HOS——130
小型操作系统的调度算法

姚卿达　欧贵文　李宏新

现代电子计算机操作系统的出现，使计算机使用效率成倍甚至几十倍的提高，而且为用户提供了方便的使用方式以及令人满意的服务质量，对计算机的深入推广有极大意义。

操作系统是用来管理计算机资源（如处理机时间、存贮空间、外围设备、数据库）以及自动调度用户的作业程序，使多个用户能有效地共享一套计算机装置的软件系统。而调度算法的设计则是设计一个操作系统的核心。计算机采取机房开放的调度规则，由调度员根据用户作业情况及急缓程度，分配给每个用户一段时间去操作。用户可以当场修改程序，还可以在执行中对程序做种种试验。但这样的调度往往90%的时间花在人工干预——如按键操作、思考、改错、失误处理等等——上面，机器利用率（＝执行时间/总时间）不超过10%，大多数外围设备闲置。

那怕是最简单的自动调度，也能大大减少由于人工干预而导致的浪费。批次处理(Batch-processing)调度算法及 spooling (simultaneous peripheral operations on—line) 调度算法可以使处理机利用率高达90%。

本文所描述的HOS/130算法是在一个小型操作系统中使用的调度算法，它采用多级优先调度使处理机与外围上作业信息的传输并行操作，采用多道交叉调度实现三道作业流共行。它在配有磁盘、多路器以及多个终端设备的DJS—130机上，允许多至20个用户在终端上同时进行不同的作业。系统采用块式与层次相结合的结构，经过现场的使用和考核，方案是可行的和有效的。在各个终端上可以进行检索、统计或者其他类型作业，响应时间平均不超过两秒。

HOS—130系统具有多路性、独占性、简易交互性及实时性，是一个具有分时兼实时功能的小型操作系统。

一、调度算法

调度算法包括外围设备的管理调度及处理机内多个作业进程的调度这两方面。

1. 外围设备的管理与调度——外进程的调度

这方面管理与调度主要是解决处理机、各外围设备（包括终端设备）并行工作问题，以处理用户随机性的输入输出。

由于多个用户在终端设备上的字符输入输出是随机的，其他外围设备的启仃也是随机的，必须有条不紊地加以处理，其办法是通过多级中断控制程序与各设备的处理程序采用多级优先调度来实现。“中断”是由外围设备在处理机相连的寄存器中设置的定时信号，处理机定时（例如执行完一条指令）地考察这些寄存器一次，一旦发生中断，处理机就自动挂起当时所执行的程序，而转入系统的中断控制与处理程序，以响应来自设备的信号。处理完毕，又恢复原先被中断了的程序，或开动另一个更急的程序。

对于每一个终端设备，有一个相应的字符输入进程及其输入缓冲区；有一个字符输出进程及输出缓冲区；还有一个设备控制表。它们的标识方法及作用如下：

Input (D) 代表设备（或用户）D的字符输入进程，它将设备D缓冲寄存器中的字符，转换为系统内部编码，加以分析，对不同类型字符作不同的处理，如字母或数字状态的转换，无效字符处理，消错与清除处理，作业结束进行挂号处理等，对于一般字符则送入该设备相应的输入缓冲区 INBUFFER D〔O：NDI〕
必要时还将此字符立即送输出缓冲区 OUTBUFFER D〔O：NDφ〕
印出该字符。

Output(D) 代表设备（或用户）D的字符输出进程，将OUTBUFFER(D)的字符取一个出来，送到D的缓冲寄存器，印出该字符。

DCT D〔1：ND〕代表设备（或某一用户）D的设备控制表，它是供 Input(D) 与Output(D)之间通信和交换信息、供系统进行调度控制的一张表，内容见附表。

