清华大学计算机系列教材

计算机操作系统教程

第2版

习题解答与实验指导

张尧学 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

清华大学计算机系列教材

计算机操作系统教程

（第 2 版）

习题解答与实验指导

张尧学　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是作者在清华大学计算机系多年教学和科研的基础上，配合清华大学计算机系列教材之一的《计算机操作系统教程》(第 2 版)而编写的相关习题解答和实验指导。全书分为两大部分：第一部分是《计算机操作系统教程》(第 2 版)中各章习题的参考解答和部分硕士研究生考试用习题及解答；第二部分为清华大学计算机系操作系统课程教学用实验指导及相应的程序设计与源代码分析。实验主要设计在 Linux 环境下用 C 语言编程完成，但也可在 UNIX System V 或其他更高版本的 UNIX 环境下完成。

本书既可作为计算机专业和其他相关专业操作系统课程的补充教材，也可供有关人员自学，或供操作系统等系统设计人员阅读和参考。

书　名：计算机操作系统教程(第 2 版)习题解答与实验指导
作　者：张尧学　编著
出版者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者：北京国马印刷厂
发行者：新华书店总店北京发行所
开　本：787×1092 1/16　印张：9.25　字数：210 千字
版　次：2000 年 8 月第 1 版　2000 年 8 月第 1 次印刷
书　号：ISBN 7-302-04004-4/TP・2350
印　数：0001～6000
定　价：11.00 元

序　言

计算机技术的飞速发展正在引发新的一轮世界性技术革命。在经济发展越来越全球化、科技创新越来越国际化、知识经济已初见端倪的今天，任何一门技术或任何一个领域离开了计算机恐怕是不可想象的。然而，计算机技术发展之迅速、计算机及其相关 IT 产品市场竞争之激烈、计算机产业让人致富速度之迅猛也同样是人们始料不及的。在新世纪，任何想在技术领域有一番作为的人，恐怕都不得不面对计算机技术的挑战。

软件技术是计算机系统的灵魂与核心，而操作系统更是计算机系统的大脑。“想发财，学软件！”在一些国家已成为深入人心的广告词。在我国，科技创新、高科技产业化的浪潮也势必会以雷霆万钧之力推动软件技术的迅猛发展与普及。21 世纪的哪一行哪一业能够离开软件呢？

学习计算机软件技术，特别是计算机操作系统技术，除了需要刻苦努力外，还需要掌握软件和操作系统的原理与设计技巧。这些原理与技巧可以说是计算机界的前辈们一代接一代不停顿的努力所留下的知识与智慧的结晶，学习和掌握它们对于激发自己的创造力和想象力是很有帮助的。

如何学习和掌握操作系统技术的原理与实际技巧呢？除了听课和读书之外，最好的方法恐怕就是在实践中练习。例如，自己设计一个小型操作系统，多使用操作系统，多阅读和分析操作源代码等。当前非常流行的 Linux 操作系统的原始版事实上也是一位优秀的大学生的练习之作。除了上述练习方法之外，习题和实验也是很重要的实践之一。

本书就是一本配合《计算机操作系统教程》(第 2 版)的习题解答与实验指导书。本书除给出《计算机操作系统教程》(第 2 版)各章所附习题的参考答案外，还给出一些相应的综合试题及其参考答案；另外还设计了 4 个在 Linux 环境下或 UNIX System V 以上版本的 UNIX 环境下的小实验，包括进程控制、进程通信、内存管理以及文件系统设计等，并给出了这 4 个实验的参考编程解答。

本书的编写得到了清华大学计算机系网络系统组的博士生王晓春、马洪军以及宋建平和段小平等同志的大力帮助和支持。他们对本书中所给出的许多习题进行了解答和完善，尽管作者在讲课过程中已多次讲解过这些习题的解答思路。在这里，作者向这些同志表示衷心的感谢！

