PAX系统8632用户技术手册

与

IMIGIT用户指南

中国科学院成都计算机应用研究所情报室

PAX system 8632 用户技术手册

Baker & Rabinowitz, Inc

刘 洪 译

时 建 校

PAX system 8632 用户技术手册

目　　录

附　　录

IMIGIT 用户指南

目　　录

一、PAX 简 介

PAX 是 Baker & Rabinowitz 公司开发的一种多任务、实时操作系统，它可以驱动多个独立的程序就好象它们是同时被执行的。

这里介绍的PAX的一个特别版本是专为IBM标准个人计算机设计的。

PAX具有模块化的结构，从而便于软件开发者使用现今流行的标准硬件（PC系列）和软件（DOS及许多应用程序）来构造高质量的软件系统。在PAX上可以开发许多适用于办公室、娱乐、工厂、工业和实验室的应用软件系统。

尤其突出的是，PAX提供了许多可用的软件块。借助这些模块，开发者可以建立一整套符合高性能实时和多任务要求的应用系统。

在设计一个基于PC的系统时，能有其它一些参考手册供你使用常常是十分有用的。我们推荐以下几本关于IBM PC的手册：

IBM DOS Technical Reference Manual

IBM Technical Reference Personal Computer

DOS技术指南描述了DOS的内部，DOS系统调用的描述特别有用。IBM PC 手册为程序员描述了硬件和BIOS。在建立一个高性能应用系统时这些手册是非常宝贵的。

1.1 PAX是什么？

PAX包括几个可由开发者以任意合适的方式使用的软件包。PAX有如下几个模块：

1. PAX操作系统
2. PAX PC设备驱动模块
3. PAX 软件工具包
4. PAX PC初始化任务

及其它几个模块。

操作系统模块提供系统原语服务和控制在PAX下运行的应用程序。

设备驱动模块是一些高度优化的汇编语言子程序，供屏幕、键盘、打印机和中断驱动的异步输入输出及其它一些输入输出设备使用。

PAX 软件工具是可选的但十分有用的模块。它提供了中断向量管理和其它一些已被证明是十分有用的功能。

初始化任务是在 PAX 下运行的一个程序，在所有的 PAX 环境里它都是用来初始化 PAX 下运行的应用程序。

1.2 PAX 如何使用PC？

从软件角度看，IBM PC及其兼容机有三个层次，程序透过这三个层次与外部世界

打交道。这些层次是：

1. DOS层
2. BIOS层
3. 机器层

PAX在以其最迅速最简洁的方式完成任务时利用了所有这三个层次。

DOS层是最高的层次，它提供了一些有用的服务例程，如打开和读写文件、目录检索及其它许多例程。然而因为它也是最慢的，高性能的工业应用系统常常需要优化的屏幕和通讯驱动模块。这就是引入PAX设备驱动模块的原因。

BIOS层次是可用的I/O例程的“第二层”。例如使用BIOS例程作屏幕输出要比使用DOS的更新速度快。BIOS提供了一些有用的功能，如选择屏幕滚动。然而DOS仍是必要的，因为BIOS并不提供大多数系统所需要的高级磁盘I/O功能。

最后一层是机器层次。应用软件可以与任何外设直接对话；只要软件开发者有足够的耐心去编写和调试这些程序。PAX设备驱动模块多数是与硬设备直接对话以满足许多实时应用环境所需的高性能输入输出要求。

PAX可以使软件开发者最好地使用这三个层次，开发者是确定他的系统如何与外部世界交互的最终选择者。

1.3 PAX是如何开发出来的？

PAX系统软件存在一个5.25的双面双密度磁盘上（该盘最多容纳360 KB）。该软件可与原有的很多用其它语言开发的目标代码模块相链接。建立一个基于PAX的系统的详细指导将在第3节中叙述。

PAX系统至少应有一个8086/8088 IBM或Microsoft DOS宏汇编语言程序。PAX软件包提供的所有源代码都是用汇编语言写成。

汇编语言也必须用来建立一个“界面”程序，即把特定编译程序的参数传递方法转换为PAX的方法。对这些方法，我们将以特定的编译程序为例来详细说明。

因为每个编译程序可能以不同的方式将参数传到子程序，故每种编译程序必须要有自己的界面程序。这样，一个用PL/M建立的模块和另一个用C语言建立的模块可以各自调用一个操作系统原语。以下文件由PAX开发盘提供：

