高等学校工科电子类教材

计算机软件实践教程

系统软件部分

郭浩志　主编

（第二版）

西安电子科技大学出版社

高等学校工科电子类教材

计算机软件实践教程

系统软件部分

（第二版）

郭 浩 志 主编

西安电子科技大学出版社
1994

(陕)新登字 010 号

内容简介

本书系《计算机软件实践教程》和《应用软件实践教程》的修订版本，旨在介绍我国高等院校计算机专业各有关课程上机实习的内容、方法和要求。各章分别给出适当示例和实习题，但不局限于具体的计算机和程序语言。全书比较全面地介绍了当前我国高等院校主要软件课程的实践性教学环节。全书共分三册。上册为基础软件部分，中册为系统软件部分，下册为应用软件部分。本册系中册，分别给出了操作系统基础、操作系统大作业、数据库、软件工程、编译原理(技术)基础、编译系统大作业和人机界面等课程的实践性教学环节。本书的组织既注意到内容的系统性和形式的一致性，又力使各章内容具有相对的独立性，使每一章可以单独用来指导该门课程的实习。

本书系高等院校计算机专业的教学用书，供大学生在整个大学期间使用，也可作为教师、研究生或工程技术人员的参考书。

高等学校工科电子类教材

计算机软件实践教程

系统软件部分

(第二版)

郭浩志　主编

责任编辑　徐德源

西安电子科技大学出版社出版

西北工业大学印刷厂印刷

陕西省新华书店发行　　各地新华书店经售

开本 787×1 092 1/16　印张 23　12/16　字数 564 千字

1985 年 9 月第 1 版　1994 年 12 月第 2 版　1994 年 12 月第 1 次印刷　印数 1—3 000

ISBN 7 - 5606 - 0316 - 5/TP · 0116(课)　　定价：13.35 元

出 版 说 明

根据国务院关于高等学校教材工作的规定，我部承担了全国高等学校和中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978～1990年已编审、出版了三个轮次教材，及时供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻国家教委《高等教育“八五”期间教材建设规划纲要》的精神，“以全面提高教材质量水平为中心，保证重点教材，保持教材相对稳定，适当扩大教材品种，逐步完善教材配套”，作为“八五”期间工科电子类专业教材建设工作的指导思想，组织我部所属的九个高等学校教材编审委员会和四个中等专业学校专业教学指导委员会，在总结前三轮教材工作的基础上，根据教育形势的发展和教学改革的需要，制订了1991～1995年的“八五”（第四轮）教材编审出版规划。列入规划的，以主要专业主干课程教材及其辅助教材为主的教材约300多种。这批教材的评选推荐和编审工作，由各编委会或教学指导委员会组织进行。

这批教材的书稿，其一是从通过教学实践、师生反映较好的讲义中经院校推荐，由编审委员会（小组）评选择优产生出来的，其二是在认真遴选主编人的条件下进行约编的，其三是经过质量调查在前几轮组织编定出版的教材中修编的。广大编审者、各编审委员会（小组）、教学指导委员会和有关出版社，为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还可能有缺点和不足之外，希望使用教材的单位，广大教师和同学积极提出批评和建议，共同为不断提高工科电子类专业教材的质量而努力。

机械电子工业部电子类专业教材办公室

再版前言

本书是高等院校工科电子类计算机专业统编教材，由中国电子工业总公司计算机教材编委会审定，并推荐出版。责任编委为浙江大学俞瑞钊教授。

全书按计算机编委会审定的编写大纲进行编写和审阅，由西安电子科技大学蔡希尧教授主审，国防科技大学郭浩志教授主编。

《计算机软件实践教程》和《应用软件实践教程》两种全国统编教材自 1985、1986 年出版以来，两次重印，得到国内高校的欢迎和专家教授的充分肯定，各界一致认为："该书是国内公开发行的第一本软件实践教材，为国内大多数理工科院校计算机课程的教学和软件人材的培养发挥了重要的作用，属国内首创"。修订版仍以计算机教学计划中对大学本科生培养目标的基本要求为出发点，着眼于加强对学生的基本技能和科学作风的培养和训练。全书着重介绍诸软件课程上机实习的目的、要求、内容和方法。诸实习内容至少都有一个示例，并提供若干实习题供各校选择，有些题目还给出必要的提示和思考题。大部分章末均列出该章的主要参考文献。