本书虽然给出了《计算机操作系统教程》(第 2 版)一书中习题的参考解答和相关实验指导，但由于作者的水平与知识所限，这些解答只是一种参考，里面完全可能存在错误和不妥之处，有待于有识之士的指教。此外，还希望读者不要局限于这些解答。

衷心希望本书能对学习计算机操作系统和计算机软件的人们有所帮助！

作　者

2000 年 3 月于清华园

目　　录

第一部分　习题解答

第1章　绪　　论

1. 什么是操作系统的基本功能？

答：操作系统的职能是管理和控制计算机系统中的所有硬、软件资源，合理地组织计算机工作流程，并为用户提供一个良好的工作环境和友好的接口。操作系统的基本功能包括：处理机管理、存储管理、设备管理、信息管理（文件系统管理）和用户接口等。

2. 什么是批处理、分时和实时系统？各有什么特征？

答：批处理系统（batch processing system）：操作员把用户提交的作业分类，把一批作业编成一个作业执行序列，由专门编制的监督程序（monitor）自动依次处理。其主要特征是：用户脱机使用计算机、成批处理、多道程序运行。

分时系统（time sharing operation system）：把处理机的运行时间分成很短的时间片，按时间片轮转的方式，把处理机分配给各进程使用。其主要特征是：交互性、多用户同时性、独立性。

实时系统（real time system）：在被控对象允许时间范围内作出响应。其主要特征是：对实时信息分析处理速度要比进入系统快、要求安全可靠、资源利用率低。

3. 多道程序（multiprogramming）和多重处理（multiprocessing）有何区别？

答：多道程序（multiprogramming）是作业之间自动调度执行、共享系统资源，并不是真正地同时执行多个作业；而多重处理（multiprocessing）系统配置多个CPU，能真正同时执行多道程序。要有效使用多重处理，必须采用多道程序设计技术，而多道程序设计原则上不一定要求多重处理系统的支持。

4. 讨论操作系统可以从哪些角度出发，如何把它们统一起来？

答：讨论操作系统可以从以下角度出发：（1）操作系统是计算机资源的管理者；（2）操作系统为用户提供使用计算机的界面；（3）用进程管理观点研究操作系统，即围绕进程运行过程来讨论操作系统。

上述这些观点彼此并不矛盾，只不过代表了同一事物（操作系统）站在不同的角度来看待。每一种观点都有助于理解、分析和设计操作系统。

5. 写出 1.6 节中巡回置换算法的执行结果。

答：1.6 节中的巡回置换算法要求：

设　i = 1,2,3,4,5,6,7

　　p[i] = 4,7,3,1,2,5,6

当　k∈[1…n]

　　k = P[…, p[k], …]。

从而有如下解：

(1) 算法

```
local x, k          /* x, k 为局部变量 */
Begin  k←1          /* 初始化 k */
  while  k<=7  do
    x←k
    repeat
      print(x)
      x←p[x]
    until x=k
    k←k+1
  od
End
```

(2) 打印结果：

k=1 时，置换过程为(1 4 1)

k=2 时，置换过程为(2 7 6 5 2)

k=1 时，置换过程为(3 3)

k=1 时，置换过程为(4 1 4)

k=1 时，置换过程为(5 2 7 6 5)

k=1 时，置换过程为(6 5 2 7 6)

k=1 时，置换过程为(7 6 5 2 7)

6. 设计计算机操作系统与哪些硬件器件有关？

答：计算机操作系统的重要功能之一是对硬件资源的管理。因此设计计算机操作系统时应考虑下述计算机硬件资源：

(1) CPU 与指令的长度及执行方式；

(2) 内存、缓存和高速缓存等存储装置；

(3) 各类寄存器，包括各种通用寄存器、控制寄存器和状态寄存器等；

(4) 中断机构；

(5) 外部设备与 I/O 控制装置；

(6) 内部总线与外部总线；

(7) 对硬件进行操作的指令集。

第 2 章　操作系统用户界面

1. 什么是作业？作业步？

答：把在一次应用业务处理过程中，从输入开始到输出结束，用户要求计算机所做的有关该次业务处理的全部工作称为一个作业。作业由不同的顺序相连的作业步组成。作业步是在一个作业的处理过程中，计算机所做的相对独立的工作。例如，编辑输入是一个作业步，它产生源程序文件；编译也是一个作业步，它产生目标代码文件。

