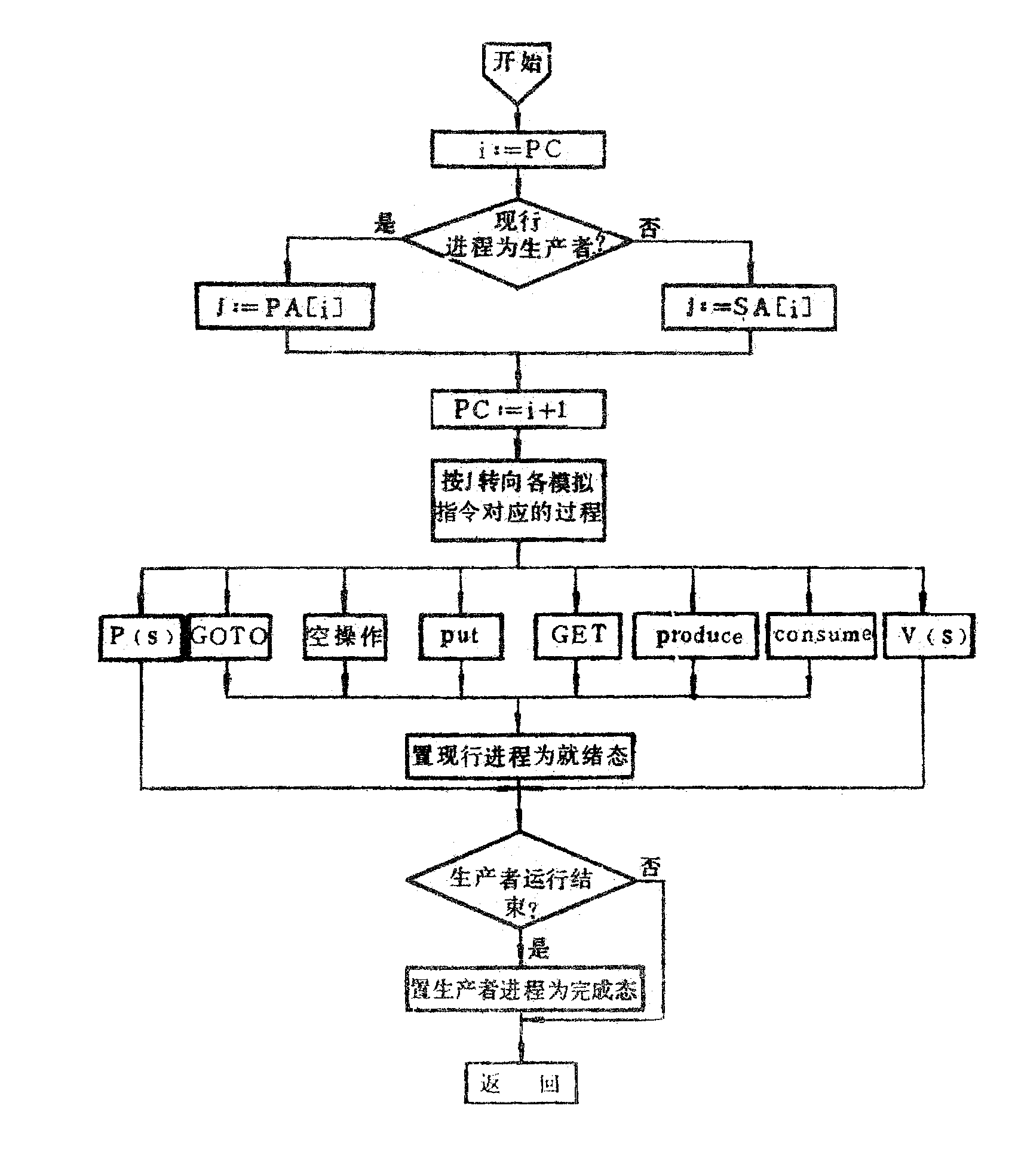




**图4 初始化流程**

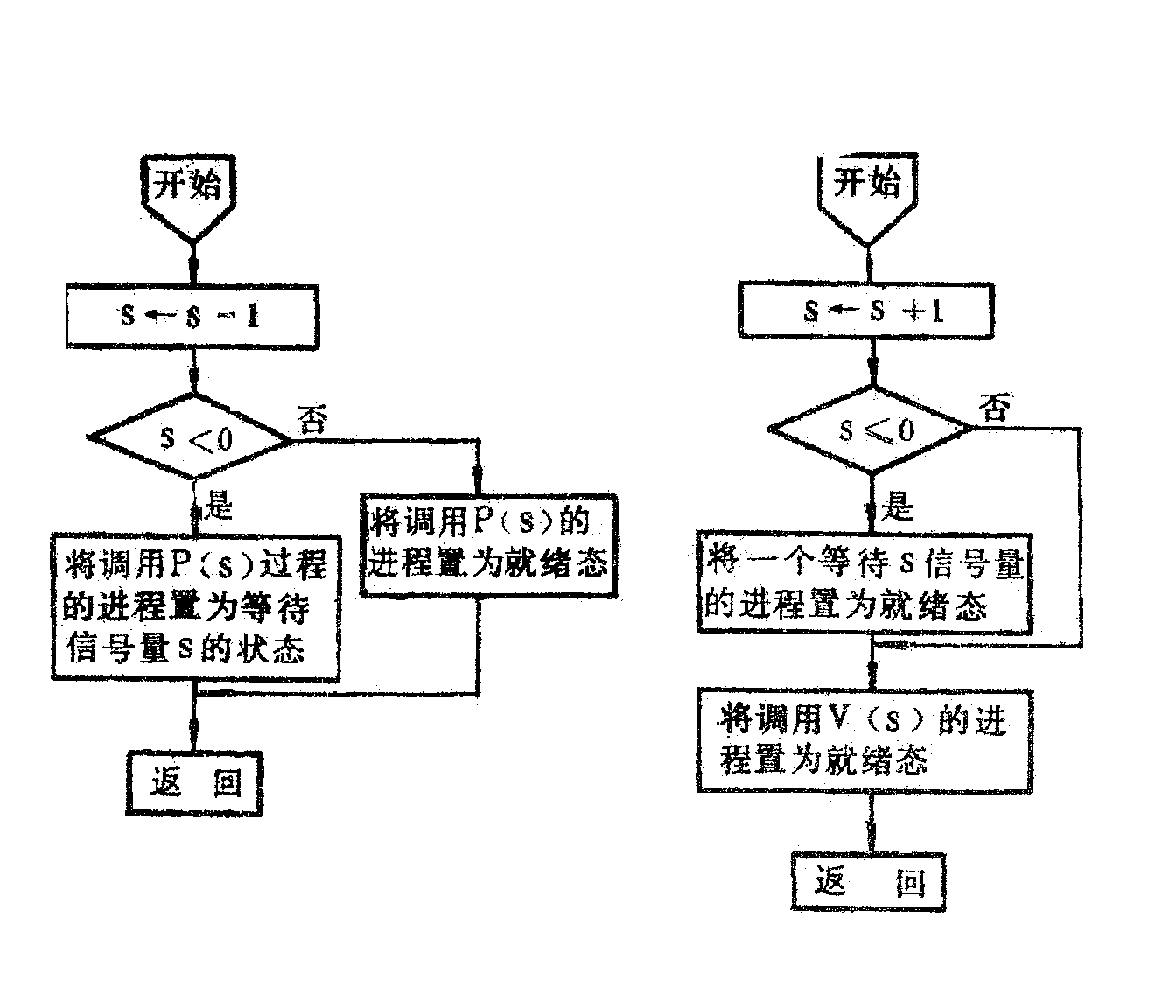


**图5 模拟处理器调度**

* **模拟处理器指令执行程序：按“指令计数器”PC之值执行指定的指令，且PC加1指向下一条指令。模拟处理器指令执行程序的流程图见图6和图7。**

**图6 模拟处理器指令执行**

**另外，为了使得模拟程序有一个结束条件，在图6中附加了“生产者运行结束”的条件判断，模拟时可以采用人工选择的方法实现。7给出了P（s）和V（s）模拟指令执行过程的流程。其它模拟指令的执行过程已在图2中指出。**

 **(a) 模拟P (s) (b) 模拟V (s)**

**图7 模拟P、V操作的执行**

**要求能输出生产者或消费者执行每条指令后的情况，以生产者为例如：应该依次输出：生产一个产品，生产者执行P操作（如果P操作使生产者进程阻塞，还应在输出中加上阻塞），放产品，生产者执行V操作，生产者执行goto；**

操作系统内核实验