为适应计算机科学技术和高等教育事业的蓬勃发展，此次再版，在保留原书特色的基础上，内容做了很大更新，主要变动如下：

- 增加离散数学、软件工程、人机界面、人工智能、专家系统和算法设计与分析等；
- 将 PDP 汇编改为 PC 汇编和 VAX 汇编；
- 将高级语言程序设计改为 PASCAL 和 C 两章；
- 删去系统程序设计；
- 对操作系统和编译分别增加大作业。

即使题目不变的各章，其内容大多数也做了很大的更新。

修订版是在总结我国若干所理工科大学软件课程近几年教学实习的实践经验和比较广泛搜集其他院校有关资料和意见的基础上编写的。编写时尽量考虑全国大多数院校的通用性。各校在采用本书时，宜根据各自的教学大纲和实际条件，结合具体的计算机和程序语言，作适当的选择和补充。

本书拟分三册出版。上册为基础软件部分，中册为系统软件部分，下册为应用软件部分。本册为中册，共分七章，依次为操作系统基础、操作系统大作业、数据库、软件工程、编译原理(技术)基础、编译系统大作业和人机界面等。

参加中册编写的有徐良贤、胡剑、章干(第十二章、胡、章参加调试)，徐良贤、王涛(第十三章)，郑怀远、卢朝霞(第十四章)，严瑜(第十五章)，郭浩志(第十六章)，刘春林(第十七章)，徐良贤、裴健、王几、许匡(第十八章)。

在本书编写过程中，得到了西安电子科技大学、浙江大学、上海交通大学、东北大学、西北工业大学、东南大学、清华大学、国防科技大学许多领导、教师和学生的关心和支持，在此表示谢意。

本书虽是再版，仍感不很适应科技与教育的飞速发展，加上修订时间仓促，编者水平不齐，错误在所难免，热诚欢迎读者批评指正。

编者

1994

目　录

第十二章 操作系统基础实习

第一节 概 述

操作系统是现代计算机最重要的大型系统软件之一，其作用是对计算机系统进行统一的调度和管理，提供各类功能齐全的系统服务，为用户创造灵活方便的使用环境。例如，操作系统具有各种文件、文本和图形的编辑功能，控制程序的编译、连接和运行，允许在一个比实际内存大得多的虚地址空间中编制和运行程序，方便地实施各种输入输出操作而毋须繁琐的程序设计等等。一个精心设计的操作系统能极大地扩充计算机系统的功能，充分地发挥系统中各种设备的使用效率，提高系统工作的可靠性。

“操作系统原理”是计算机科学与工程学科各专业的一门主要专业课程，它涉及计算机系统中各种软、硬件资源的管理，内容比较繁杂，具有很强的实践性。只有把理论和实践紧密地结合起来，才能真正领会本课程的精髓，取得较好的学习效果。

培养计算机专业学生的系统程序设计能力，也是本课程的重要目标。系统程序要求结构清晰、合理、可读性好，有正确而简明的注释。通过实习可以培养学生正规程序设计的风格和能力。

本章内容包括下列 5 个方面：

(1) 进程调度；

(2) 存贮器管理；

(3) 假脱机输入、输出；

(4) 文件管理系统；

(5) 死锁现象的观察和避免。

上述每个实习约需 10～14 个学时，如果课程学时数不够，可删去 1～2 个实习或分组选做。操作系统是大学高年级开设的课程，全部实习均要求学生独立完成，并写出正规的实习报告，不能照搬示例。