2. 作业由哪几部分组成？各有什么功能？

答：作业由三部分组成：程序、数据和作业说明书。程序和数据完成用户所要求的业务处理工作，作业说明书则体现用户的控制意图。

3. 作业的输入方式有哪几种？各有何特点？

答：作业的输入方式有 5 种：联机输入方式、脱机输入方式、直接耦合方式、SPOOLING (Simultaneous Peripheral Operations Online) 系统和网络输入方式，各有如下特点：

(1) 联机输入方式：用户和系统通过交互式会话来输入作业。

(2) 脱机输入方式：又称预输入方式，利用低档个人计算机作为外围处理机进行输入处理，存储在后援存储器上，然后将此后援存储器连接到高速外围设备上和主机相连，从而在较短的时间内完成作业的输入工作。

(3) 直接耦合方式：把主机和外围低档机通过一个公用的大容量外存直接耦合起来，从而省去了在脱机输入中那种依靠人工干预来传递后援存储器的过程。

(4) SPOOLING 系统：可译为外围设备同时联机操作。在 SPOOLING 系统中，多台外围设备通过通道或 DMA 器件和主机与外存连接起来，作业的输入输出过程由主机中的操作系统控制。

(5) 网络输入方式：网络输入方式以上述几种输入方式为基础，当用户需要把在计算机网络中某一台主机上输入的信息传送到同一网中另一台主机上进行操作或执行时，就构成了网络输入方式。

4. 试述 SPOOLING 系统的工作原理。

答：在 SPOOLING 系统中，多台外围设备通过通道或 DMA 器件和主机与外存连接起来，作业的输入输出过程由主机中的操作系统控制。操作系统中的输入程序包含两个独立的过程，一个过程负责从外部设备把信息读入缓冲区，另一个过程是写过程，负责把缓冲区中的信息送入到外存输入井中。

在系统输入模块收到作业输入请求后，输入管理模块中的读过程负责将信息从输入装置读入缓冲区。当缓冲区满时，由写过程将信息从缓冲区写到外存输入井中。读过程和写过

程反复循环，直到一个作业输入完毕。当读过程读到一个硬件结束标志后，系统再次驱动写过程把最后一批信息写入外存并调用中断处理程序结束该次输入。然后，系统为该作业建立作业控制块JCB，从而使输入井中的作业进入作业等待队列，等待作业调度程序选中后进入内存。

5. 作业说明书和作业控制块有何异同？

答：作业说明书主要包含三方面内容：作业的基本描述、作业控制描述和资源要求描述。作业基本描述主要包括用户名、作业名、使用的编程语言名、允许的最大处理时间等。而作业控制描述则大致包括作业在执行过程中的控制方式，例如是脱机控制还是联机控制、各作业步的操作顺序以及作业不能正常执行时的处理等。资源要求描述包括要求内存大小、外设种类和台数、处理机优先级、所需处理时间、所需库函数或实用程序等。

而作业控制块是作业说明书在系统中生成的一张表格，该表格登记该作业所要求的资源情况、预计执行时间和执行优先级等。从而，操作系统通过该表了解到作业要求，并分配资源和控制作业中程序和数据的编译、链接、装入和执行等。

6. 操作系统为用户提供哪些接口？它们的区别是什么？

答：操作系统为用户提供两个接口，一个是系统为用户提供的各种命令接口，用户利用这些操作命令来组织和控制作业的执行或管理计算机系统。另一个接口是系统调用，编程人员使用系统调用来请求操作系统提供服务，例如申请和释放外设等类资源、控制程序的执行速度等。