目录	文件名	说明
PAX	PAX.OBJ	可连结的PAX原语系统
	TASK0.ASM	初始化源代码
	TASK0.OBJ	初始化目标代码
	DRIVERS.ASM	PAX PC驱动模块源代码
	DRIVERS.OBJ	PAX PC驱动模块目标代码
	TOOLS.ASM	PAX PC工具包源代码
	TOOLS.OBJ	PAX PC工具包目标代码
DEMO	（上面各文件都在此目录里）	

目录	文 件 名	说 明
DEMO	SERIAL.ASM	串行输入示范源代码
	SERIAL.OBJ	串行输入示范目标代码
	PARALLEL.ASM	并行输入示范源代码
	PARALLEL.OBJ	并行输入示范目标代码
	TYPFILE1.ASM	显示文件1示范源代码
	TYPFILE1.OBJ	显示文件1示范目标代码
	TYPFILE2.ASM	显示文件2示范源代码
	TYPFILE2.OBJ	显示文件2示范目标代码
	KEYSTROK.ASM	等待击键源代码
	KEYSTROK.OBJ	等待击键目标代码
	DEMOLINK.BAT	示范链结批文件
	DEMOLINK.LNK	示范链结文件表列
	DEMO.EXE	可执行的示范文件

PAX 目录包含那些可用于建立一个应用系统的最基本文件。它们包括软件开发文件的未修改过的和“清洁”的付本。

DEMO目录包含那些被用来建立一个示范系统的文件。目录中包含一个可执行文件DEMO.EXE，它完成一个简单的多任务系统示范。所有用来建立DEMO.EXE的应用源代码也包含在该目录里。

软件开发者可把PAX和附带的目标模块相链接建立一个特殊的可执行的应用程序。

1.4 多任务系统简介

对那些不完全熟悉实时和多任务操作系统理论的工程师和程序员，下面几节包含几个可能令人感兴趣的论题。

1.4.1 多任务系统的优点

多任务是指一个计算机操作系统可以独立地运行两个或多个任务（作业或程序）就象它们是同时被执行的一样。大多数 IBM PC 和兼容机所配置的DOS操作系统对所有用途来说都是个批处理系统，批处理系统一次最多只能驱动一个进程。

多任务系统广泛应用在现代机器人、事务处理、娱乐和多用户计算机系统中。多任务系统允许软件开发者建立一些独立的程序，它们为最终应用场合的各个特定方面提供服务。用这种方法可获得高可靠性、易维护性和高效率。

以机器人应用为例，控制机器人的软件系统必须能够：

1. 根据用户的程序移动每个活动轴。
2. 通过串行通道与主计算机通讯。
3. 响应来自操纵者控制台的命令。
4. 其它任务。

仅前三项任务就表明这将是个复杂的项目。一个选择是设计一个具有这些功能的联

机、单任务程序，不幸的是，没有那个一流的机器人软件是在一个单任务、批处理环境写出的。

另一个选择是建立一个多任务系统，它可使软件设计者把每一项工作分解成独立的模块，例如通讯任务能响应来自主机的命令并把这些请求置入全局有贮区。在轴控制任务执行时它能检查全局区域并依其指定请求行动。

从本质上说，多任务系统是由于使系统具有可扩展性从而扩展了应用的前景。当需要新的功能时，这一增加的任务可以被建立并执行，而不需对现有程序作什么改动。

1.4.2 多任务系统的性质

所有的多任务操作系统都具有一些共同的特性，下面就详细说明这些性质。

1.4.2.1 并发性

逻辑并发性是指多个操作可以（从表面上看来）同时进行。例如，在一个任务从串行接口读入数据的同时另一个任务可以把数据传送到打印机。对用户来说，这些操作仿佛是同时的。实际情况是：CPU控制权在任务之间迅速地交换，便产生任务并发性。

1.4.2.2 异步性

异步性是指系统输入的固有不可预测性。例如，在一个字节到达串行口的同时，用户敲击键盘。操作系统提供的同步子程序保证了对应于每一个处理输入的作业能以适当的方式协调行动。

