第一章　　习题及解答

1-2 批处理系统和分时系统各具有什么特点？为什么分时系统的响应较快？

答：在批处理系统中操作人员将作业成批装入计算机，在程序运行期间用户不能干预，用户使用计算机的方式是脱机操作方式。批处理系统中作业成批处理，系统内多道程序是并发执行的，所以其特点是：系统吞吐率高，但作业周转时间长，用户使用不方便。

在分时系统中不同用户通过各自的终端以交互方式共同使用一台计算机，计算机以“分时”的方法轮流为每个用户服务，用户使用计算机的方式是联机操作方式。分时系统的主要特点是多个用户同时使用计算机的同时性，人机问答方式的交互性，每个用户独立使用计算机的独占性以及系统快速响应的及时性。

分时系统一般采用时间片轮转的方法，使一台计算机同时为多个终端用户服务，因此分时系统的响应较快。

1-3 实时信息处理系统和分时系统从外表看来很相似，它们有什么本质的区别呢？

答：实时信息处理系统和分时系统从外表来看，都是一台计算机连接一个或多个终端设备；用户以联机方式直接与计算机交互。二者的本质区别是：

实时信息处理系统采用的进程调度策略是优先调度策略，而分时系统采用的进程调度策略是时间片轮转调度策略。

实时信息处理系统的终端设备通常只是作为执行装置或咨询装置，不允许用户编写新的程序或修改已有的程序。而分时系统的用户可以通过终端设备修改程序，可以与系统交互以控制程序的运行。

1-5 什么是多道程序设计技术？试述多道程序运行的特征？

答：多道程序设计技术是指同时多个作业或程序进入主存并允许它们交替执行和共享系统中的各类资源。当一道程序因某种原因如 I/O 请求而暂停执行时，CPU 立即转去执行另一道程序。多道程序运行具有如下特征：

多道：计算机内存中同时存放几道相互独立的程序。

宏观上并行：同时进入系统的几道程序都处于运行过程中，它们先后开始了各自的运行，但都未运行完毕。

微观上串行：从微观上看，主存中的多道程序轮流或分时地占有处理机，交替执行。

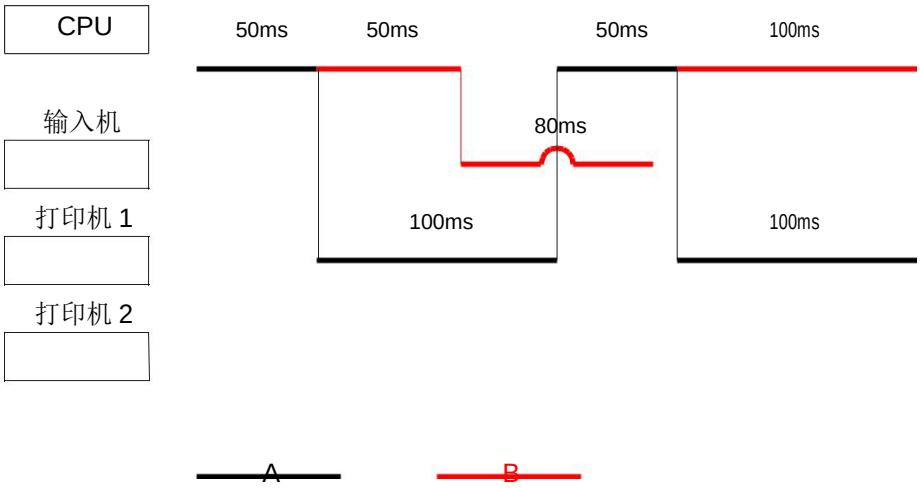
1-9 设一计算机系统有输入机一台、打印机两台，现有 A、B 两道程序同时投入运行，且程序 A 先运行，程序 B 后运行。程序 A 的运行轨迹为：计算 50ms，打印信息 100ms，再计算 50ms，打印信息 100ms，结束。程序 B 运行的轨迹为：计算 50ms，输入数据 80ms，再计算 100ms，结束。回答如下问题。

（1）用图画出这两道程序并发执行时的工作情况。

（2）说明在两道程序运行时，CPU 有无空闲等待?若有，在哪段时间内等?为什么会空闲等待?

（3）程序 A、B 运行时有无等待现象?在什么时候会发生等待现象?

答：（1） 这两道程序并发执行时的工作情况如下图所示。



（2）CPU 有空闲等待，当 B 50ms 计算完后，A 100ms 打印仍在进行，中间 CPU 空闲 50ms。

（3）程序 A、B 运行时有等待现象。当 B 80ms 输入完成后，需等待 20ms 后才能进行计算。

第二章　　习题及解答

2-1 什么是操作系统虚拟机？

答：操作系统是最基本的系统软件，它是硬件功能的第一层扩充。配置了操作系统的计算机称为操作系统虚拟机。

操作系统虚拟机除了可使用原来裸机提供的各种基本硬件指令，还可以使用操作系统提供的操作命令和系统调度命令。

2-3 什么是处理机的态？为什么要区分处理机所谓态？

答：所谓处理机的态就是处理机当前处于何种状态，正在执行哪类程序。为了保护操作系统，至少需要区分两种状态：管态和用户态。

操作系统是计算机系统中最重要的系统软件，为了能正确地进行管理和控制，其本身是不能被破坏的。为此，系统应能建立一个保护环境。当用户程序执行时，应有所限制，其所需资源必须向操作系统提出请求，自己不能随意取用系统资源，如不能直接启动外部设备的工作，更不能改变机器状态等。因此系统必须区分处理机的工作状态，即区分当时正在执行的程序的类别。

2-4 什么是管态？什么是用户态？二者有何区别？

答：管态又称为系统态，是操作系统的管理程序执行时机器所处的状态。在此状态下中央处理机可以使用全部机器指令，包括一组特权指令（例如，涉及外部设备的输入/输出指令、改变机器状态或修改存储保护的指令），可以使用所有的资源，允许访问整个存储区。

用户态又称为目态，是用户程序执行时机器所处的状态。在此状态下禁止使用特权指令，不能直接取用资源与改变机器状态，并且只允许用户程序访问自己的存储区域。

二者的区别如下所述。

（1）处理机当前正在执行的程序类别不同。管态执行的是系统程序；用户态执行的是用户程序。

（2）执行的指令范围不同。管态下可以执行全部指令；用户态不能执行特权指令。

（3）使用资源范围不同。管态可以使用全部系统资源；用户态只能使用用户私有资源，如只能访问自己的存储区域。

2-6 按中断的功能来分，中断有哪几种类型？

答：按中断的功能来分，中断有如下五种类型：

(1) I/O 中断

(2) 外中断

(3) 硬件故障中断

(4) 程序性中断

(5) 访管中断

2-11 什么是程序状态字？在微机中它一般由哪两个部分组成？

答：程序状态字是指反映程序执行时机器所处的运行状态的代码。在微机中它一般由指令计数器 PC 和处理机状态寄存器 PS 组成。

2-12 什么是向量中断？什么是中断向量？

答：向量中断是指当中断发生时，由中断源自己引导处理机进入中断服务程序的中断过程。中断向量就是存储该类型中断服务例行程序的入口地址和处理器状态字的存储单元。

2-18 Linux 系统的中断处理为什么要分为上半部和下半部?

答：操作系统的中断机制实现了 I/O 设备与 CPU 以及多进程之间的同时执行，大大提高了系统效率。

操作系统的中断处理程序比较复杂，而且在系统一级处理时不允许打断。如何提高中断处理的效率？如何解决处理时间短的要求和处理事务复杂性的矛盾？Linux 提出了一个很好的解决办法。Linux 系统将中断处理程序分为两部分，将中断响应后必须立即处理的工作即刻执行（而且其执行时必须关中断），而将更多的处理工作向后推迟执行。即将中断处理程序分为上半部（tophalf）和下半部（bottom half）。Linux 系统将中断处理程序分为上半部和下半部的目的是为了缩短关中断的时间，提高系统的处理能力。

第三章　　习题及解答

3-3 处理应用程序分为哪几个步骤？这些步骤之间有什么关系？

答：处理应用程序分为四个步骤：编辑，编译，连接和运行。这些步骤是相互关联、顺序执行的。具体表现为：

每个步骤处理的结果产生下一个步骤所需要的文件；

每一个步骤能否正确地执行，依赖于前一个步骤是否成功地完成。

3-5 用户与操作系统的接口是什么？一个分时系统提供什么接口？一个批处理系统又提供什么接口？

答：用户与操作系统的接口是操作系统提供给用户与计算机打交道的外部机制。一个分时系统提供的接口有系统功能调用和键盘操作命令。一个批处理系统提供的接口有系统功能调用和作业控制语言。