7. 作业控制方式有哪几种？调查你周围的计算机的作业控制方式。

答：作业控制的主要方式有两种：脱机方式和联机方式。

脱机控制方式利用作业控制语言来编写表示用户控制意图的作业控制程序，也就是作业说明书。作业控制语言的语句就是作业控制命令。不同的批处理系统提供不同的作业控制语言。

联机控制方式不同于脱机控制方式，它不要求用户填写作业说明书，系统只为用户提供一组键盘或其他操作方式的命令。用户使用操作系统提供的操作命令和系统会话，交互地控制程序执行和管理计算机系统。

8. 什么是系统调用？系统调用与一般用户程序有什么区别？与库函数和实用程序又有什么区别？

答：系统调用是操作系统提供给编程人员的唯一接口。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速度等。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。它与一般用户程序、库函数和实用程序的区别是：系统调用程序是在核心态执行，调用它们需要一个类似于硬件中断处理的中断处理机制来提供系统服务。

9. 简述系统调用的实现过程。

答：用户在程序中使用系统调用，给出系统调用名和函数后，即产生一条相应的陷入指令，通过陷入处理机制调用服务，引起处理机中断，然后保护处理机现场，取系统调用功能号并寻找子程序入口，通过入口地址表来调用系统子程序，然后返回用户程序继续执行。

10. 为什么说分时系统没有作业的概念？

答：因为在分时系统中，每个用户得到的时间片有限，用户的程序和数据信息直接输入到内存工作区中和其他程序一起抢占系统资源投入执行，而不必进入外存输入井等待作业调度程序选择。因此，分时系统没有作业控制表，也没有作业调度程序。

11. 试述 UNIX 的主要特点。

答：UNIX 的主要特点是：

(1) UNIX 系统是一个可供多用户同时操作的交互式分时操作系统；

(2) 为了向用户提供交互式功能和使得用户可以利用 UNIX 系统的功能，UNIX 系统向用户提供了两种友好的界面或接口：系统调用和命令；

(3) UNIX 系统具有一个可装卸的分层树型结构文件系统，该文件系统使用方便、搜索简单；

(4) UNIX 系统把所有外部设备都当成文件，并分别赋予它们对应的文件名。从而，用户可以像使用文件那样使用任一设备而不必了解该设备的内部特性，这既简化了系统设计，又方便了用户；

(5) UNIX 系统核心程序的绝大部分源代码和系统上的支持软件都用 C 语言编写。且 UNIX 系统是一个开放式系统，即具有统一的用户接口，使得 UNIX 用户的应用程序可在不同的执行环境下运行。

正是由于 UNIX 具有上述这些特点，使得 UNIX 系统得到了广泛的应用和发展。

12. UNIX 操作系统为用户提供哪些接口？试举例说明。

答：UNIX 系统为用户提供两个接口，即面向操作命令的接口 Shell 和面向编程用户的接口：系统调用。常见的 Shell 命令如：login，logout，vi，emacs，cp，rm，ls，cc，link，adduser，chown，dbx，date 等；常见的系统调用如：ioctl，read，write，open，close，creat，execl，flock，stat，mount，fork，wait，exit，socket 等。

13. 在你周围装有 UNIX 系统的计算机上，练习使用后台命令、管道命令等 Shell 的基本命令。

答：例 1：用 Shell 语言编制一 Shell 程序，该程序在用户输入年、月之后，自动打印输出该年该月的日历：

```
echo "Please input the month:"
read month
echo "Please input the year:"
read year
```

```
cal $month $year
```

例 2：用 Shell 语言编制一 Shell 程序，当用户输入要搜索的字符串之后，该程序从指定文件中搜索出有关字符串，且该搜索过程后台执行：

```
echo "Please input the string to be searched:"
read str
echo "Please input the filename:"
read filename
grep $str $filename&.
```

14. Shell 变量和参数的作用是什么？操作系统默认定义的 Shell 变量有哪些？

答：正是因为有了 Shell 变量和参数，加上控制语句和条件语句，从而使 Shell 可以像一般程序语言那样进行编程，所不同的只是 Shell 编程的对象是系统命令，可以使得 Shell 命令文件中的命令按 Shell 编程所规定的顺序执行。