第二节 进 程 调 度

一、目的与要求

(一) 目的

进程是操作系统最重要的概念之一，是了解操作系统实质的关键。进程调度又是操作系统内核的主要内容。本实习要求学生独立地用高级语言编写和调试一个简单的进程调度程序。调度算法可自由选择或自行设计，例如，简单轮转法、优先数法、反馈排队法等。本实习可加深对于进程、进程调度和各种调度算法的理解。

（二）要求

（1）设计一个含有 n 个进程共行的进程调度程序。每个进程可由一个进程控制块(PCB)表示。进程控制块通常应包含下述信息：进程名、进程优先数、进程需要运行的时间、每次调入占用 CPU 的时间以及进程的状态等，且可按调度算法的不同而增删。

（2）调度程序应包含 2～3 种不同的调度算法，运行时可任选一种，以利于各种算法的分析比较。

（3）系统应能显示或打印各进程状态和参数的变化情况，便于观察诸进程的实际调度过程。

二、示例

[例 12.2.1]

1. 题目

本示例选用优先数法和简单轮转法对 5 个进程进行调度。每个进程可处于运行 R(un)、就绪 W(ait)和完成 F(inish)3 种状态之一。假定各进程的起始状态都是就绪状态 W。

设进程控制块 PCB 如图 12.2.1 所示。

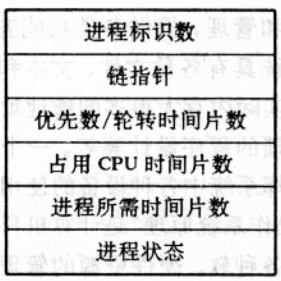

图 12.2.1　进程控制块 PCB

运行进程和就绪进程队列的 PCB 键结构如图 12.2.2 所示。

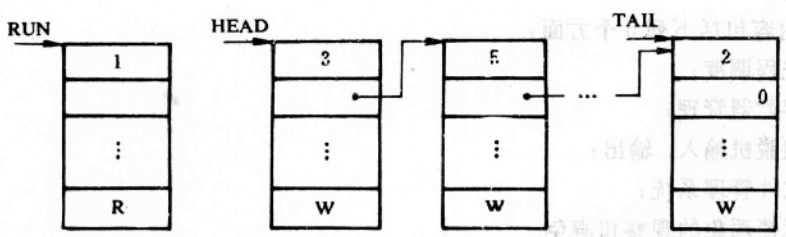

图 12.2.2　PCB 链结构

其中，RUN——当前运行进程指针；

HEAD——就绪进程队列的 PCB 链链首指针；

TAIL——就绪进程队列的 PCB 链链尾指针。

为了便于处理，程序中进程的运行时间以逻辑时间片(以下简称时间片)为单位计算。各进程的优先数或轮转时间片数以及进程完成需要的时间片数均由伪随机数发生函数产生。

2. 算法与框图

（1）优先数法(Priority)：就绪进程队列的 PCB 链是按优先数大小，从高到低排列，链首的进程首先调入运行。每过一个时间片，进程完成所需运行的时间片数减 1，说明它已运行了一个时间片。常规算法，只要依次完成队列中的各个进程就可。为了更能说明问题，假定每运行一个时间片，如果进程仍未完成，则它的优先级要降低(如优先数减 3)，或者说进程如果在一个时间片中完成不了，它的优先级就降低一级，以此类推，这样，每过一

个时间片，就要重新计算现行进程的优先数(即减 3)，随后比较现行进程和就绪链链首进程的优先数，如果仍是现行进程高或者相同，就让现行进程继续运行，否则，调度就绪链链首进程投入运行，使之成为现行进程。而原运行进程再按其优先数的大小插入就绪队列，且应改变它们各自的状态，直至所有进程都运行完各自的时间片数(即各自所需运行的时间片数减至 0)为止。