3-8 什么是系统调用？对操作系统的服务请求与一般的子程序调用有什么区别？

答：系统调用是用户在程序一级请求操作系统服务的一种手段。编程人员利用系统调用，在源程序一级动态请求和释放系统资源，调用系统中已有的系统功能来完成那些与机器硬件部分相关的工作以及控制程序的执行等。因此，系统调用像一个黑箱子那样，对用户屏蔽了操作系统的具体动作而只提供有关的功能。

系统调用与一般过程调用的主要区别如下：（1）程序的性质不同。系统调用服务例程是操作系统程序的一部分，它在核态下执行。而用户子程序是用户程序的一部分，它在用户态下执行。（2）调用方式不同。系统调用是通过陷入到操作系统内核来实现的，调用它们需要中断处理机制来提供系统服务。而子程序调用是在用户程序中直接调用。

3-10 简述系统调用的执行过程。

答：系统调用命令的具体格式因系统而异，但由用户程序进入系统调用的步骤及执行过程大体相同。其执行过程如下：

1．保护用户程序的现场信息，同时把系统调用命令的编号等参数放入指定的存储单元；

2．根据系统调用命令的编号查找系统调用入口表，找到相应系统功能调用子程序的入口地址；

3．转到该子程序执行，当系统调用命令执行完毕，相应的结果通常返回给参数，这些参数放在指定的存储单元里；

4. 系统调用命令执行完毕后恢复用户程序执行的现场信息，同时把系统调用命令的返回参数或参数区首址放入指定的寄存器中，供用户程序使用。

3-12 在 Linux 系统中，增加一个新的系统调用需要做哪些工作?

答：在 Linux 系统中，增加一个新的系统调用需要做的工作包括如下几个方面。

(1) 编写一个新增加的功能的服务例程。编写新增的服务例程，并加到内核中去，即在/usr/src/linux/kernel/sys.c 文件中增加一个新的函数。

(2) 增加一个新的系统调用号。在 linux 中，每个系统调用被赋予一个唯一的系统调用号。找到 linux 中定义系统调用号定义的文件 (在 include/asm-i386/unistd.h 头文件中)。在此文件中按其规定的格式添加一项。

(3) 在系统调用表中登记新的系统调用号以及对应的服务例程。系统调用表记录了内核中所有已注册过的系统调用，它是系统调用的跳转表，实际上是一个函数指针数组，表中依次保存所有系统调用的函数指针。找到 linux 中的系统调用表 (Linux 系统调用表保存在 arch/i386/kernel/下的 entry.S 中)。在此文件中按其规定的格式增加一个新的系统调用号以及对应的服务例程。

(4) 新增加的服务例程要为 Linux 系统接受，必须重新编译内核，生成新的包含新增服务例程的内核。

第四章　习题及解答

4-3 什么是进程？进程与程序的主要区别是什么？

答：进程是一个具有一定独立功能的程序关于某个数据集合的一次活动。进程与程序的主要区别是：

(1) 程序是指令的有序集合，是一个静态概念。进程是程序在处理机的一次执行过程，是一个动态概念。进程是有生命期的，因创建而产生，因调度而执行，因得到资源而暂停，因撤消而消亡；

(2) 进程是一个独立的运行单元，是系统进行资源分配和调度的独立单元，而程序则不是。

(3) 进程与程序之间无一一对应关系。一个程序可以对应多个进程，一个进程至少包含一个程序。

4-4 图 4.2 标明程序段执行的先后次序。其中：I 表示输入操作，C 表示计算操作，P 表示打印操作，下角标说明是对哪个程序进行上述操作。请指明：

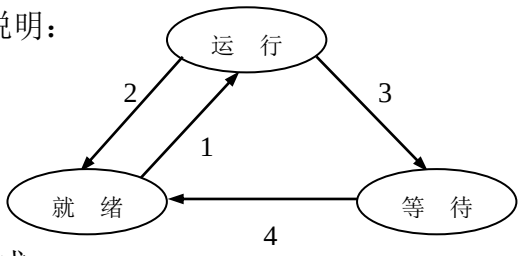
（1） 哪些操作必须有先后次序? 其原因是什么？

（2） 哪些操作可以并发执行? 其原因又是什么？

答：(1) ① I_n、C_n 和 P_n 之间有先后顺序要求，这是由于程序本身的逻辑要求。② 使用同一设备的不同的程序段，如 $C_1 \cdots C_n$，$I_1 \cdots I_n$，$P_1 \cdots P_n$，之间有先后顺序要求，这是由于设备某一时刻只能为一个程序服务。

(2) 不同程序使用不同设备时，占用不同设备，无逻辑关系，可以并发执行，如 I_2 和 C_1；I_3、C_2 和 P_1。

4-9 某系统进程调度状态变迁图如图 4.31 所示，请说明：



(1) 什么原因会导致发生变迁 2、变迁 3、变迁 4 ？

答：发生变迁 2 的原因：时间片到

发生变迁 3 的原因：请求 I/O 或其他系统调用

发生变迁 4 的原因：I/O 完成或其他系统调用完成

(2) 在什么情况下，一个进程的变迁 3 能立即引起另一个进程发生变迁 1 ？

答：一个进程的变迁 3 能立即引起另一个进程发生变迁的条件是，就绪队列非空。

(3) 下列因果变迁是否可能发生？若可能，需要什么条件？

a. 2→1； b. 3→2； c. 4→1

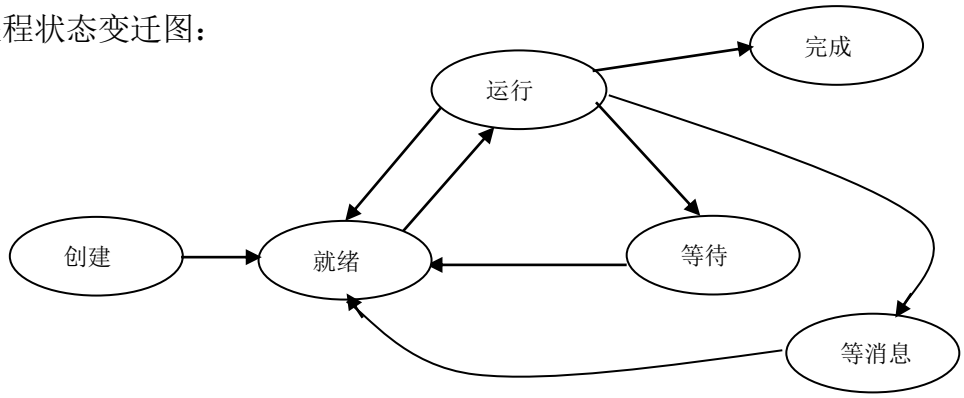
答：a. 2→1 不需要条件，一定会发生。

b. 3→2 不可能发生。

c. 4→1 可能发生，条件：就绪队列为空，或在可剥夺调度方式下，转变为就绪状态的进程优先级最高。

4-10 某系统 jincheng 状态除了 3 个最基本状态外，又增加了创建状态、完成状态、因等消息而转变为等待状态 3 种新的状态，试画出增加新状态后的进程状态变迁图，并说明发生每一个变迁的原因。

答：进程状态变迁图：



进程状态变迁原因：

运行—>等待：请求 I/O； 等待—>就绪：请求完成；

运行—>就绪：时间片到； 就绪—>运行：CPU 空闲，进程调度程序工作；

创建—>就绪：进程创建； 运行—>等消息：等待消息；

等消息—>就绪：收到消息；运行—>完成：任务完成

4-12 n 个并发进程共用一个公共变量 Q，写出用信号灯实现 n 个进程互斥时的程序描述，给出信号灯值的取值范围, 并说明每个取值的物理意义。

解：(1) 程序描述

```
main ( ){
  int mutex=1;  //公共变量 Q 的互斥信号灯
  cobegin
```

P_1; P_2; … P_n;

coend

}

P_i(){

…

P(mutex);

使用 Q；

V(mutex);

…

}

(2) 信号灯值的取值范围：[– (n–1)，1]