操作系统默认定义的 Shell 变量及定义如表 E1.1 所示。

表 E1.1

Shell 变量	定　义
$#	参数个数
$*，或$@	Shell 的全部参数
$-	指定给 Shell 的操作集
$?	Shell 程序中最后执行的命令的返回值
$$	Shell 进程号
$!	最后被执行的命令的进程号
$HOME	cd 命令的标准参数值(HOME 目录)
$PATH	查找 Shell 命令文件的路径
$MAIL	指定接收邮箱的打印接收信息
$PS1	显示提示符
$argv[0]，$argv[1]，…，$argv[n]	输入参数变量

第3章 进程管理

1. 有人说，一个进程是由伪处理机执行的一个程序，这话对吗？为什么？

答：对。

因为伪处理机的概念只有在执行时才存在，它表示多个进程在单处理机上并发执行的一个调度单位。因此，尽管进程是动态概念，是程序的执行过程，但是，在多个进程并行执行时，仍然只有一个进程占据处理机执行，而其他并发进程则处于就绪或等待状态。这些并发进程就相当于由伪处理机执行的程序。

2. 试比较进程和程序的区别。

答：(1) 进程是一个动态概念，而程序是一个静态概念，程序是指令的有序集合，无执行含义，进程则强调执行的过程。

(2) 进程具有并行特征(独立性，异步性)，程序则没有。

(3) 不同的进程可以包含同一个程序，同一程序在执行中也可以产生多个进程。

3. 我们说程序的并发执行将导致最终结果失去封闭性。这话对所有的程序都成立吗？试举例说明。

答：并非所有程序均成立。

如：

```
Begin
    local x
    x := 10
    print(x)
End
```

上述程序中 x 是内部变量，不可能被外部程序访问，因此这段程序的运行不会受外部环境影响。

4. 试比较作业和进程的区别。

答：一个进程是一个程序对某个数据集的执行过程，是分配资源的基本单位。作业是用户需要计算机完成某项任务，而要求计算机所做工作的集合。一个作业的完成要经过作业提交、作业收容、作业执行和作业完成 4 个阶段。而进程是已提交完毕的程序所执行过程的描述，是资源分配的基本单位。其主要区别关系如下：

(1) 作业是用户向计算机提交任务的任务实体。在用户向计算机提交作业之后，系统将它放入外存中的作业等待队列中等待执行。而进程则是完成用户任务的执行实体，是向系统申请分配资源的基本单位。任一进程，只要它被创建，总有相应的部分存在于内存中。

(2) 一个作业可由多个进程组成。且必须至少由一个进程组成，但反过来不成立。

(3) 作业的概念主要用在批处理系统中。像 Unix 这样的分时系统中，则没有作业概念。而进程的概念则用在几乎所有的多道程序系统中。

5. UNIX System V 中，系统程序所对应的正文段未被考虑成进程上下文的一部分，为什么？

答：因为系统程序的代码被用户程序所共享，因此如果每个进程在保存进程上下文时，都将系统程序代码放到其进程上下文中，则大大浪费了资源。因此系统程序的代码不放在进程上下文中，而是统一放在核心程序所处的内存中。

6. 什么是临界区？试举一临界区的例子。

答：临界区是指不允许多个并发进程交叉执行的一段程序。它是由于不同并发进程的程序段共享公用数据或公用数据变量而引起的。所以它又被称为访问公用数据的那段程序。例如：

```
getspace:
        Begin local g
            g = stack[top]
            top =top - 1
        End
release(ad):
          Begin
            top = top + 1
            stack[top] = ad
          End
```

7. 并发进程间的制约有哪两种？引起制约的原因是什么？

答：并发进程所受的制约有两种：直接制约和间接制约。

直接制约是由并发进程互相共享对方的私有资源所引起的。间接制约是由竞争共有资源而引起的。

8. 什么是进程间的互斥？什么是进程间同步？

答：进程间的互斥是指：一组并发进程中的一个或多个程序段，因共享某一公有资源而导致它们必须以一个不许交叉执行的单位执行，即不允许两个以上的共享该资源的并发进程同时进入临界区。