DCT（设备控制表）

輸入缓冲区已存字符个数	輸出缓冲区末指針
輸入缓冲区取字符指針	輸出缓冲区容量字
輸入缓冲区送字符指針	設备启动特性(1表示已启动，0未启动)
輸入缓冲区头指針	字母狀态字
輸入缓冲区末指針	启动指令
輸入缓冲区容量字	屏蔽字
輸出缓冲区已存字符个数	鏈單元
輸出缓冲区取字符指針	用戶号（或設备碼）
輸出缓冲区送字符指針	工作單元
輸出缓冲区头指針	工作單元

对于其它非终端设备，数据传输过程及其DCT有所差异，这里只将磁盘的设备控制表及时钟设备控制表列出：

調盘字Ⅰ（驅动号、盘面、区号、讀写区数）	再定标計数
調盘字Ⅱ（工作模式、道号）	再R/W計数
調盘字Ⅲ（內存交換地址）	設备碼
盘狀态字	屏蔽字
再定标特征	工作單元
R/W 特征	工作單元

时钟的DCT：

TS（时間片标志）
RTC P（频率）
RTC C（时針）
RTC M（分針）
RTC S（秒針）
RTC T（計数）
屏蔽字

关于调盘进程以及时钟处理进程在讨论作业进程调度时再谈。

有了Input(D)、Output(D)、DCTD、INBUFFER D、OUTBUFFER D之后，还需有一个多级中断控制程序 Iocontrol 以及退中断分析程序 disinterrupt。

调度规则：当中断发生后，由硬件提供的手段以及系统事先的安排，自动地开动

Iocontral（中断控制）

它保护现场信息到一个后进先出的堆阵STACK中，然后对中断源进行分析，若是掉电中断，则转入掉电处理进程。对于设备来的中断请求，按其中断级别先响应级别高的，由Iocontral开动相应的Input(D)或Qutput(D)，处理字符的输入或输出，处理完毕转入

disinterrupt（退中断）

它根据系统作业进程调度算法及有关信号，或者退出中断后恢复现场，返回断点，继续执行被中断的进程；或者退出中断后将现场信息从中断堆阵STACK捞出放到作业进程保护栈保存，而去开动或恢复另一个进程，处理更急的事情。

在进程 Input(D)与Output(D)中。又允许级别更高的数据传输进程所中断，

如磁盘进程可以中断时钟处理进程，时钟处理进程可以中断电传输入输出进程，而电传输入输出进程又可以中断作业进程(见后)等等。中断堆阵 STACK 正是为了多级中断而设置的现场保护栈。

输入输出进程又称为外进程，每个外进程具有潜伏、运行、挂起三种状态。通常它总是处于潜伏状态，中断发生后被激发，转入运行状态，当更高级的中断发生时，处于挂起状态，退出高级中断时，恢复运行状态，输入或输出处理完毕，又处于潜伏状态。

外进程调度与控制图如下（见下页）

2．处理机内多个作业进程的调度

用户的作业程序在内存的执行过程，就叫**作业进程**。从实际情况出发，允许三道作业进程共行，而每一个作业进程又由若干有顺序的步进程来完成。

为了描述作业进程的调度算法，本节采用算法语言ALGOL60（扩充），它允许在源程序中使用汇编语句（Assemble）和增加了一些数据类型。具体扩充如下：

第一，增加机器字变量（word）及寄存器变量（register）。

机器字变量以机器字（二进制或八进制码）为值，用来作为栈区、缓冲区指针、信号灯或栈数组分量的值。

寄存器变量可以取机器字、实型数、井型数、布尔量为值，它对应于硬件的通用寄存器，用于各种运算之中。

机器字变量及寄存器变量在分程序首部加以说明，其作用域与其他简变一样。

第二，增加栈数组（stack array）及字节数组（byte array）。

栈数组以机器字作为分量之值，栈数组用于描述表格和栈区。

字节数组以字符代码为分量之值，它用来存放字符，描述输入输出缓冲区等用。

第三，增加汇编语句（assembly），格式如下：

assembly *begin* 〈汇编指令或机器指令集合〉
end;