(3) mutex 每个取值的物理意义：

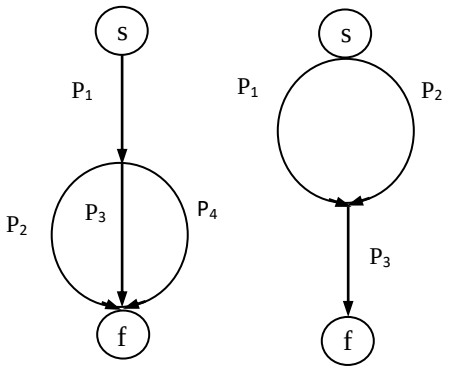
mutex = 1　说明没有进程进入临界段执行；

mutex = 0　说明有一个进程进入临界段执行；

mutex = –1　说明有一个进程进入临界段执行，另有一个进程正在等待进入；

mutex = – (n–1)　说明有一个进程进入临界段执行，另有(n–1)个进程正在等待进入。

4-13 图 4.32(a)、(b) 分别给出了两个进程流图。试用信号灯的 p、v 操作分别实现图 4.32(a)、(b)所示的两组进程之间的同步，并写出程序描述。



(a)　(b)

图 4.32

图(a) 解：

```
main(){
    int s12=0,s13=0,s14=0;   // s12 表示进程 P2 可否执行；s13 表示进程 P3 可否执行；s14 表示进程 P4 可否执行//
    cobegin
        P1; P2; P3; P4;
    coend
}
P1(){
    p1   execute;
    V(s12);
    V(s13);
    V(s14);
}

P2(){
    P(s12);
    p2   execute;
}

P3(){
    P(s13);
    p3   execute;
}

P4(){
    P(s14);
    p4   execute;
}
```

图(b)

```
main(){
    int s13=0,s23=0; // s13 表示进程 P1 执行完成否；s23 表示进程 P2 执行完成否 //
    cobegin
        P1; P2; P3;
    coend
}

P1(){
    P1  execute;
    V(s13);
}

P2(){
    P2   execute;
    V(s23);
}

P3(){
    P(s13);
    P(s23);
    P3  execute;
}
```

4-15 如图 4.34 所示，get、copy 和 put 三个进程共用两个缓冲区 s 和 t (其大小每次存放一个记录) ，get 进程负责不断地把输入记录输入缓冲区 s 中，copy 进程负责从缓冲区 s 中取出记录复制到缓冲区 t 中，而 put 进程负责从缓冲区 t 中取出记录打印。试用 PV 操作实现这三个进程之间的同步，并写出程序描述。

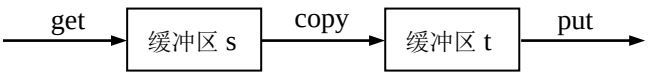


图 4.34

解：

```
main(){
    int s1=1,s2=0; // s1 表示缓冲区 S 是否为空，s2 表示是否已满//
    int s3=1, s4=0; // s3 表示缓冲区 T 是否为空，s4 表示是否已满//
    cobegin
        get;
        copy;
        put;
    coend
}

get(){
    while(1){
        P(s1);
        input data to buffer S;
        V(s2);
    }
}

copy (){
    while(1){
        P(s2);
        copy data from buffer S;
        V(s1);
        P(s3);
        input copy-data to buffer T;
        V(s4);
    }
}
```

```
put(){
    while(1){
        P(s4);
        output data to buffer S;
        V(s3);
    }
}
```

4-16 什么是进程的互斥与同步？同步和互斥这两个概念有什么联系和区别？

答：在操作系统中，当某一进程正在访问某一存储区域时，就不允许其他进程读出或者修改该存储区的内容，否则，就会发生后果无法估计的错误。进程之间的这种相互制约关系称为互斥。

所谓同步，就是并发进程在一些关键点上可能需要互相等待与互通消息，这种相互制约的等待与互通信息称为进程同步。

同步和互斥这两个概念都属于同步范畴，描述并发进程相互之间的制约关系。同步是指并发进程按照他们之间的约束关系，在执行的先后次序上必须满足这种约束关系。而互斥是同步的一种特例，是指并发进程按照他们之间的约束关系，在某一点上一个时刻只允许一个进程执行，一个进程做完了，另一个进程才能执行，而不管谁先做这个操作。

4-22 什么是线程？线程和进程有什么区别?

答：线程也称为轻量级进程，它是比进程更小的活动单位，它是进程中的一个执行路径，一个进程可能有多个执行路径，即线程。

线程和进程的主要区别如下。

(1) 线程是进程的一个组成部分，一个进程可以有多个线程，而且至少有一个可执行的线程。

(2) 进程是资源分配的基本单位，它拥有自己的地址空间和各种资源；线程是处理机调度的基本单位，它只能和其他线程共享进程的资源，而本身并没有任何资源。

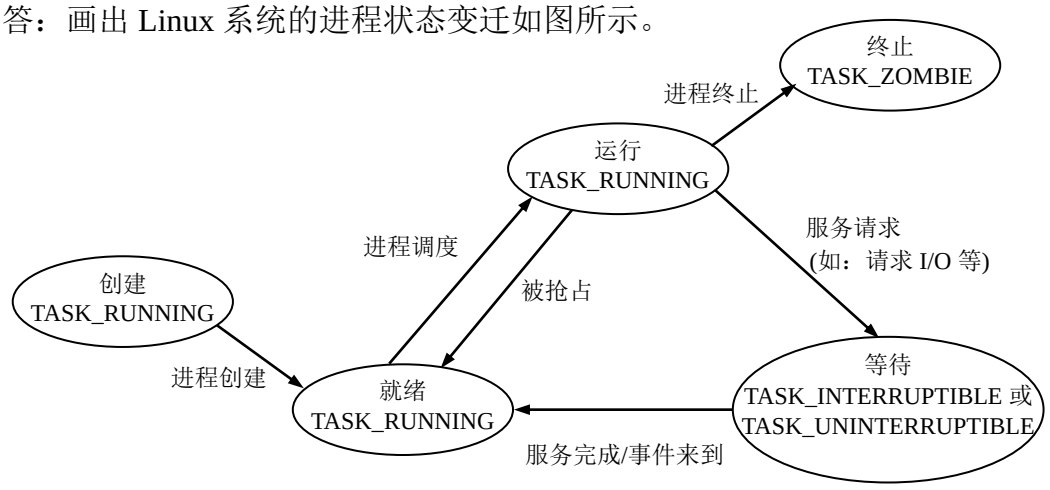
(3) 进程的多个线程都在进程的地址空间内活动。这样，在以线程为单位进行处理机调度和切换时，切换时间较短；而以进程为单位进行处理机调度和切换时，由于涉及到资源转移及现场保护等问题，将导致切换时间变长和资源利用率下降。

(4) 线程和进程一样，都有自己的状态和相应的同步机制，但是，由于线程没有自己单独的程序和数据空间，因而不能像进程那样将程序和数据交换到外存去。

(5) 进程的调度和控制大多由操作系统的内核完成，而线程的控制既可以由操作系统内核完成，也可以由用户控制完成。

4-27 试画出 Linux 系统的进程状态变迁，并说明这些变迁可能的原因。

答：画出 Linux 系统的进程状态变迁如图所示。



(1) 进程创建

当系统或用户需要创建一个新进程时，调用 fork()系统调用，被创建的新进程被置为就绪状态 TASK_RUNNING。

(2) 进程调度

当调度时机来到时，进程调度程序从进程运行队列中选择优先级最高的进程，将其投入运行，设置状态为运行状态。

(3) 被抢占

正在 CPU 上运行的进程，当其优先级低于处于就绪状态的某一个进程的优先级时，它被抢占而被迫让出 CPU 的控制权，此时，该进程的状态转为就绪状态。

(4) 进程等待

若正在运行的进程因等待某一事件而暂时不能运行下去时，进入相应的等待队列，设置为等待状态。

(5) 进程唤醒

当某个进程等待的原因撤销时，该进程被唤醒，将其从等待队列中移出，进入就绪队列。

(6) 进程终止

当正在运行的进程完成其任务时，通过 exit()系统调用终止自己而进入终止状态。

4-29 某公园有一个长凳，其上最多可以坐 5 个人。公园里的游客遵循以下规则使用长凳：

(1) 如果长凳还有空间可以坐，就坐到长凳上休息，直到休息结束，离开长凳。

(2) 如果长凳上没有空间，就转身离开。

试用信号灯的 P、V 操作描述这一场景。

答：

```
main(){
    int count=5; //  长凳上可坐的人数
    int mutex=0; //  访问 count 的互斥信号灯
    cobegin
        guesti;    // i =1,2,3,4……
    coend
}
guesti {
    P(mutex);
    if (count == 0) {
        V(mutex);
        return;
    }
    count--;
    V(mutex);
    休息；
    P(mutex);
    Count++;
```

V(mutex);