**本实验要求同学们通过QEMU虚拟一个RISC-V处理器环境，通过在该虚拟环境下运行并修改Linux操作系统内核，来加深对操作系统原理的理解。首先同学们要根据自己的电脑配置，下载安装QEMU模拟器，能够虚拟RISC-V处理器环境，然后需要安装 RISC-V交叉编译工具链，使得编译出来的操作系统内核可以在虚拟的RISC-V硬件上运行，以测试对内核的修改是否能正常运行。**

**本次实验有四个部分，第一个部分是搭建实验环境，后序的内核实验分别涉及系统调用、内存管理、进程管理和文件系统。**

**操作系统内核实验需要阅读大量的资料，完成该实验的同学有机会得到(>=90分）的分数，同时也将获得巨大的收获和成就感。**

**对操作系统内核实验的具体指导和对应的参考文献将陆续发布。**

操作系统复现实验

**本实验要求同学们按下面的步骤实现一个操作系统，这是一个具有挑战性的任务，需要花费大量的时间学习新的知识。完成该实验的同学有机会得到(>=90分）的分数，同时也将获得巨大的收获和成就感。以下复现操作系统的步骤主要参考《自己动手写操作系统》，同学们也可以参照其他资料来复现一个操作系统。本文最后列出了一些有用的参考资料，同时欢迎同学们在复现操作系统的过程中根据自己的需求去寻找更多的资料。**

