空 军 工 程 大 学 学 报(自然科学版)
第1卷第2期 JOURNAL OF AIR FORCE ENGINEERING UNIVERSITY Vol.1 No.2
2000 年 6 月 (NATURAL SCIENCE EDITION) Jun. 2000

# 一种基于函数的多任务调度算法

禄乐滨， 刘明东
(空军工程大学 电讯工程学院，陕西 西安 710077)

**摘 要：** 提出了一种新颖的伪任务概念，通过构造任务控制表及和任务自陷及等待的方法，设计与实现了一种非剥夺按时间片循环调度算法。

**关键词：** 伪任务；调度；自陷；等待
**中图分类号：** TP311 **文献标识码：**A **文章编号：**1009-3516(2000)02-0056-04

为了研究工作的方便，这里提出伪任务的概念，即完成一定独立任务工作的函数。多任务概念中的任务在一般意义上讲是可执行程序，多任务并行执行是由操作系统来调度完成的。提出伪任务并行执行，是希望在一个伪任务执行时，另外一个伪任务也可以被启动执行。这里的伪任务从概念上讲是程序中的函数，希望两个或多个函数同时执行必须解决相关的一些问题。

## 1 具体实现

### 1.1 任务的状态和任务控制表(PCB)

(1)任务状态

伪任务有着“运行—暂停—运动”的活动规律，并非一直处于活动状态，它有四个基本状态，状态之间转换如图1所示。

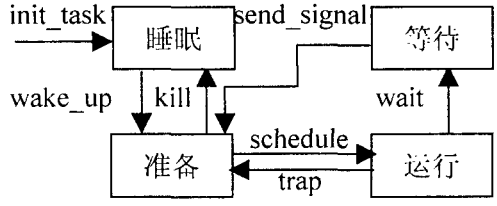


图1 状态转换图

①睡眠(TASK _ SLEEP)状态：当一个伪任务还没有被提供参数时，它处于睡眠状态，一旦该伪任务被提供所要参数时，它将处于准备状态。

②准备(TASK _ READY)状态：处于准备状态的伪任务等待调度程序处理另外的任务后，将CPU提供给该准备状态的任务，使其运行。

③运行(TASK _ RUN)状态：占有CPU。

④等待状态：当一个运行的伪任务需要另外一个任务完成一定工作后，再处理该任务，它就处于等待状态。

(2)任务控制表(PCB)

为描述伪任务的运动变化过程，采用一个与伪任务相联系的任务控制表(PCB)。主控程序根据PCB而感知伪任务工作状态。

PCB的数据结构如下，它是全局变量

```
struct PCBTYPE
{
    int pno;                    /* 任务号 */
    int task _ class;           /* 任务类型 */
    int status;                 /* 状态 */
    int signal;                 /* 等待信号 */
```

**收稿日期：**1999-12-17
**作者简介：**禄乐滨(1963-)，男，讲师.

```
    struct SIGNAL _ BOX _ NODE * signal _ box;        /* 信号量 */
    int breakpoint;                                   /* 断点号 */
    int sub _ task;                                   /* 子任务号 */
    int reuse _ no;                                   /* 任务重入数 */
    void (* func)(void * env);                        /* 任务名 */
    void * env;                                       /* 任务环境指针 */
  }pcbtab[MAXPCB];
```

pcbtab[MAXPCB]:MAXPCB 代表任务函数个数(c 语言数组下标从 0 开始)

另外一个有关的全局变量定义如下:

```
int pcb _ point      /* 当前正在运行的伪任务下标 */
```

pcb _ point 初始化为－1,表示没有伪任务被启动。

在程序开始应该初始化 PCB 表,它是一个静态表,不象操作系统中进程表是动态建立的。初始化函数为 init _ pcb( )

### 1.2 任务控制

"任务控制"的职责是对全局的任务实施有效的管理。

任务控制函数有:

(1)创建任务(init _ task)

void init _ task(int pno,int task _ class,void( * func)(void * env),void * env,int reuse _ no)

在整个任务运行前,我们知道究竟可能有多少任务能参加运行,所以应该对所有的任务进行初始化。它的操作过程是,先向 PCB 索取一个无用的 PCB 结构,记录它的任务号,并置它的状态为"睡眠状态",然后记录它的相关信息。

(2)唤醒任务(wake _ up)

int wake _ up(int pno,void * env)

当处于"睡眠状态"任务所需要的参数被提供,此时就应及时使 PCB 表中的该任务的状态变成"准备状态",并将它所需要的参数挂在它相应的位置,等待调度程序运行它。

(3)撤销任务(kill)

当一个任务完成任务后应予以撤销,以使得它相应的 PCB 的状态为"睡眠状态",schedule 调度程序不再对其工作。

(4)等待任务(_ wait)