}

4-31 6 个进程合作完成一项计算任务的并发描述如图 4-38 所示，程序中，s1、s2、s3、s4、s5、s6 分别是同步信号灯，x、y、z 是共享数据变量。试给出变量 z 的最终结果，并画出 6 个进程合作的进程流图。

```
int s1=s2=s3=0;
int s4=s5=0;
int x=y=z=0;
main()
{
Cobegin
P1();P2();P3();
P4();P5();P6();
Coend
}
```

```
P1() {
x=1;y=1;
V(s1);
V(s1);
}
```

```
P2() {
P(s1);
x=x+2; z=x;
V(s2) ;V(s2);
}
```

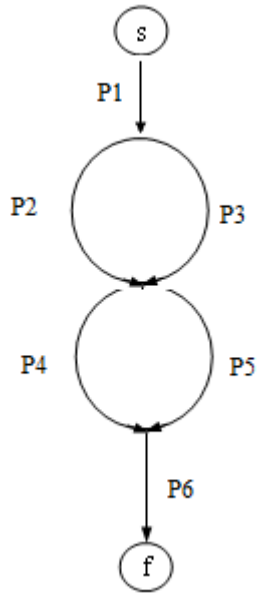
```
P3(){
P(s1);
y=y*2;
V(s3); V(s3);
}
```

```
P4(){
P(s2);
P(s3);
x=x+y;
V(s4);
}
```

```
P5(){
P(s2);
P(s3);
z=z+y;
V(s5);
}
```

```
P6(){
P(s5);
P(s4);
z=x+y+z;
}
```

答：

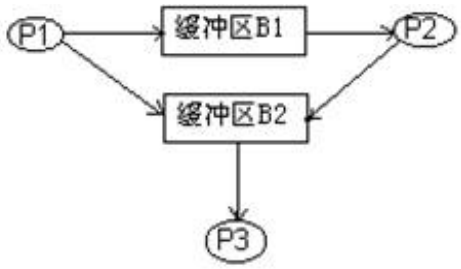


Z=12

4-32 现有 3 个并发进程 P1、P2 和 P3，如图 4.39 所示。3 个并发进程共享两个单缓冲

区 B1 和 B2。进程 P1 负责不断从输入设备读数据，若读入的数据为正数，则直接送入 B2，否则应先将数据送入 B1，经 P2 取出加工后再送入 B2，P3 从 B2 中取信息输出。请使用信号灯和 P、V 操作描述进程 P1、P2、P3 实现同步的算法。

答：缓冲区 B1 设置信号灯 s1、t1，s1 表示缓冲区 B1 是否空闲，t1 表示缓冲区 B1 是否有数据，缓冲区 B2 设置信号灯 s2、t2，s2 表示缓冲区 B2 是否空闲，t2 表示缓冲区 B2 是否有数据。




```
Main()
{
Int s1=s2=1;
Int t1=t2=0;
Cobegin
P1();p2();p3();
Coend
}
```

```
P1() {
While (未结束）
      读取数据；
      If (数据>0) {
      P(s2);
数据放入 B2;
      V(t2);
      } else {
      P(s1);
数据放入 B1;
      V(t1);
      }
```

```
P2()
{
While (未结束) {
P(t1);
取 B1 数据；
V(s1);
P(s2);
数据放入 B2;
V(t2);
}
```

```
P3()
{
While (未结束) {
P(t2);
取 B2 数据；
V(s2);
输出数据；
}
```

4-34 某商场有一个地下停车场，有 N 个停车位，车辆只能通过一个指定的通道进出停车场，通道处只能容一辆车通过，请设计信号灯和 P、V 操作给出进、出车辆两种进程的程序描述。

答：

```
main()
{
    int count1=count2=0;/*计数器*/
     int mutex1=mutex2=1;/*两个计数器的互斥访问信号灯*/
     int mutex=1; /*通道互斥访问信号灯*/
     cobegin
          in();
          out();
```

```
    coend
}
 in()
 {
    P(empty);
    P(mutex1);
    Count1++;
    if   count1==0   P(mutex);
    V(mutex1);
    开车进地下停车场；
    P(mutex1);
    count1--;
    if count1==0 V(mutex);
    V(mutex1);
 }
 out()
 {
    P(mutex2);
    Count2++;
    if   count2==0   P(mutex);
    V(mutex2);
    开车出地下停车场；
    P(mutex2);
    Count2--;
    if count2==0 V(mutex);
    V(mutex2);
    V(empty);
 }
```

第五章　习题及解答

5-5 假设一个可移动磁头的磁盘具有 200 个磁道，其编号为 0~199,当它刚刚结束了 125 道的存取后，现正在处理 143 道的服务请求，假设系统当前的请求序列以请求的先后次序排列如下: 86、147、91、177、150、102、175、130。试问对以下几种磁盘 IO 请求调度算法而言，满足以上请求序列，磁头将分别如何移动?

（1） 先来先服务算法（FCFS）

（2） 最短寻道时间优先调度（SSTF）

（3） 扫描算法（SCAN）

（4） 循环扫描算法（CSCAN）

答：

（1） FCFS：143→86→147→91→177→150→102→175→130；

（2） SSTF：143→147→150→130→102→94→91→86→175→177；

（3） SCAN：143→147→150→175→177→130→102→94→91→86；

（4） C-SCAN：143→147→150→175→177→86→91→94→102→130。

5-9 三个进程共享四个同类资源，这些资源的分配与释放只能一次一个，已知每一进程最多需要两个资源，试问该系统会发生死锁吗？为什么？

答：该系统不会发生死锁。

因为最坏情况是每个进程都占有一个资源，申请第二个资源，而此时系统中还剩一个资源，不管这个资源分给哪个进程，都能满足它的资源要求，因此它能在有限时间内运行结束而释放它所占有的两个资源，这两个资源又可以分配给另外两个进程，使它们能够运行结束，所以系统不会发生死锁。

5-10 p 个进程共享 m 个同类资源，每一个资源在任一时刻只能供一个进程使用，每一进程对任一资源都只能使用一有限时间，使用完便立即释放，并且每个进程对该类资源的最大需求量小于该类资源的数目，设所有进程对资源的最大需求数目之和小于 p+m，试证在该系统中不会发生死锁。

解：采用“反证法”，假定 max(i)为第 i 个进程最大资源需求量，need(i)为第 i 个

进程还需要的资源量，alloc(i)为第 i 个进程已分配的资源量，则

max(i)<=m

max(i)=need(i)+alloc(i)

max(1)+ + max(p)=(need(1)+ …+need(p))+(alloc(1)+ …+alloc(p))<p+m

若发生死锁，则需要满足下面两个条件，

① 全部分配，alloc(1)+…+alloc(p)=m；② 所有进程无限等待

由①②可得， need(1)+…+need(p)<p

则死锁后，p 个进程需要的资源小于 p，则一定存在进程 i，need (i) = 0，进程已获得全部资源，进程 i 可以执行完，同假设发生矛盾，所以不会发生死锁。

5-11 图 5.9 表示一带闸门的运河，其上有两架吊桥，吊桥坐落在一条公路上，为使该公路避开一块沼泽地而其横跨运河两次。运河和公路的交通都是单方向的，运河的基本运输由驳船担负。在一艘驳船接近吊桥 A 时就拉汽笛警告，若桥上无车辆，吊桥就吊起，直到驳船尾部通过该桥为止，对吊桥 B 按同样次序处理

(1) 一艘典型驳船的长度为 200 米，当它在河道航行时是否会产生死锁？若会，其理由是什么？

(2) 如何能克服一个可能的死锁？请想出一个防止死锁的办法。

(3) 如何利用信号灯的 P、V 操作实现车辆和驳船的同步？

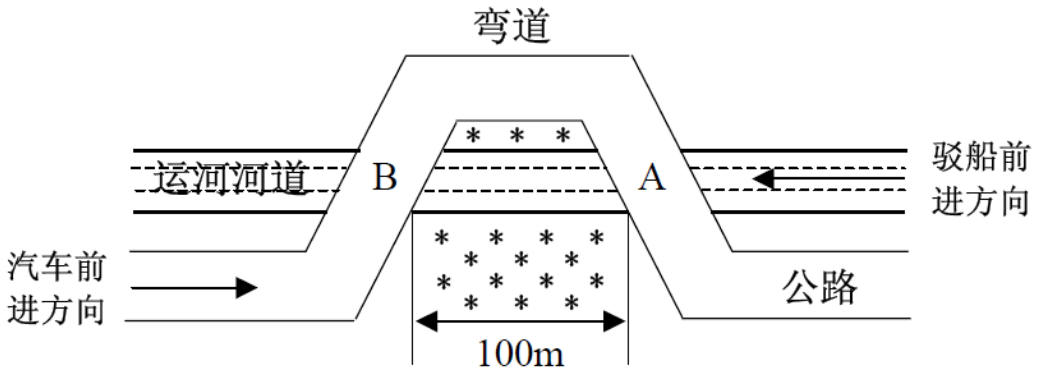


图 5.9

(1) 答：驳船长 200 米，当驳船通过了 A 桥，其船头到达 B 桥，请求 B 桥吊起，而此时它的尾部占据 A 桥，若这个时候 B 桥及 B 桥到 A 桥之间的公路都