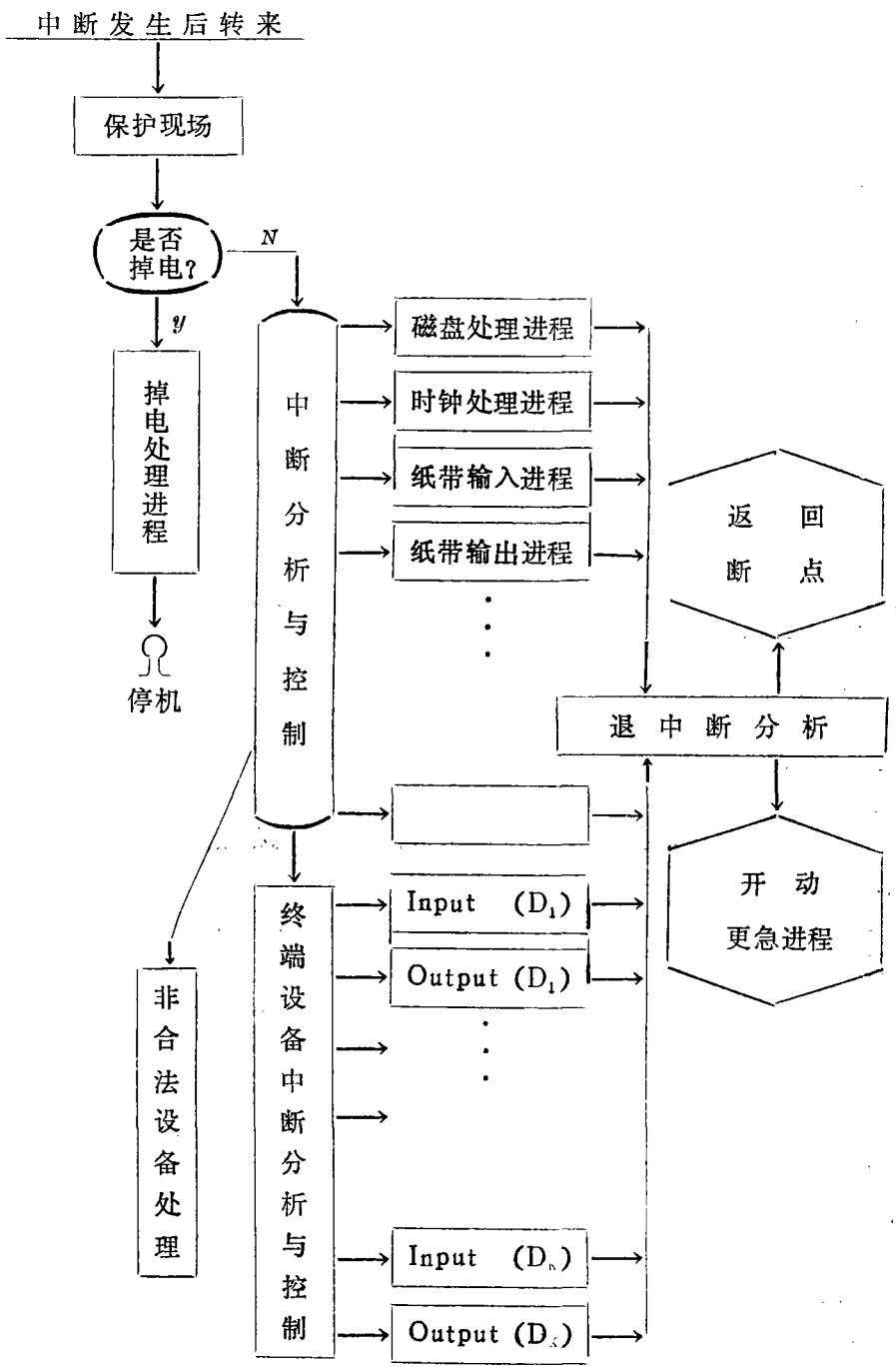
它用于描述难以用ALGOL语言表达或无法表达的操作，汇编语句与代码过程（code procedure）所不同的是，它可以象其他语句一样，直接插入到程序中，而不必通过过程语句来调用，并且可以直接使用本层或外层分程序首部已说明的标识符。采用ALGOL60 写软件比用汇编语言或机器语言显得直观、精炼、容易查错，已有不少这样的做法了，见〔4〕、〔5〕。