进程间的同步是指：异步环境下的一组并发进程因直接制约互相发送消息而进行互相合作、互相等待，是各进程按一定的速度执行的过程。

9. 试比较 P,V 原语法和加锁法实现进程间互斥的区别。

答：互斥的加锁实现是这样的：当某个进程进入临界区之后，它将锁上临界区，直到它退出临界区时为止。并发进程在申请进入临界区时，首先测试该临界区是否是上锁的，如果

该临界区已被锁住，则该进程要等到该临界区开锁之后才有可能获得临界区。

但是加锁法存在如下弊端：(1) 循环测试锁定位将损耗较多的 CPU 计算时间；(2) 产生不公平现象。

为此，P，V 原语法采用信号量管理相应临界区的公有资源，信号量的数值仅能由 P，V 原语操作改变，而 P，V 原语执行期间不允许中断发生。其过程是这样的：当某个进程正在临界区内执行时，其他进程如果执行了 P 原语，则该进程并不像 lock 时那样因进不了临界区而返回到 lock 的起点，等以后重新执行测试，而是在等待队列中等待由其他进程做 V 原语操作释放资源后，进入临界区，这时 P 原语才算真正结束。若有多个进程做 P 原语操作而进入等待状态之后，一旦有 V 原语释放资源，则等待进程中的一个进入临界区，其余的继续等待。

总之，加锁法是采用反复测试 lock 而实现互斥的，存在 CPU 浪费和不公平现像，P，V 原语使用了信号量，克服了加锁法的弊端。

10. 设在书 3.6 节中所描述的生产者-消费者问题中，其缓冲部分为 m 个长度相等的有界缓冲区组成，且每次传输数据长度等于有界缓冲区长度以及生产者和消费者可对缓冲区同时操作。重新描述发送过程 deposit(data)和接收过程 remove(data)。

答：设第 I 块缓冲区的公用信号量为 mutex[I]，保证生产者进程和消费者进程对同一块缓冲区操作的互斥，初值为 1。设信号量 avail 为生产者进程的私用信号量，初值为 m。信号量 full 为消费者进程的私用信号量，初值为 0。从而有：

```
deposit(data)
    Begin
        P(avail)
        选择一个空缓冲区 i
        P(mutext[I])
        送数据入缓冲区 i
        V(full)
        V(mutext[I])
    End

Remove(data)
    Begin
        P(full)
        选择一个满缓冲区 I
        P(mutext[ I ])
        取缓冲区 i 中的数据
        V(avail)
        V(mutext[ I ])
    End
```

11. 两进程 P_A，P_B 通过两 FIFO 缓冲区队列连接(如图 E1.1)，每个缓冲区长度等于传

送消息长度。

进程 P_A,P_B 之间的通信满足如下条件:

(a) 至少有一个空缓冲区存在时,相应的发送进程才能发送一个消息。

(b) 当缓冲队列中至少存在一个非空缓冲区时,相应的接收进程才能接收一个消息。

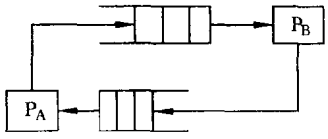

图 E1.1

试描述发送过程 send(i,m)和接收过程 receive(i,m)。这里 i 代表缓冲队列。

答:定义数组 buf[0],buf[1],bufempty[0],buffull[1]是 P_A 的私有信息量,buffull[0],bufempty[1]是 P_B 的私有信息量。

初始时:

```
bufempty[0] = bufempty[1] = n,(n 为缓冲区队列的缓冲区个数)
buffull[0] = buffull[1] = 0

send(I,m)
Begin
    local x
    P(bufempty[I])
    按 FIFO 方式选择一个空缓冲区
    buf[I](x)
    buf[I](x) = m
    buf[I](x)置满标记
    V(buffull[I])
End

Receive(I,m)
Begin
    Local x
    P(buffull[I])
    按 FIFO 方式选择一个装满数据的缓冲区 buf[I](x)
    m = buf[1](x)
    buf[1](x)置空标记
    V(bufempty[I])
End
```