## 1 实验环境及环境搭建

**本实验在Ubuntu操作系统下完成，若同学们的工作环境是Windows操作系统，则需要首先在Windows操作系统下安装虚拟机软件（如Virtualbox，VMware等），然后在虚拟机上安装Ubuntu操作系统。Bochs是一款X86硬件平台的开源模拟器，可以通过纯软件的方式模拟硬件，从启动到重启，包括PC外设键盘、鼠标、VGA卡、磁盘、网卡等全部通过软件来模拟，因此Bochs模拟器非常适合开发操作系统，其自带bochsdbg调试器使得调试操作系统变得更容易。因此本实验将在Bochs虚拟出来的计算机上进行我们所设计的操作系统的模拟测试和运行。参考的版本为：VMware 15.5.1；Ubuntu 20.04.2.0；NASM 2.15.05；Bochs 2.6.9。**

**简要步骤如下：**

1. **卸载系统中已有的虚拟机如Virtualbox或VMware，安装最新版本的虚拟机；**
2. **在虚拟机上安装Ubuntu；**
3. **下载bochs源码，编译安装；**
4. **开始重新参考书上的第一个例子，操作系统引导盘的制作。**

## 2 动手写一个最小的“操作系统”

**实验内容：通过编译一段最基本的asm代码来初次体验操作系统的设计以及了解NASM编译的使用方法、dd命令写入磁盘的方法以及Bochs的使用方法。**

## 3 实现保护模式

**实验内容：认识保护模式，实现从实模式到保护模式的转换，GDT描述符；实现实模式大于1MB内存的寻址能力，并接着上一次实验，从保护模式返回到实模式，重新设置各个段寄存器的值；LDT描述符；学会使用挂载指令和运行程序。**

## 4 切换到保护模式

**实验内容：引导扇区突破512个字节的限制，将工作分给loader；加载loader进入内存并运行；将控制权交给loader。**

## 5 操作系统内核的雏形

**实验内容：在Linux下用汇编写Hello, World！；进一步，汇编和C同时使用；从loader到kernel内核，把kernel内核加载到内存；将控制权交给kernel内核；跳入保护模式,并显示内存的使用情况。**

## 6 进程与进程调度

**实验内容：进程切换；丰富中断处理程序，比如让时钟中断处理可以不停地发生而不是只发生一次，进程状态的保存与恢复，进程调度，解决中断重入问题。具体可以见博客，博客的网址为**<https://blog.csdn.net/qq_35353673/article/details/119010254>

## 7 操作系统的输入/输出系统

**实验内容：实现简单的I/O，从键盘输入字符的中断开始；获取并打印扫描码；创建对应打印扫描码解析数组，打印对应字符。**

# 参考资料

1.《自己动手写操作系统》，电子工业出版社，于渊；该书第2版名为《Orange'S：一个操作系统的实现》。虚拟机bochs可能不太好用，可以使用其他虚拟机软件，如vmware。

2.《自己动手写操作系统》读后感，网址：

<http://blog.csdn.net/zgh1988>

3. 维基百科上与操作系统开发相关的网站：

http://wiki.osdev.org/Main\_Page

4. 如何从零开始写一个简单操作系统，网址：

<https://www.zhihu.com/question/25628124>

5.《操作系统真相还原》，人民邮电出版社，郑钢，2016

6. MIT的操作系统设计课程，网址：

<https://pdos.csail.mit.edu/6.828/2017/xv6.html>

xv6起源于MIT的实验课程“操作系统工程”，英文名称是“Operating Systems Engineering”。xv6是对Dennis Ritchie和Ken Thompson的Unix版本6（v6）的重新实现，它松散地遵循v6的结构和风格，但使用ANSI C并基于x86多处理器实现。

7. 《Linux内核完全注释》，赵炯，机械工业出版社， 2007。基于某Linux版本，在此基础上改写系统部分代码。

8. 《30天自制操作系统》，川合秀实，人民邮电出版社，2012。

9. 《一个64位操作系统的设计与实现》，田宇，人民邮电出版社，2018年。

10. 清华大学操作系统设计课程网址：

<https://chyyuu.gitbooks.io/ucore_os_docs/content/>

11. https://os.educg.net/ 是2021年全国大学生计算机系统能力大赛操作系统设计赛的网站，技术报告那一栏有一些材料，感兴趣的同学可以下载来看。也可以将实验与参赛结合起来实现。