(1) 作业的分类及其进程表示

A型——需调盘的检索作业；

B型——不需调盘的统计作业；

C型——小型服务性作业、综合作业或运行时间长的后台作业，为了区分，有时又以CB表示后台作业。



三道作业进程共行的调度算法叫多道交叉调度，它以A道为主流，B道见缝扦针，C道为后台。

A型作业的步进程包括：输入——挂号——选题——编译——调盘——查表——输出处理——字符输出。

B型作业的步进程包括：输入——挂号——选题——编译——计算——输出处理——字符输出。

C型作业的步进程包括：输入——挂号——选题——编译——调盘／计算——输出处理——字符输出。

另外，对每一作业进程还需加上返回解释、挂起、恢复、出错处理等步进程，这些进程的启动有随机性。

各步进程用一个ALGOL过程调用语句表示、过程参数一般为作业进程标识符A、B、C或CB（C道后台作业）。

input (D)与ontput (D)分别为字符输入，输出进程，前面已介绍。挂号进程亦包括在输入进程之中。其他还有

select (X) 表示选题进程，与此进程有关的是作业挂号表JOBT及作业信号灯JS。挂号时，将用户作业名称及用户号等信息送入挂号表，且JS加1。选题时，按优先规则——A型优先，B型其次，A、B皆空选C，同一类型中先来先服务——选一作业，并使该进程转入运行，JS减1。

Compiler (X) 表示编译进程，采用一趟扫描完成编译工作，它从提出相应作业的输入缓冲区 INBUFF 取字符，逐个进行分析和编译，并按规定的信息格式存入X型作业数据区 JOBD，传给后面进程。

running (X) 表示运行进程，从作业数据区取信息和数据加以执行，并将执行结果存入结果缓冲区 JOBR。

list (X) 表示输出处理（排表）进程，从结果缓冲区接过结果，按特定格式表格排列字符，送OUTBUFF。

suspend (X) 表示挂起进程。作业挂号之后，转入**就绪状态**，选题之后转入**运行状态**，因等待外部设备上数据传输或其他原因而挂起时，转入**等待状态**，一旦条件成熟（如数据传输完成）又恢复运行状态，当输出处理完毕时，转入**终结状态**。挂起进程将作业X的现场送入暂挂保护栈保护起来，使X转入等待状态，

restor (X) 表示恢复进程，从暂挂保护栈捞出X现场信息，使之恢复，转入运行状态。

return(X) 表示返回解释进程，X从等待转入运行时，该转入何处继续呢？就由这一进程来解释和控制，返回点已由Suspend (X)保护在暂挂保护栈中。

凡因外部设备中断的现场，在不换道情况下，则以Iosuspend、Io Restor及Io Return进程实现挂起与恢复。

error (X) 表示出错处理进程，将出错的作业进程撤消，强迫终止，并印出信

息告该用户。

〈2〉与调度算法密切相关的两个进程及其表示

Disk (X, C_1, C_2, C_3)表示调盘进程，X为作业进程标识符，C_1,C_2,C_3为调盘字，调盘进程又由定位（Diskseek）、读写(Diskrw)、未端错处理（enderror）以及其他错误处理进程组成，当A道或C道作业进程发出调盘命令后，便由定位进程挂起该道作业X,且调盘信号灯SDX置1。调盘进程中,定位平均等待时间70ms，找区平均等待20ms，交换数据又需若干ms，这时为了使处理机不闲，因此又开动B道（或C道）作业进程，当Disk结束时，会发出R/w完成中断信号，将B或C挂起，返回X道继续。调盘进程与其他进程的通讯与同步主要通过其设备控制表DCT来实现。