1.4.2.3 资源共享性

资源共享是指一个特定系统的有限I/O 能力。两个并发执行的作业都要打印一个文件是说明这一性质的好例子。如果系统不提供资源共享，将导致两个硬拷贝混杂在一起。一个具有有限资源的系统，例如打印机，要对两个竞争作业进行有效的仲裁从而使两个完整的硬拷贝相继输出。

1.4.3 多任务操作系统的类型

就目前来说，基本上有两种实时多任务操作系统。

第一种（如PAX）是一种简单有效的时间分享系统，它不需或只需很少的系统生成（即让系统适合于特定的应用）。分时轮转系统中的每一个作业对CPU具有同等的要求，尽管它们受到不同的调度。

第二种是一种较复杂的系统，其中每一个作业都得到一个执行优先权，优先权决定了作业能得到多少CPU时间。

这两种系统都各有其优缺点。时间分享系统保证了每个作业都能得到CPU 时间而不管系统的负荷有多重；它事先也无需或仅需很少的系统生成（必须组织和剪裁的时间）。

分配优先权的操作系统在特殊的控制应用中十分有用。仍以机器人为例，完成运动控制的作业显然比串行输出作业具有较高的优先权。然而系统生成和调试周期常需数周或数月，这样的系统也不能保证并发作业都将会得到执行。

在大多数IBM及兼容PC机中，操作系统可使用的唯一实时中断时钟是55ms。在无须添加硬件或刚性环境修正的软件的情况下，分时轮转操作系统就具有非常重要的意义。

此外，DOS功能处理程序多半都是不可再入的，一个请求DOS功能调用，被中断的低优先权作业是不能安全地把控制权交给也想使用DOS的高优先权作业。这些限制表明一个时间分享系统可能是最安全、最简单和最可靠的实时多任务解决办法。

象PAX这样的时间分享操作系统被使用于工厂自动化系统。电视游戏、通讯控制器和其它许多计算机系统之中。固有的可靠性、简单性和PAX的性能是使用PAX的强有力的保证。

二、PAX 技术概论

如前所述，PAX不仅是个操作环境。同时PAX也提供了工具、设备驱动模块和可执行作业（示范系统的一部分）来帮助开发者构造高速、实时系统。这些模块将在本节里加以详述。

2.1 PAX 操作系统

PAX可以认为是一个实时的、多任务和具有优先权（或时间分享）的操作系统。这一操作系统为许多独立的程序（作业）提供了一个环境，使得它们似乎是并发执行的，并能分享计算机系统的有限资源。

在某一时刻PAX操作系统被初始化（使用系统调用SYSINI），分时系统便开始工作。PAX所做的第一件事情就是请求一个初始化作业（TASK0），这个作业建立起PAX实时中断并完成其它必要的核心操作，例如启动一些应用作业，于是系统便建立起来了。

2.1.1 作业状态

作业是独立地被PAX操作系统执行的各个程序。为跟踪这些程序，PAX为每个作业指派一个唯一的作业描述子（作业ID）。PAX正是通过这些描述子管理系统中的作业。

作业描述子包含一个状态单元及其它一些信息。状态单元描述了当前作业状态。表2.1.1—1给出了将会遇到的大多数作业状态。

每个描述子中的作业状态单元使得PAX可迅速确定作业的性质。

2.1.2 PAX 操作系统核心

PAX操作系统由如下四个部分组成：

1. 内核
2. 实时中断处理程序
3. 系统原语
4. 作业描述子块

2.1.2.1 内核

内核是PAX的调度者。内核本质上是一个决定下一个可执行作业并以一个合适的

表2.1.1—1　PAX作业状态

状态	解　释
AVAILABLE	该作业描述子未被指派，可用于新作业；
SUSPENDED	该作业描述子已被指派但作业现在不在运行同时也未排定在将来运行；
RUNNING	相应的作业正在执行；
SLEEPING	相应的作业过去在执行，但它通过请求系统调用 SYSSLP或SYSPEP，或间接地请求系统调用SYQPND悬挂等待讯号，而把自己置于睡眠状态；
INTERRUPTED	相应的作业过去在执行但被PAX所中断，因为它已经耗尽了一个作业所允许的连续执行时间；