(2) 简单轮转法(Round Robin)：就绪进程队列的 PCB 链是按各进程进入的先后次序排列的，各进程每次占用处理机的轮转时间可按其重要程度登入 PCB 中的轮转时间片数记录项(对应于优先数法的优先数记录项位置)。每过一个时间片，运行进程所需运行的时间片数减 1，而它占用处理机的时间片数加 1，然后比较其占用处理机的时间片数是否与该进程的轮转时间片数相等，如相等说明已到达轮转时间，应将现运行进程排到就绪队列末尾，再调度链首进程占用处理机，且改变它们的进程状态 直至所有进程完成各自的时间片(即各进程所需运行的时间片数减至 0)为止。

(3) 程序框图如图 12.2.3 所示。

3. 程序、操作和结果

(1) C 语言程序：

```
#include <stdio.h>
#include <stdlib.h>
#define MAXPROCNUM 5
/* the algorithm */
#define TIME   0   /* "round robin" */
#define PRIO   1   /* "priority" */
/* the state of process */
#define WAIT     0
#define RUN      1
#define FINISH   2

typedef struct
    {int id,
     pri,
     cputime,
     alltime,
     next,
     state;
    }
    PROC; /* PCB structure */

int run, head, tail;
PROC procchain[MAXPROCNUM];
FILE *f;

void insert(int q)
/* insert a process into queue in terms of its priority */
{int p,s;
```