调盘进程的开动与结束，正是各道作业进程之间的切换点，这种切换总是使处理机不闲，是一种很自然的方式，比起时间片轮转法调度来说，大大减少了开销。

调盘进程中，定位完成、R/w完成、再定标完成以及盘出错时都有中断信号，会转入相应的中断处理进程，由于磁盘是中断级别最高的设备，它可以中断其他设备的中断处理进程，因此正确处理磁盘来的各种中断，以及在退中断分析时正确控制其转向，乃是实现本调度算法的一个关键。

为了退中断分析进程能得到有关消息，在调盘的各种中断处理中均给出相应的标志,特别是R/w完成时将R/w完成标志置1,而将调盘信号灯置0,即DRWS:=1;SDX:=0;

RTC(q)代表实时时钟处理进程。它主要用来计时和中断后台作业的运行。当给出频率q以及启动时钟之后，经过时间间隔为1/q时，时钟会发出中断请求，由中断分析控制转来此RTC进程,它统计中断次数，即RTCT:=RTCT+1；当RTCT等于给定值时，就发出时间片到的信号TS:=1，并暂停时钟。如时间片未完，则在RTC内处理时针（RTCC）、分针（RTCM）、秒针（RTCS），并继续启动时钟。用算法语言描述如下：

```
Procedure    RTC(q);
begin    RTCT:=RTCT+1;
  if  RTCT≥TSO  then
  begin  TS:=1;  goto  out;  end;
  if  RTCT=q  then  RTCS:=RTCS+1;
  if  RTCS=60    then  RTCM:=RTCM+1;
  if  RTCM=60    then  RTCC:=RTCC+1;
  assemble  NIOS  RTC;
Out:    end;
```

其中TSO为时间片比较值。

在本调度算法中，C道的后台作业按时间片进行剥夺式调度，一旦进入C道后

台作业，便启动时钟①，而且给计时指针及统计时间的计数器RTCT清0，当恢复后台作业进程时，也要启动时钟①，但时、分、秒针不清0，即

进入后台作业时：

```
procedure STRTC;
begin RTCC:=RTCM:=RTCS:=0; SCB:=1;
    TS:=0;    RTCT:=0;
    assemble    begin  LDA   AC, q;
                       DOAS  AC  RTC;
                end;
end;
```

恢复后台作业时，则上述语句第一行删去即可。

当后台作业运行了一个时间片之后，就剥夺其运行权利，将处理机分配给A道或B道作业。这就防止了某些时间长的后台作业垄断处理机。但对于C型的综合作业及小型服务作业来说，没有实行这种剥夺，因为这些作业或者运行时间不长，或者中间需要调盘，在调盘等待中又转入B道了。

(3) 信 号 灯

信号灯是进程之间同步与通讯的工具，它用一个整型变量或机器字变量表示。用一个信号灯或几个信号灯的组合状态来管制某进程，决定它是否可以推进或等待。

信号灯的信号由某些进程发出，而又由某进程所接收，发送信号的进程在发信号之后，它自己继续进行，等信号进程则根据信号来决定行动。

这里用了下列信号灯：

JSA、JSB、JSC分别表示A、B、C型作业信号灯（job semaphore），初态为0，挂号进程发信号，计数加1，选题进程接收信号，计数减1。

SA、SB、SC分别表示作业进程A、B、C的运行状态，初态为0，选题时发信号，置1，表示有该道作业进程从就绪状态转入运行或等待。输出处理完毕或出错处理时，置0，表示该作业进程终结。对于C型后台作业，还有一个特别信号SCB，进入时置1，终结时置0，供时钟处理进程判断用。