被汽车占据，而汽车又要求通过 A 桥。这样驳船和汽车都无法前进，形成死锁的局面。

(2) 答：方案之一。可规定资源按序申请和分配，从而破坏了死锁的循环等待条件，防止死锁的发生。规定如 B 桥的序号小于 A 桥的序号，驳船和汽车都必须先申请序号小的资源 B 桥，申请得到满足后，再申请序号大的资源 A 桥。

(3) 答：将每台车的行驶看作是进程，则有 $Auto_1$，$Auto_2$， $Auto_i$ i 个汽车进程。将每条驳船的航行看作是进程，则有 $Ship_1$，$Ship_2$， $Ship_j$ 个驳船进程。桥 A 和桥 B 对车和船为互斥资源。

方案 1：

```
main{
    int SA=1；//A 桥的互斥信号量//
    int SB=1；//B 桥的互斥信号量//
    cobegin
        Auto1;Auto2;  Autoi;
        Ship1; Ship2;   Shipj;
    coend
}

Autoi(){
    车在公路上行驶；
    P（SB）；
    过 B 桥；
    V（SB）；
    过弯道；
    P（SA）；
    过 A 桥；
    V（SA）；
    车在公路上行驶；
}
```

$Ship_j$(){

运河航行；

P（S_B）；

P（S_A）；

吊起过 A 桥；

运河航行；

吊起过 B 桥；

V（S_A）；

V（S_B）；

运河航行；

}

方案 2：方案 1 的缺点是船在过 A 桥前需要先占用 B 桥，使桥的通过率降低，增加交通阻塞的可能。因此方案 2 将弯道作为有界缓中区，基本思想是只要弯道有空，车就可以通过 B 桥，进入弯道。而船则不用先占用 B 桥，使桥的通过率降低。

main{

int SA=1；//A 桥的互斥信号量

int SB=1；//B 桥的互斥信号量

int Sn=n；//弯道可容纳汽车数 n

cobegin

$Auto_1$;$Auto_2$； $Auto_i$;

$Ship_1$; $Ship_2$； $Ship_j$;

coend

}

$Auto_i$(){

车在公路上行驶；

P（Sn）；

P（SB）；

过 B 桥；

V（SB)；

过弯道；

P（SA）；

上 A 桥；

V（Sn）；

过 A 桥；

V（SA）；

车在公路上行驶；

}

$Ship_j$(){

运河航行；

P（SA）；

船头行驶至 B 桥；

P（SB）；

运河航行；

船尾过 A 桥；

V（SA）；

船尾过 B 桥；

V（SB）；

运河航行；

}

5-14 在采用银行家算法管理资源分配的系统中，有 A、B、C 三类资源可供 5 个进程 P1、P2、P3、P4、P5 共享。3 类资源的总量为(17, 5, 20)，即 A 类 17 个，B 类 5 个，C 类 20 个。假设 T_0 时刻各进程对资源的需求和分配情况如下表所示。

表 5.2　T_0时刻各进程对资源的需求和分配情况

进　程	最大需求数			已占有资源		
	A	B	C	A	B	C
P1	5	5	9	2	1	2
P2	5	4	6	4	0	2
P3	4	0	11	4	0	5
P4	4	2	5	2	0	4
P5	8	2	4	3	1	4

（1） 现在系统是否处于安全状态？如是，给出一个安全序列。

（2） T_0时刻，如果进程 P4 和 P1 依次提出 A、B、C 资源请求（2,0,1）和（0,2,0），系统能否满足它们的请求?请说明原因。

答：（1）系统处于安全状态，如 P4→P2→P3→P5→P1。

（2）不能满足。由于 P4 与 P1 提出请求后，A、B、C 剩余（0，1，2），此时 A 类无，只能等待拥有足够 A 类资源的进程结束释放 A 类资源，别的进程才能执行，而此时 P4 需（0，2，0），P3 需（0，0，6），而剩余（0，1，2），不能满足要求，产生死锁。

第六章习题及答案

6-3 某系统的进程状态变迁图如图 6.12 所示（设该系统的进程调度方式为非剥夺方式)。

(1) 说明一个进程发生变迁 3 的原因是什么？发生变迁 2、变迁 4 的原因又是什么？

(2) 下述因果变迁是否会发生，如果有可能的话，在什么情况下发生 ？

① 2→5； ② 2→1； ③ 4→5； ④ 4→2； ⑤ 3→5

(3) 根据此进程状态变迁图叙述该系统的调度策略、调度效果。

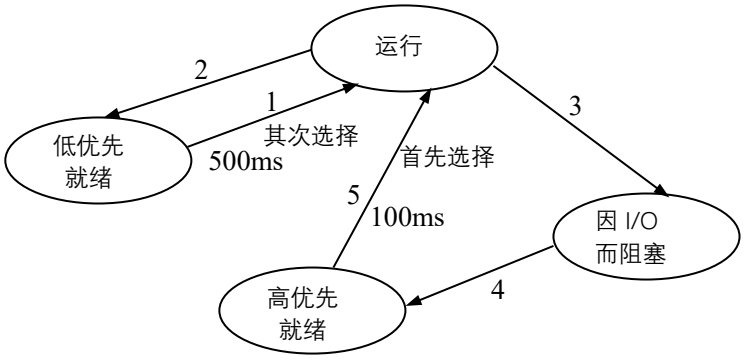


图 6.12

(1) 答：

发生变迁 3 的原因：当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行时会发生变迁 3。

发生变迁 2 的原因：运行进程在分得的时间片 100ms 或 500ms 内未完成，当其时间片到时将发生变迁 2。

发生变迁 4 的原因：当等待进程等待的事件发生了，将会发生变迁 4。

(2) 答：

① 2→5 的因果变迁可能发生。条件是：高优先就绪队列非空。

② 2→1 的因果变迁可能发生，当运行进程的时间片到时发生的变迁 2，若此时高优先就绪队列为空，必然引起低优先就绪队列中的一个就绪进程被调度执行而发生变迁 1。

③ 4→5 的因果变迁不可能发生，因为采用的是非剥调度夺式。

④ 4→2 的因果变迁不可能发生。

⑤ 3→5 的因果变迁可能发生，条件是：高优先就绪队列非空。

(3) 答：

调度策略：首先调度高就绪队列中的进程 (一般是 I/O 型进程) 投入运行，给高优先就绪队列中的进程分配的时间片大小为 100ms。只有当高就绪队列中的所有进程全部运行完或因等待某事件发生处于阻塞状态，高就绪队列中没有进程可运行时，才调度低优先就绪队列中的进程 (一般是计算型进程)，给低优先就绪队列中的进程分配的时间片大小为 500ms。若一个运行进程时间片 100ms 或 500ms 到时未完成就进入低优先就绪队列。若某进程在运行期间因等待某事件发生而进入阻塞队列，则当所等待事件完成后，它将进入高优先就绪队列。

调度效果：这种算法优先照顾了 I/O 量大的进程 (高优先级)，但通过给计算型进程分配更长的时间片也适当照顾了计算型进程。

6-10 Linux2.6 版本为了实现 O(1)级算法复杂度，采用了什么措施?