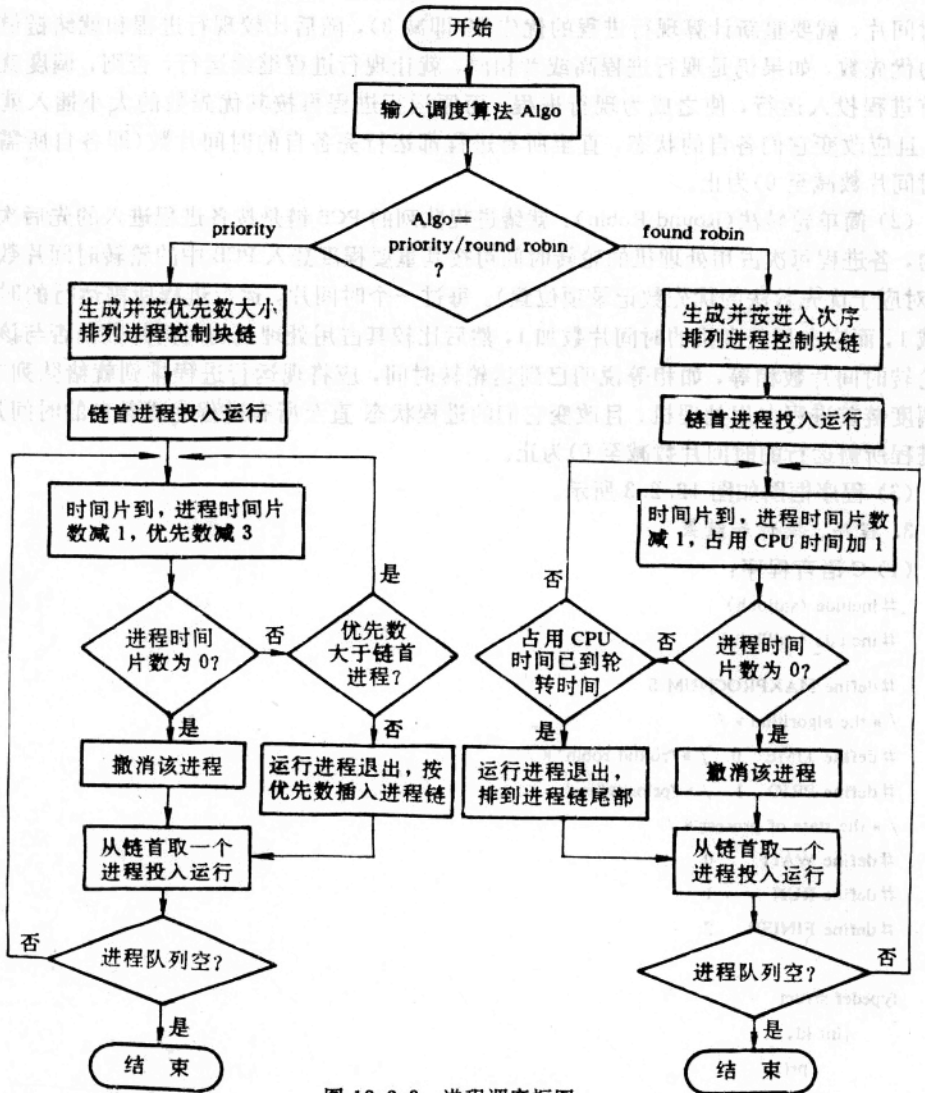

图 12.2.3　进程调度框图

```
p=head;
s=procchain[head].next;
while (procchain[q].pri<procchain[s].pri &&s)
      {p=s;
       s=procchain[s].next;
      }
procchain[p].next=q;
procchain[q].next=s;
}

void insert2(  )
/* put a process into the tail of the queue */
```

```
{procchain[tail].next=run;
tail=run;
procchain[run].next=0;
}

void print(  )
/* print the running process, waiting queue and PCB sequence list */
{
    int i, p;
    for (i=0; i<40; i++)
      fprintf(f, " = ");
    fprintf(f, " \n ");
    fprintf(f, "RUNNING PROC                    WAITING QUENE\n");
    fprintf(f, "   %d                ", procchain[run], id);
    p=head;
    while (p)
      {fprintf(f, " %5d ", p);
      p=procchain[p].next;
      }
    fprintf(f, " \n ");
    for (i=0; i<40; i++)
      fprintf(f, " = ");
    fprintf(f, "\nID                ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf(f, "%5d", procchain[i].id);
    fprintf(f, " \nPRIORITY          ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf(f, " %5d ", procchain[i].pri);
    fprintf(f, " \nCPUTIME           ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf(f, " %5d ", procchain[i].cputime);
    fprintf(f, " \nALLTIME           ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf (f, " %5d ", procchain[i].alltime);
    fprintf(f, " \nSTATE             ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf(f, " %5d ", procchain[i].state);
    fprintf(f, " \nNEXT              ");
    for (i=1; i<=MAXPROCNUM; i++)
      fprintf(f, " %5d ", procchain[i].next);
    fprintf(f, " \n ");
    for (i=0; i<40;i++)
      fprintf(f, " = ");
```

```
    fprintf(f, " \n ");
}
void init (int alg)
/* create a waiting queue */
{int i;
randomize( );
head=0;
if (alg==PRIO)
    for (i=1; i<=MAXPROCUNM; i++)
        {procchain[i].id=i;
        procchain[i].pri=(random(100)+11)%41;
        procchain[i].cputime=0;
        procchain[i].alltime=(random(100)+1)%7;
        procchain[i].state=WAIT;
        procchain[i].next=0;
        if ((procchain[i].pri<procchain[head].pri) && (head!=0))
          insert(procchain[i].id);
        else
          {procchain[i].next=head;
          head=procchain[i].id;
          }
        }
else
  {/*alg==TIME */
  for (i=1; i<=MAXPROCNUM; i++)
  {procchain[i].id=i;
  procchain[i].pri=(random(100)+1)%3+1;
  procchain[i].cputime=0;
  procchain[i].alltime=(random(100)+1)%7;
  procchain[i].state=WAIT;
  procchain[i].next=(i+1)%(MAXPROCNUM+1);
  }
    head=1;
    tail=MAXPROCNUM;
    procchain[MAXPROCNUM].next=0;
  }
run=head;
procchain[run].state=RUN;
head=procchain[head].next;
print( );
}
void prisch( )
```

```
/* schedule the processes in terms of their prioriy */
{while (run)
 {procchain[run].cputime++;
 procchain[run].pri-=3;
 if(procchain[run].alltime>0) procchain[run].alltime--;
 if (procchain[run].alltime<=0)
   {procchain[run].state=FINISH;
   procchain[run].next=0;
   if (head)
     {run=head;
     procchain[run].state=RUN;
     head=procchain[head].next;
     }
   else
     {procchain[0].id=procchain[run].id;
     run=0;
     }
   }
 else
   if (head && procchain[run].pri<procchain[head].pri)
     {procchain[run].state=WAIT;
     insert(run);
     run=head;
     procchain[run].state=RUN;
     head=procchain[head].next;
     }
 print( );
 }/* end of while */
 }
void timesch( )
/* schedule the process with round robin algorithm  */
{while (run)
 {if (procchain[run].alltime>0) procchain[run].alltime--;
 procchain[run].cputime++;
 if (procchain[run].alltime<=0)
   {procchain[run].state=FINISH;
   procchain[run].next=0;
   if (head)
     {run=head;
     procchain[run].state=RUN;
     head=procchain[head].next;
     }
   else
```

```
        {procchain[0].id=procchain[run].id;
        run=0;
        }
    }
    else
        if (head && procchain[run].cputime==procchain[run].pri)
            {procchain[run].state=WAIT;
            procchain[run].cputime=0;
            insert2( );
            run=head;
            procchain[run].state=RUN;
            head=procchain[head].next;
            }
    print( );
  }/* while */
}
main( )
{int alg, i;
do{printf(" TYPE THE ALGORITHM (0 FOR ROUND ROBIN AND 1 FOR PRIORITY )\n ");
    printf(" input 0 or 1 : ");
    scanf(" %d ", &alg);
    }
while (alg! =PRIO && alg! =TIME);
if (alg==TIME)
    {f=fopen(" TIME. TEXT", " w ");
    if (f==NULL) exit(1);
    fprintf(f, " OUTPUT OF ROUND ROBIN\n ");
    init(TIME);
    timesch( );
    }
else
    {f=fopen(" PRIORITY. TEXT", "w");
    if (f==NULL) exit(1);
    fprintf(f, " OUTPUT OF PRIORITY\n ");
    init (PRIO);
    prisch( ) :
    }
for (i=0; i<40; i++)
    fprintf(f, " = ");
fprintf(f, " \n ");
fprintf(f, " SYSTEM FINISHED\n ");
folose(f);
}
```