DSA、DSC分别表示A道调盘、C道调盘信号灯，进入调盘进程时发信号，置1，盘读写完成置0。

TS表示时间片信号灯，初态为0，启动C进程或恢复C进程（当C道为后台作业时）时，开始计时，当C运行一个时间片时，由RTC进程发信号，即TS:=1，调度程序接到此信号，便去开动suspend(C)，且TS:=0。C转入终结时，亦置0。

① 若是因等待A的调盘完成而转入或恢复后台作业，则不必启动时钟，待A调盘完成便自动剥夺后台运行而返A。

IS 表示中断层次信号，进一层中断加1，退一层中断减1。

mask 表示关中断信号灯，mask为1不响应中断，为0才允许中断。

(4)缓冲区、栈区与作业控制块

缓冲区与栈区前面已分散谈过，这里只是罗列一下而已。它们用栈数组表示：

STACKIO〔1:n1, 1:n2〕	表示多级中断堆阵
INBUFFD〔o:nDI〕	表示设备D的输入缓冲区
OUTBUFFD〔o:nDo〕	表示设备D的输出缓冲区
DCTD〔1:nD〕	表示设备D的设备控制表
JOBTX〔1:nTX,1:2〕	表示X型作业挂号表
JOBDX〔1:nIX〕	表示X型作业数据区
JOBRX〔1:noX〕	表示X型作业结果区
SAVEX〔1:nSX〕	表示X进程现场保存栈
STACKA〔1:nA〕	表示A道作业进程保护栈
STACKB〔1:nB〕	表示B道作业进程保护栈
STACKC〔1:nC〕	表示C道作业进程保护栈

对于每一个作业进程，有相应的作业控制块JCB，它包括该作业进程的运行状态、信号灯、用户号、优先度、保护栈等内容，供调度之用。调度程序对每个作业进程来说，只关心其名字和JCB，而对其内部操作是不感兴趣的。

(5) 算法描述

```
begin
        integer JSA, JSB, JSC, SA, SB, SC, DRWS, DSA, DSC, TS, IS,MASK,
              SCB;
        stack array STACKIO 〔1:n1, 1:n2〕,…
        register ACo, AC1, AC2, AC3;
        Procedure input (D);
                 Value D; integer D;…
        Procedure output (D);
                    Value D; integer D;…
        Procedure Select (X);
                         Value X; integer X;…
        Procedure Compiler (X);
                value X; integer X;…
        Procedlure running (X);
                value X; integer X;…
        Procedure suspend (X);
                value X; integer X;…
        Procedlure restor (X);
```

```
        value X; integer X;…
Procedure return (X);
        value X; integer X;…
Procedure error (X);
        value X; integer X;…
Procedure diskseek (X, C1,C2,C3);
        value X, C1, C2, C3;
        integer X;
        word C1, C2, C3;
procedure rtc (q);
        value q; word q;…
procedure Iorestor;…
procedure Ioreturn;…
procedure Iocontrol;…
procedure strtc;…
procedure disinterrupt;
    begin…
if TS=1 ∧ SCB=1 then
Begin suspend (CB);
          goto START;
        end;
        if IS >1 then goto LIO;
        if DSA=1 ∨ DRwS=1 then
begin if SB=1 then suspend (B) else suspend (C); DSA:=DRwS:=
                restor (A);
                return (A);
        end;
    if DSC=1 ∧ DRWS=1 then
    begin suspend (B); DSC:=DRWS:=0;
        restor (C);
        return (C);
    end;
LIO: begin IORestor;
          IOReturn;
            end;
        end;
        . . . . . .
WAIT: if JSA=O ∧ JSC=O then
        goto WAIT;
START:
```

```
        LA1; if JSA=0 then goto LB;
                select (A); JSA:=JSA-1; SA:=1;
                compiler (A);
                diskseek (A, A1, A2, A3); SDA=1;
                suspend (A);
        LB: if SB=0 then goto LBI;
                revtor (B);
                return (B);
        LBI: if JSB=0 then goto LA;
                select (B); JSB:=JSB-1; SB:=1;
                compiler (B);
                running (B)
                list (B); SB:=0;
        LA: if DSC=1 then goto LBI;
                if JSA ≥1 ∧ DSA=0 then goto LA1;
        LC: if SC=0 then goto LC1;
                restor (C);
                return (C);
        LC1: if JSC=0 then goto LA1;
                select (C); JSC:=JSC-1; SC:=1;
                compiler (C);
            if SCB=0 then
            legim diskseek (C,C1,C2,C3);
                    DSC:=1;
        suspend (C);
        goto LB1;
        end;
          running (CB);
          list (CB); SC:=SCB:=0;
          goto LA1;
ContinueA:  running (A);
            list (A), SA:=0;
            goto LA1;
ContinueC:  running (C);
            list (C);   SC:=SCB:=0;
            goto LA1;
end
```