答：Linux 系统进程调度用的数据结构最重要的是运行队列结构，该结构给出了处理机上可运行进程的链表。该结构中包含一个称为优先级数组的结构数组。每个数组都表示一个可运行进程集合，包括两个重要信息：① 一个优先级位图；② 140 个双向链表头，每个链表对应一个可能的进程优先级队列。

Linux 系统采用优先调度策略。在 Linux2.6 版本的进程调度程序中，基于上述进程调度用数据结构，查找系统中优先级最高的进程这一问题转化为查找优先级位图中第一个置为 1 的位。找到这一位就是找到了最高优先级链表，即可确定优先级最高的、可运行的进程。由于优先级个数是定值，所以查找时间恒定。许多体系结构提供 find_first_bit 指令(字操作指令)，找到第一个设置为 1 的位所花费的时间微不足道。这是保证 Linux 系统进程调度具有 O(1)级算法复杂度的关键所在。

第七章　习题及解答

7-11 如图 7.45 所示，主存中有两个空白区。现有如下程序序列：程序 1 要求 50KB；程序 2 要求 60KB；程序 3 要求 70KB。若用首次适应算法和最佳适应算法来处理这个程序序列，试问：哪一种算法可以分配得下 ？简要说明分配过程 (假定分区描述器所占用的字节数已包含在程序所要求的主存容量中) 。

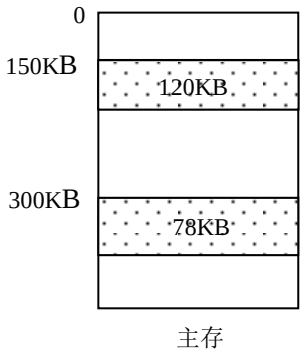


图 7.45

答：(1) 首次适应法：

程序 1 要求 50KB，在起始地址为 150KB，大小为 120 KB 的空白区进行分割。120KB–50KB=70KB，分割后剩 70KB 的空白区。

程序 2 要求 60KB，在剩余的 70KB 空白区进行分割。70KB–60KB=10KB，分割后剩 10KB 的空白区。

程序 3 要求 70KB，在起始地址为 300KB，大小为 78KB 的空白区进行分割。78KB–70KB=8KB，分割后剩 8KB 的空白区。

因此首次适应法可满足该程序序列的需求。

(2) 最佳适应法

程序 1 要求 50KB，在起始地址为 300KB，大小为 78 KB 的空白区进行分割。78KB–50KB=28KB，分割后剩 28KB 的空白区。

程序 2 要求 60KB，在起始地址为 150KB，大小为 120KB 的空白区进行分割。120KB–60KB=60KB，分割后剩 60KB 的空白区。

程序 3 要求 70KB，。此时系统中有大小为 28KB 和 60KB 的两个空白区，它们均不能满足程序 3 的需求。

因此最佳适应法不能满足该程序序列的需求。

7-12 已知主存有 256KB 容量，其中 OS 占用低址 20KB，可以有这样的一个程序序列。

程序 1 要求 80KB；程序 2 要求 16KB；程序 3 要求 140KB。

程序 1 完成；程序 3 完成。

程序 4 要求 80KB；程序 5 要求 120KB。

试分别用首次适应算法和最佳适应算法分别处理上述程序序列 (在存储分配时，从空白区高址处分割作为已分配区)，并完成以下各步骤。

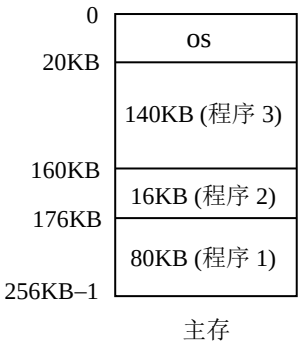
(1) 画出程序 1、2、3 进入主存后主存的分配情况。

(2) 画出程序 1、3 完成后主存分配情况。

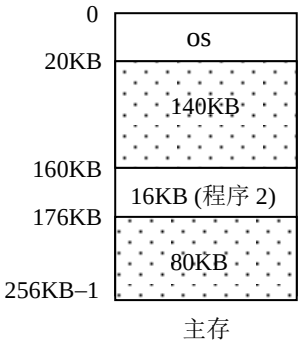
(3) 试用上述两种算法中画出程序 1、3 完成后的空闲区队列结构 (要求画出分区描述器信息，假定分区描述器所需占用的字节数已包含在程序所要求的主存容量中) 。

(4) 哪种算法对该程序序列而言是适合的？简要说明分配过程。

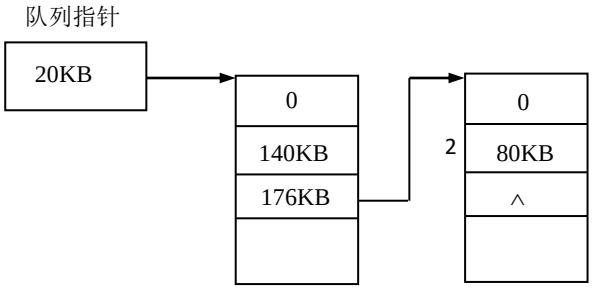
(1) 答：程序 1、2 和 3 进入主存后，主存的分配情况如下图所示。



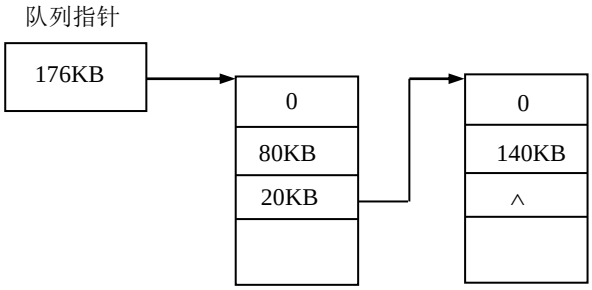
(2) 答：程序 1、3 完成后，主存的分配情况如下图所示：



(3) 答：首次适应法下，空闲区队列结构如下图所示。



首次适应法下，空闲区队列结构如下图所示。



(4) 答：程序 4 要求　80KB；程序 5 要求 120KB。

首次适应法：

程序 4 要求 80KB，在起始地址为 20KB，大小为 140 KB 的空白区进行分割。140KB−80KB=60KB，分割后剩 60KB 的空白区。

程序 5 要求 120KB，此时系统中有大小为 60KB 和 80KB 的两个空白区，它们均不能满足程序 5 的需求。

因此首次适应法不能满足该程序序列的需求。

最佳适应法：

程序 4 要求 80KB，在起始地址为 176KB，大小为 80 KB 的空白区进行分割。80KB−80KB=0KB，正好装下程序 4。

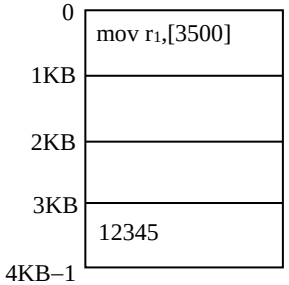
程序 5 要求 120KB，在起始地址为 20KB，大小为 140 KB 的空白区进行分割。140KB−120KB=20KB，分割后剩 20KB 的空白区。

因此最佳适应法能满足该程序序列的需求。

7-14 已知主存容量为 64K 字节，某一程序 A 的地址空间如图 7.46 所示，它的 4 个页面 (页面大小为 1KB 字节) 0、1、2、3 被分配到主存的 2、4、6、7 块中。

(1) 画出 A 的页面映像表；

(2) 当 200 号单元处有一条指令“mov r_1，[3500]”执行时，如何进行正确的地址变换，以使 3500 处的内容 12345 装入 r_1 中 ？



程序 2 地址空间

图 7.46

(1) 答：程序 A 的页面映射表如下图所示。

页号	块号
0	2
1	4
2	6
3	7

(2) 答：每页大小为 1KB=1024 字节，而 3500=3×1024+428，可知逻辑地址 3500 对应的页号为 3，页内地址为 428，根据页号检索页表可知对应的物理块号为 7，所以物理地址为 7×1024+428=7596。

第八章　习题及解答

8-1 什么是设备独立性？引入这一概念有什么好处？

答：所谓设备独立性是指，用户在编制程序时所使用的设备同实际使用的设备无关，也就是在用户程序中仅使用逻辑设备。

引入设备独立性，可使应用程序独立于物理设备。此时，用户编程只需用逻辑设备去请求使用某类设备。当系统中有多台该类设备时，系统将其中的任一台备分配给请求进程，而不必局限于某一指定设备。这样，可以显著地提高资源的利用率和可适应性。

独立性还可以使用户程序独立于设备类型。例如，在进行输出时，既可以利用显示终端进行输出，也可以利用打印机进行输出。有了这种适应性，就可以很方便地实现输出重定向，类似地可以实现输入重定向。

8-4 什么是缓冲？引入缓冲的原因是什么？

答：缓冲是两种不同速度的设备之间传输信息时平滑传输过程的常用手段。

引入缓冲技术的原因有如下几点。

(1) 缓和 CPU 和 I/O 设备之间速度不匹配的矛盾。

(2) 减少中断次数和 CPU 的中断处理时间。如果没有缓冲，慢速 I/O 设备每传一个字节就要产生一个中断，CPU 必须处理该中断；如果采用了缓冲，则慢速 I/O 设备将缓冲填满时，才向 CPU 发出中断，减少了中断次数和 CPU 的中断处理时间。