当一个任务正在运行而它期待的某一任务尚未出现时这个运行的任务就应该使自己通过等待把自己 PCB 值变为"等待状态",交出 CPU。

(5)测试(test)

init test(int name)

name:是要测试的资源名

函数返回值 0 表示资源可用;1 表示资源不能用

### 1.3 调度方法

调度方法采用的是任务函数自陷或者等待后,循环查询 PCB 来完成宏观上并行的执行。

```
void schedule( )
{struct PCBTYPE * ptr;
  ptr=pcbtab+pcb _ point;
  while(system==RUN)
  {++ptr;
  if((++pcb _ point)>=MAXPCB)
    {pcb _ point=0;ptr=pcbtab}
  if(ptr->status==TASK _ READY && ptr->pno>0)
    {current _ task=ptr->pno;
    signal _ box=ptr->pno;
    ptr->signal _ box=NULL;
    (* ptr->func)(ptr->env);}
   }
```

}

### 1.4 伪任务函数

伪任务函数分为两类:不可重入任务函数;可重入任务函数。不可重入任务函数:表示该程序只可能被调用一次。可重入任务函数:表示该程序可能被调用多次。

(1)不可重入任务函数的编写规范

```
fun _ name(……)
{常量及变量说明
#define MAX _ TRAP n
#include "TRAP. h"
函数体
}
```

函数体

```
TRAP(1)
……
TRAP(2)
……
WAIT(k)
……
WAIT(n)
```

其中定义为 MAX _ TRAP 的 n 为函数体中自陷(TRAP)和等待(WAIT)之和的个数,n 要在 TRAP. H 中使用,如果省略了这个宏定义,编译时将会发生错误。PCB 中只记录了任务函数的断点号,如果不记录断点处的信息,下一次任务函数进入断点时现场信息将会丢失。为了保证伪任务函数重新进入时能够保留现场信息,必须在变量说明中使用静态局部变量。

(2)可重入任务函数的编写规范

```
设可重入次数为 mm
fun _ name(……)
{#define REUSE _ NO mm
  #define p REF(p)
    常量及变量说明
  #define MAX _ TRAP n
  #include "TRAP. h"
  函数体
  }
```

函数体

```
TRAP(1)
……
TRAP(2)
……
WAIT(k)
……
WAIT(n)
```

可重入函数和不可重入函数编程基本一样,只是在开始定义了二个宏定义。通过 TRAP. h 中的宏定义,将其转为一维数组,数组长为可重入次数 mm,PCB 表中记录着 REUSE _ NO 个同一个函数的任务,相同的函数通过 REUSE _ NO 来区分。

### 1.5 TRAP. h

任务函数将 TRAP. h 库在变量说明后即装入库,它有 WAIT、TRAP 的定义,并可根据 pcb 表所对应任务断点号来决定,任务函数从那里进行工作。

```
#ifndef MAX _ TRAP
#define MAX _ TRAP 1
#endif
#undef REUSE _ NO
#define REUSE _ NO pcbtab[pcb _ point]. reuse _ no-1
#undef WAIT(num,SIGNAL)
#define WAIT(num,SIGNA) \
  {_ wait(SIGNAL);\
    pcbtab[pcb _ point]. break _ point=num;\
    return;}\
    B _ P _ # #num;
#define TRAP(num)\
while (signal _ box! =NULL)\
     {signal _ box->box _ ptr=NULL;\
```

```
    signal_box=signal_box->next;\
        }\
  pcbtab[pcb_point].breakpoint=num;\
  pcbtab[pcb_point].status=TASK_READY;\
  return;\
  B_P_##num;

switch (pcbtab[pcb_point].breakpoint)
  {
  case 0:goto B_P_0;
  #define B_POINT_NUM 1
  #if MAX_TRAP>=B_POINT_NUM
    case 1:goto B_P_1;
  #endif
  #undef B_POINT_NUM
  #define B_POINT_NUM 2
  #if MAX_TRAP>=B_POINT_NUM
    case 2:goto B_P_2;
  #endif
  ..............
  #undef B_POINT_NUM
  #define B_POINT_NUM 10
  #if MAX_TRAP>=B_POINT_NUM
    case 10:goto B_P_10;
  }
  #undef B_POINT_NUM
  B_P_0:
```

## 2　结束语

本文提出了一种新颖的伪任务概念，通过构造任务控制表和伪任务自陷和等待的方法，设计与实现了一种非剥夺按时间片循环调度算法，该算法已调试通过，并且可行。

### 参 考 文 献


[1] Programming in VAX C.

[2] 谭浩强. C 程序设计[M]. 北京：清华大学出版社，1991.

[3] 汤子瀛，杨成钟. 计算机操作系统[M]. 西安：西北电讯工程学院出版社，1984.


## A New Algorithm for Multitask Schedule Based on Function


LU Le-bin,　LIU Ming-dong

(The Telecommunication Engineering Institute, AFEU., Xi'an 710077, China)



**Abstract**:　The Author provides a new pseudo-task concept to fit the concrete fact, designs and implements a non-preemptive cycle schedule algorithm based on time slice, by the way of building a task-control table and pseudo-task"trap"and"wait".



**Key words**:　pseudo-task; task schedule; trap; wait