(2) PASCAL 程序：

```
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  *     THIS PROGRAM MADE BY * * * * * * * 93--10--15              *
  *     TO SCHEDULE 5 CONCURRENT PROCESSES USING THE               *
  *     ALGORITHM  " ROUND ROBIN " AND  " PRIORITY. "              *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
PROGRAM SCHEDULE;
  LABEL 1;
  CONST FURTHEST=5;      (* NUMBER OF CURRENT PROCESSES *)
  TYPE PROCNUM=0..FURTHEST;
      ALGORITHM=(RR, PRIO); (* 'ROUND ROBIN' AND 'PRIORITY' *)
      PROCESS=
       RECORD          (* PCB STRUCTURE *)
        ID : PROCNUM;
        PRIORITY : INTEGER;
        CPUTIME : INTEGER;
        ALLTIME : INTEGER;
        STATE : STRING;
        NEXT : PROCNUM;
       END;
VAR PROCHAIN : ARRAY[FROCNUM] OF PROCESS;
      RUN, HEAD, TAIL : PROCNUM;
      OUTFILE : TEXT;
      OUTNAME : STRING;
      ALGO : STRING;
      J : INTEGER;
PROCEDURE PRINT;
      (* PRINT THE RUNNING PROCESS, WAITING QUEUE
        AND PCB SEQUENCE LIST *)
  VAR K : INTEGER; P : PROCNUM;
  BEGIN
    FOR K:=1  TO  40  DO
      WRITE(OUTFILE, '=');
    WRITELN(OUTFILE);
    WRITE(OUTFILE, 'RUNNING PROC.', '          ');
    WRITELN(OUTFILE,  ' WAITING QUEUE ');
    WRITE(OUTFILE, '          ', PROCHAIN[RUN].ID,
                   '          ');
P:=HEAD;
WHILE(P<>0) DO
  BEGIN
    WRITE(OUTFILE, P:5);
    P:=PROCHAIN[P].NEXT;
```