(3) 解决 DMA 或通道方式下数据传输的瓶颈问题。DMA 或通道方式都用于成批数据传输，在无缓冲的情况下，慢速 I/O 设备只能一个字节一个字节的传输信息，成了 DMA 或通道方式数据传输的瓶颈。缓冲的设置适应了 DMA 或通道方式的成批数据传输方式，解决了数据传输的瓶颈问题。

8-5 常用的缓冲技术有哪些？

答：常用的缓冲技术有双缓冲、环形缓冲和缓冲池。

引入双缓冲以提高处理机与 I/O 设备之间的并行操作程度，例如，输入设备备先将第一个缓冲装满数据，在输入设备向第二个缓冲装数据时，处理机就可以

从第一个缓冲中取出数据进行处理。第一个缓冲的数据处理完毕，若第二个缓冲已经装满数据，则处理机又可以从第二个缓冲中取出数据进行行处理，而输入设备又向第一个缓冲装填数据。

为了在 CPU 与 外设对信息的操作速度相差甚远时仍能得到良好并行效果，可以采用环形缓冲技术。环形缓冲技术是在主存中分配一组大小相等的存储区作为缓存区，并将这些缓存区链接起来，每个缓存区中有一个指向下一个缓存区的指针，最后一个缓存区的指针指向第一个缓存区，这样 n 个缓存区就成了一个环形缓冲外，系统中有个缓冲链首指针指向第一个缓存区。环形缓冲用于输入输出时，需要两个指针 in 和 out，in 指向第一个空缓存区，out 指向第一个装满数据的缓存区。输入时，把数据输入到 in 所指的空缓存区中，然后 in 模取后移一位，指向下一个空缓存区。输出时， 从 out 所指的满缓存区中取出数据，然 out 模取后移一位，指向下一个满缓存区。

缓冲池是由若干个大小相等的缓存区组成的。缓冲池中的每一个缓存区都由系统统一管理和动态分配。若某个进程需要使用缓冲时便提出申请，由系统将缓存区分配给它，进程不再使用缓存区时，就将缓存区交还给缓冲池。这样，就可以用少量的缓存区服务更多的进程。缓冲池通常将缓存区排成 3 个队列：空闲缓存区队列、输入缓存区队列和输出缓存区队列。

8-8 什么是独占设备？对独占设备如何分配？

答：独占设备是指在一段时间内只允许一个用户进程访问的设备。系统一旦把这类设备分配给某进程后，便由该进程独占直到使用完后释放。多数低速 I/O 设备都属于独占设备，如打印机等。

独占设备采用独占分配方式，即将一个独占设备分配给某进程后便一直由它独占，直到该进程完成或释放该设备时，系统才能将该设备分配给其他进程。

8-9 什么是共享设备？对共享设备如何分配？

答：共享设备是指在一段时间内允许多个进程同时访问的设备，如磁盘。对共享设备可将其同时分配给多个进程，使用共享分配方式显著提高了设备的利用率，但对设备的访问需进行合理的调度。

8-10 什么是虚拟设备技术？什么是虚拟设备？如何进行行虚拟分配？

答：所谓虚拟设备技术，是在一类物理设备上模拟另一个物理设备的技术，是将独占设备转换成共享设备的技术。目前广泛流行的虚拟设备技术是 SPOOLing 技术，网络环境中的虚拟打印机。

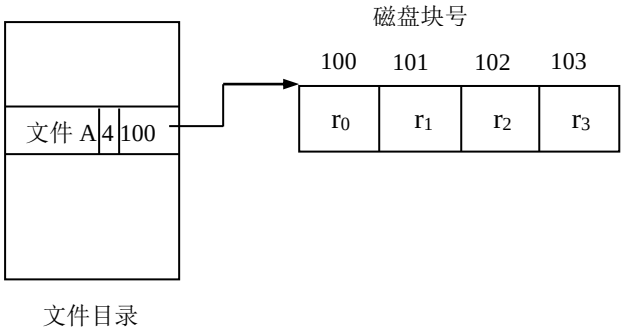
虚拟设备是指通过虚拟技术将一独占设备变换成若干台逻辑设备，供若干个用户进程使用，通常把这种经过虚拟技术处理的设备称为虚拟设备。引入虚拟设备的目的是为了克服独占设备速度较慢、资源利用率较低的缺点，以提高设备的利用率。

虚拟分配是针对虚拟设备而言的。当进程申请独占设备时，由系统分配给它共享设备，如磁盘的一部分存储空间。当进程要和设备交换信息，系统就将要交换的信息放到这部分存储空间中，在合适的时候，系统再将存储空间中的信息传到独占设备。

第九章　习题及解答

9-5 设文件 A 按连续文件构造，并由四个逻辑记录组成 (每个逻辑记录的大小与磁盘块大小相等，均为 512B) 。若第一个逻辑记录存放在第 100 号磁盘块上，试画出此连续文件的结构。

答：连续文件的结构如下图：



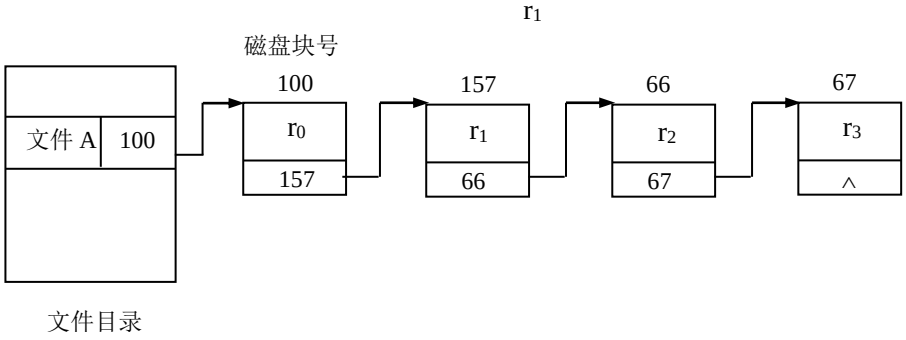
9-6 设文件 B 按串联文件构造，并由四个逻辑记录组成 (其大小与磁盘块大小相等，均为 512B)。这四个逻辑记录分别存放在第 100、157、66、67 号磁盘块上，回答如下问题。

(1) 画出此串联文件文件的结构，

(2) 若要读文件 B 第 1560 字节处的信息，问要访问哪一个磁盘块？为什么？

(3) 读文件 B 第 1560 字节处的信息需要进行多少次 I/O 操作？为什么？

(1) 答：此串联文件结构如下图所示。



(2) 答：1560/512=3 余 24，因此文件第 1560 逻辑字节在 r_3 逻辑块上，该逻辑块被分配在 67 号磁盘块上。

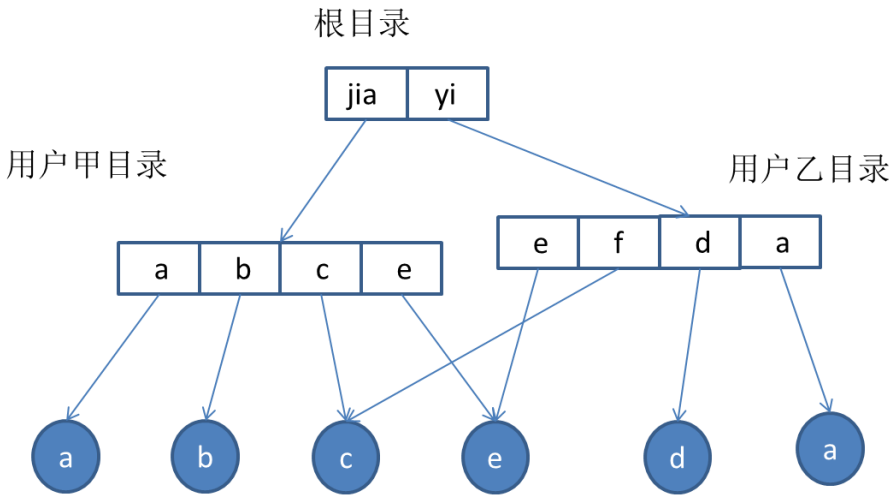
(3) 答：要访问 67 号磁盘块，需要先找到文件目录，然后依次访问 100、157 和 66 号磁盘块，最后读取 67 号磁盘块。因此若文件已打开 (文件目录信息已在内存中) 需要 4 次 I/O 操作，文件未打开需要 5 次 I/O 操作。

9-16 什么是“重名”问题 ？二级文件目录结构如何解决这一问题?