其中 ContinueA, ContinueC 是由Return (X)转来。

3．调度算法所遵循的优先规则

不论是那些与终端设备，外围设备相关的输入输出进程，还是各类型作业进程，都是主控程序调度之下协同工作，通过中断技术、信号灯及数据缓冲区的方法，实现同步与通讯，有节奏地运行。

在整个调度中，遵循如下优先规则：

第一，磁盘数据通道优先；

第二，各外围设备中断处理优先，当若干设备同时申请中断时，按中断级别高低择优响应；

第三，用户输入字符优先；

第四，字符输出优先；

第五，作业挂号优先；

第六，A道作业优先；

第七，B道作业见缝插针；

第八，C道作业当作后台处理；

第九，对于同型作业，先来先服务。

其中第一到第五通过中断技术实现，后面几条通过多道交叉调度实现。

二、系统的性能参数

为了衡量系统的性能，这里给出本系统基本参数的计算结果。

性能参数有下列几种：

(1) 系统的负荷量——最大用户数及最大吞吐量（作业个数/小时），平均吞吐量。

(2) 资源利用率——主要是指处理机利用率（=作业执行时间/总时间）。

(3) 平均响应时间——用户提交作业完毕到系统给出回答的时间。

(4) 平均响应比——响应时间与该作业实际服务时间之比，它反映了由于存在其它作业及所用的调度算法，使该作业所感觉到的处理机速度的变坏程度。

计算这些参数的步骤是这样：首先给出A、B、C型作业的平均执行时间（通过统计执行指令或实测而得），然后对于不同的作业比例以及变化的用户数目，计算出一个参数表。计算中使用了下列记号

UA、UB、UC分别为进行A、B、C型作业的用户数目。

TA、TB、TC分别为A、B、C型作业的平均执行时间，以ms为单位。

RTA、RTB、RTC分别为A、B、C型作业的响应时间，以ms为单位。

JA、JB、JC分别为一个终端上进行一种作业（A、B或C型）的吞吐量(个/分钟)。

SA、SB、SC为井个系统的各类作业吞吐量（个/分钟）。

RRA、RRB、RRC代表响应比。

UE代表处理机利用率。

DT代表调盘平均等待时间(ms)。

经过计算与实测，作业的平均执行时间与吞吐量的值如下：

TA＝50＋DT，DT＝92ms，

TB＝330ms，

TC＝100ms（后台作业除外）。

JAmax＝10个/分钟　　JA正常＝5个/分钟

JBmax＝2个/分钟　　JB正常＝1.7个/分钟

JCmax＝5个/分钟　　JC正常＝3个/分钟

参数计算结果见附表。

表中，RTAmax、RTBmax及RTCmax是响应时间的最大值，它们是在A、B、C三类作业均处于有最长作业队列的情况下计算的，称之为“最坏情况”。现在让我们用初等概率理论来估计“最坏情况”出现的概率。