P_A 调用 send(0,m)和 receive(1,m)

P_B 调用 send(1,m)和 receive(0,m)

12. 在和控制台通信的例中,设操作员不仅仅回答用户进程所提出的问题,而且还能独立地向各用户进程发出指示。对于这些指示,操作员不要求用户进程回答,但它们享有比其他消息优先传送的优先度。即如果 inbuf 中如有指示存在,系统不能进行下一次通信会话。试按上述要求重新描述 CCP 和 KCP,DCP。

答：KCP 描述如下：

设 T_Ready 和 T_Busy 分别为键盘 KP 和键盘控制进程 KCP 的私有信号量，其初值为 0 和 1。设 inbuf 为 inbuf 的共有信号量，初值为 1，表示其中没有控制消息。

初始化{清除所有 inbuf 和 echobuf}

```
Begin
    local x
    P(T_Ready)
    从键盘数据传输缓冲 x 中取出字符 m 记为 x.m
    if 为控制消息
    P(inbuf)
    Send(x.m)
    将 x.m 送入 echobuf
    V(T_Busy)
End
```

键盘控制进程 DCP：

```
repeat
    P(T_Busy)
    把键入字符放入数据传输缓冲
    V(T_Ready)
Until 终端关闭
```

显示其控制进程 DCP：

```
设 D_Ready 和 D_Busy 分别为 DP 和 DCP 的私有信号量且初值为 0 和 1.
初始化{清除输出缓冲 outbuf,echo 模式置 false}
Begin
    if outbuf 满
    then
        receive(k)
        P(D_Busy)
        把 k 送入显示器数据缓冲区
        V(D_Ready)
    else
    Else
        Echo 模式置 true
        Echobuf 中字符置入显示器数据缓冲区 fi
End
```

显示器动作 DP：

```
repeat
    if echo 模式
    then
```

```
        打印显示器数据缓冲区中字符
    else
        P(D_Ready)
        打印显示器数据缓冲区中消息
        V(D_Busy)
Until 显示器关机
```

设过程Read(x)把inbuf中的所有字符读到用户进程数据区x处，过程write(y)把用户进程y处的消息写到outbuf中. read(x)和write(y)可分别描述如下：

```
read(x):            write(y)
    略                  略
```

13. 编写一个程序使用系统调用fork生成3个子进程，并使用系统调用pipe创建一管道，使得这3个子进程和父进程公用同一管道进行信息通信。

答：

```
main( )
{
    int i,r,p1,p2,fd[2];                          /* fd[2]为管道文件读写标识 */
    char buf[50],s[5];
    pipe(fd);                                     /* 创建管道 pipe( ) */
    while((p1 = fork(1) == -1);                   /* 创建子进程 1 */
    if(p1 == 0)                                   /* 在子进程 1 中执行 */
    {
        lockf(fd[1],1,0);                         /* 锁定写过程 */
        sprintf(buf,"child process P1 is seding message! \n");
        printf("child process P1! \n");
        write(fd[1],buf,50);                      /* 将 buf 中数据写入 pipe */
        sleep(5);                                 /* 睡眠等待父进程读出 */
        lockf(fd[1],0,0);                         /* 解锁 */
        exit(0);
    }
    else
    {
        while((p2 = fork( )) == -1);              /* 创建进程 2 */
        if(p2 == 0)                               /* 在子进程 2 中执行 */
        {
            lockf(fd[1],1,0);                     /* 锁定写过程 */
            sprintf(buf,"child process P2 is sending message! \n");  /* 数据入 buf */
            printf("child process P2! \n");
            write(fd[1],buf,50);                  /* buf 数据写入 pipe */
            sleep(5);                             /* 同步等待父进程读 */
            lockf(fd[1],0,0);                     /* 解锁 */
```