答：重名是指不同用户对不同文件起了相同的名字。在二级文件目录结构中，每个用户建立用户文件目录，系统建立主目录，登记所有用户目录的信息，用目录名加文件名唯一标识每个文件解决重名问题。

9-18 假设两个用户共享一个文件系统，用户甲要用到文件 a、b、c、e，用户乙要用到文件 a、d、e、f。已知：用户甲的文件 a 与用户乙的文件 a 实际上不是同一文件；用户甲的文件 c 与用户乙的文件 f 实际上是同一文件；甲、乙两用户的文件 e 是同一文件。试拟定一个文件组织方案，使得甲、乙两用户能共享该文件系统而不致造成混乱。

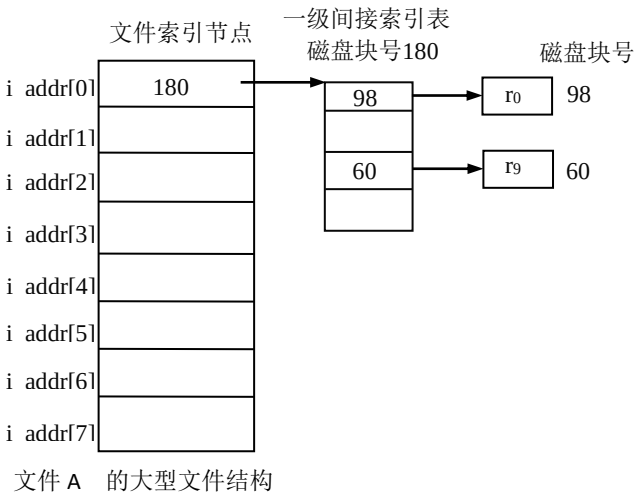
答：如下图所示。用户甲的主目录名为 jia，有四个文件，文件名为 a、b、c、e。用户乙的主目录名为 yi，有四个文件，文件名为 a、d、e、f。



9-27 设某文件 A 有 10 个逻辑块，另一文件 B 有 500 个逻辑块，试分别用 UNIX 7 版本的索引结构画出这两个文件的索引结构图。

答：文件 A 有 10 个逻辑块：按 UNIX 7 版本的索引结构则要构造大型文件结构。这时数组 i_addr[ ]用作一级间接索。

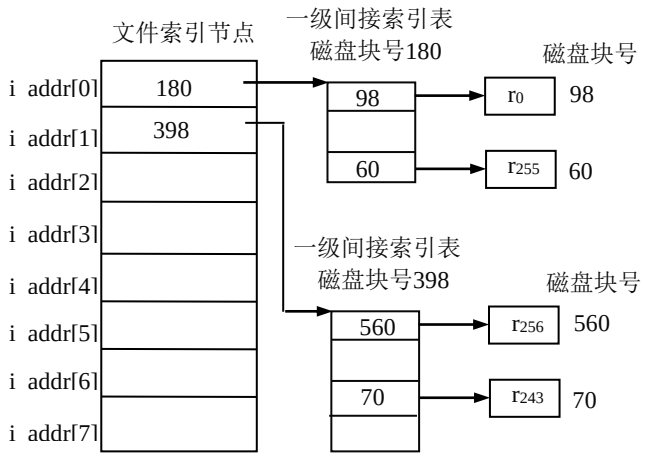
文件 A 的文件索引结构如下图所示（ 磁盘块号随意设置 ）。



文件 A 的大型文件结构

文件 B 有 500 个逻辑块：按 UNIX 7 版本的索引结构则要构造大型文件结构。这时数组 i_addr[ ]用作一级间接索。

500/256 = 1，余 244　文件 B 的文件索引结构如下图所示（磁盘块号随意设置）。



文件 B 的大型文件结构

9.38 某文件系统中，辅存为硬盘，物理块大小为 512B。有文件 A，包含 590 个逻辑记录，每个记录占 255B，每个物理块存放 2 个逻辑记录。文件 A 所在的目录如图 9.32 所示，此树型目录结构由根目录节点、作为目录文件的中间节点和作为信息文件的叶节点组成。每个目录项占 127B，每个物理块放 4 个目录项，根目录的第一块常驻主存。回答如下问题：

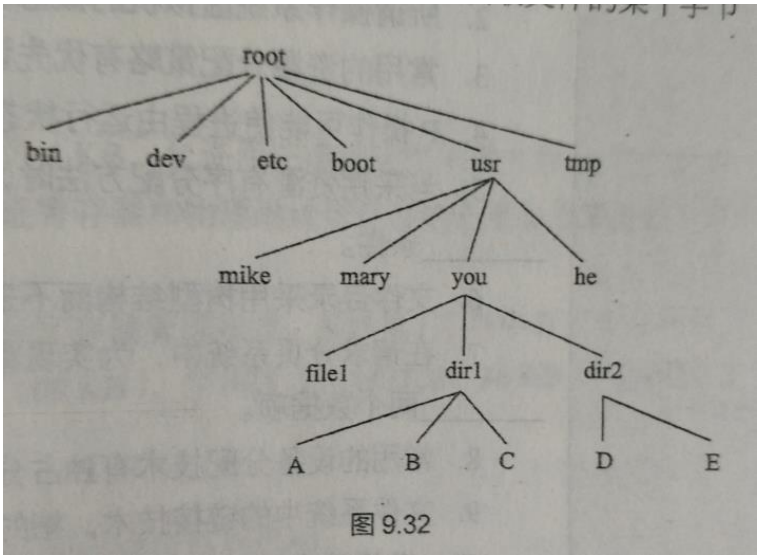


图 9.32

(1)若文件采用串连结构，链接字占 2B，那么，要将 A 读入主存，至少要存取几次硬盘?为什么?

(2) 若文件采用连续结构，那么，要将 A 的第 480 号记录读入主存，至少要存取几次硬盘?为什么?

答：（1）第一次读根目录的第二块， 获得 usr 的地址；

第二次读 usr 目录的第一块，获得 you 的地址；

第三次读 you 目录的第一块，获得 dir1 的地址；

第四次读 dir1 目录的第一块，获得 A 的地址；

第五次开始遍历串联的 A 的逻辑记录，590 个逻辑记录占用 295 个磁盘块，需要存取 295 次。

（2）连续结构可随机访问，第四次获得 A 的地址后就可计算得到 A 的第 480 号记录的地址，然后直接第五次访问该记录。

9-39 某系统采用成组链接法来管理系统盘的空闲存储空间， 目前，磁盘的状态如图 9.33 所示。回答如下问题:

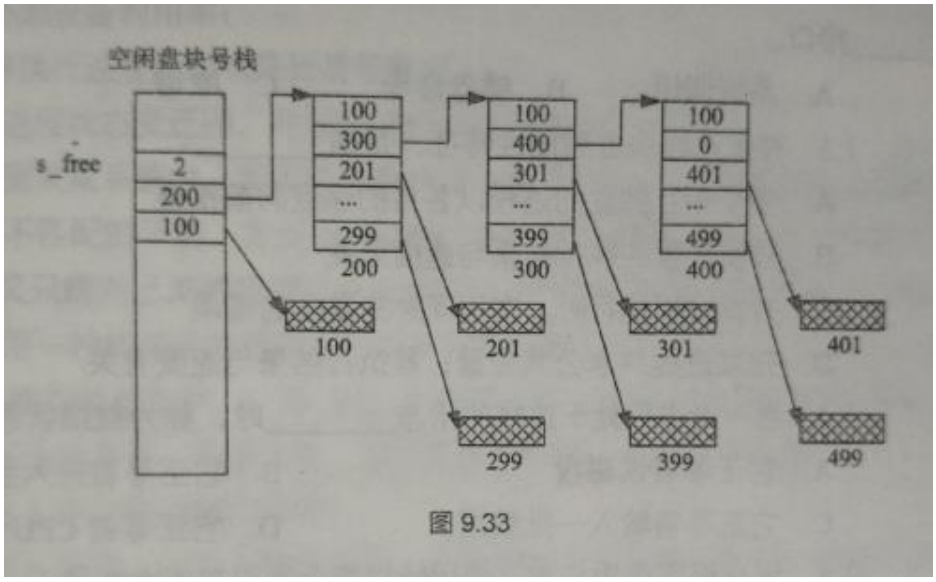


图 9.33

(1)该磁盘中目前还有多少个空闲盘块?

(2)系统需要给文件 F 分配 3 个磁盘块，试给出将被分配出去的磁盘块号。

(3)接着(在创建文件 F 之后),系统要删除另一个文件，并回收它所占的 5 个盘块，它们的盘块号依次为 700、711、703、788、701，试给出回收后的盘块链接情况。

答：

（1）301 个空闲盘块。

（2）100，200，299。

（3）s_free=4, s_free[0]=711, s_free[1]=703 s_free[2]=788 s_free[3]=701，原 299 所在位置存放盘块号 700。