我们把作业的来到看成是独立的、随机事件，对于本系统，可以假定：

<1> 作业流是平稳的。对于任何t>o及井数k≥o，在时间区间(a, a＋t)内有k个作业来到的概率对于任何a≥o都是一样的。即在给定时间间隔内来到的作业数目仅与时间间隔的长度有关，而与时间起点无关。

以$P_k(t)$来表示这个概率。

<2> 作业流无后效性。$P_k(t)$与时刻a之前来到的作业情况无关，或者说与系统以前的历史无关。

<3>作业流具有普通性。在任何一个很小的时间间隔之内，有多个作业（两个或更多）来到的概率可以忽略不计。

这样的作业流是一种poisson流，可以推导出作业来到的概率分布为poisson分布[6],[7]，长度为t的时间区间内有k个作业来到的概率可用下式表示：

$$Pk(t)=e^{-\lambda t}\ \frac{(\lambda t)^k}{k!}\quad (k=0,1,2,\cdots\cdots) \tag{9}$$

其中λ是一正常数，叫做来到率。

若以F(t)表示“来到间隔”（即两次相继的作业来做之间的时间间隔）小于或等于t的概率，则有[1]

$$F(t)=1-e^{-\lambda t} \tag{10}$$

这表明“来到间隔”为指数分布，且其平均值为

$$E(t)=\int_o^\infty tdF(t)=\frac{1}{\lambda} \tag{11}$$

在一段时间T内，平均来到的次数为λT。从我们的实际观测，$\lambda^{-1}=15$秒。

在本系统中，所指的“最坏情况”是在一个最短的作业执行时间内每个用户至少提交了一个作业。即在50ms内来到20个以上作业。

以$\lambda^{-1}=15$(秒)，$t=50ms$，$k=1,2,\cdots\cdots 20$代入(9)得出下表：

K =	1	2	3	4	5
P =	0.00332	0.000006	0.61×10^{-8}	0.51×10^{-11}	0.34×10^{-14}
K =	6	7	8	$\cdots$	20
P =	0.19×10^{-17}	0	0	$\cdots$	0

这表明在50ms内有1个作业来到的概率为千分之三，已很小！而两个以上的作业来到的概率可以忽略不计，而七个以上作业来到(包括最坏情况)的概率为0了！

如果从“时间间隔”来看，利用<10>式也同样算出“最坏情况”出现的概率完全可以忽略不计。

因此我们可以按平均特性来估计RTA、RTB和RTC之值。作业的“来到间隔”平均为15秒，在15秒内，足以处理每个用户提交的两个以上长达330ms(即B类作业平均时间)的作业，所以每个作业来到之后，几乎不必等候，立即可以处理，也就是平均响应时间可以用其执行时间加上调度时间来表示，而调度时间在本系统中几乎可以忽略。从实测和计算结果告诉我们，平均响应时间不超过2秒。

关于处理机利用率及负荷量问题，从表看出，在没有后台作业的情况下，利用率还不超过20%，这说明用户数目还可以增加四倍左右，即可以多至约80个用户。但是，增加用户却使响应时间成倍增加，（不管采取什么算法都如此），而且由于内存所限，实际上是难以实现的。

对于本系统来说，增加用户不是方向，最理想的方案是对于作业提出这样的限制：在A、B作业进行的同时，适当搭配一些后台作业，使处理机满足A、B之外的时间由后台作业包下来，这样就可以大大提高处理机利用率，取得满意的效果。

参 考 资 料

〔1〕. per Brinch Hansen: 操作系統原理1973.

〔2〕. G. M. Bull 等: 分时系統1971.

〔3〕. D. W. Barron: 計算机操作系統1971.

〔4〕. Alan C. Shaw: The Logical Design of Operating Systems. 1974.

〔5〕. P. J. Brown: writting Software in ALGOL. SOFTWARE V.4. NO.2, 1974.

〔6〕. 王梓坤: 概率论基础及其应用 科学出版社1976.

〔7〕. А. Я. 欣钦: 公用事业理论的数学方法 科学出版社 1958.