方式授予这个作业控制权的单一程序。例如，如果一个作业要被唤醒，只须简单的将控制权传到它进入睡眠的地方。

内核能够并发地驱动32个作业。较大的应用系统，例如，例如PAX系统8648（能驱动达2048个作业）也能被构造出来。

2.1.2.2　实时中断处理程序

PAX需要实时中断。IBM及其兼容系列提供一个55ms的高优先权中断，用于更新日时钟。PAX使用这一周期中断来更新内部系统时钟并中断一个超时作业。PAX的实时中断处理程序是SYSRTI。

为能执行软件开发者的前台作业，在实时中断处理程序中使用“INT 60H”指令，提供了一个未用的中断（中断60H，向量地址为0000：0180H），通常，该向量指向一个“IRET (interrupt return) 指令，无动作发生。

INT 60H中断向量指向TASK0.ASM模块中的USRRTI。软件开发者可以使用这一中断处理程序来执行他自己的前台作业。

2.1.2.3　系统原语

系统原语是控制应用系统操作的过程，PAX原语提供基本的控制操作：起动和终止一个作业、信号灯实现、作业协调等。

表2.1.2.3—1给出了系统原语的一个说明，详细说明见第4节。

2.1.2.4　作业描述子块

它是作业描述子存贮的地方。该块是16字节描述子的连续阵列。第一个元素描述作业0，最后的描述作业31。

debug状态下，开发者可通过搜寻串“Task Descriptors”来定位作业描述子块，这个16字节的字符串直接加在作业描述子块之前。每个16字节的描述子映射见表2.1.2.4—1。

每个作业的当前状态简洁地由每个描述子中的作业状态字节表示。状态字节各位意义见表2.1.2.4—2。

表2.1.2.3—1 PAX系统原语概览

名　称	解　　释
作业管理	
SYSCRE	建立一个作业
SYSDEL	删除一个作业
SYSDOS	请求DOS功能处理程序（等于INT 21H）
SYSDTI	禁止作业中断（作业切换）
SYSETI	允许作业中断（作业切换）
SYSEXT	终止多任务系统并退出到DOS命令层
SYSINI	初始化多任务系统，把控制权交给TASK0
SYSKIL	终止一个作业（不能用于作业终止自己）
SYSPER	高度可重复的周期请求的睡眠
SYSSLP	睡眠一段特定的时间
SYSSTR	起动一个作业
作业 通讯/同步	
SYQACC	从队列中接收一个讯号
SYQCRE	建立一个队列
SYQINQ	完成一个队列查问
SYQPND	挂起等待队列中的一个讯号
SYQPST	发送一个讯号到队列中

表　2.1.2.4—1　作业描述子的结构

字　节	描　　述	字　节	描　　述
0	作业号	8—9	初始栈段
1	作业状态	10—11	内容栈指针
2—3	初始的IP码偏移量(起始地址)	12—13	内容栈段
4—5	初始码段址（起始地址）	14—15	系统时钟参照
6—7	初始栈指针		

表　2.1.2.4—2　作业状态字节

位模式	状态	解释
10000000	INHIBIT	作业执行（手动地）被禁止
00100000	AVAICABCE	描述子未被指派，可以使用
00010000	INTERRUPT	作业被中断（限时耗尽）
00000100	ASLEEP	作业睡眠(SYSSLP或SYSPER调用)
00000010	AWAKNED	作业从睡眠状态被唤醒
00000001	RUN	作业就绪等待执行

禁止位需要作些说明。禁止位（状态＝1××××××××H）不能由PAX系统设置和清除，它是被提供来让开发者手动控制作业执行。

假若在一次典型的调试中，一个全局存贮单元被某个应用作业不正确地消除了。开发者不能猜测出是哪个作业干的，他必须反复地运行，花费大量时间来找出"罪犯"。而现在通过每次禁止一个有嫌疑的作业，他就能迅速确定这个作业。

通常的非禁止作业的状态字节如下所示。

表 2.1.2.4—3 普通作业状态字节

字节	解 释
00H	作业描述子已被分配，但作业不在运行
01H	作业第一次或在中断以后执行
03H	作业在被唤醒后执行
05H	作业适合于执行但当前为睡眠
11H	作业适合于执行但被中断
20H	作业描述子未被分配

2.1.3 使用系统原语

使用系统原语非常简单。下面这个利用Pascal及汇编语言的例子展示了可以多么容易地把系统调用与代码结合起来。然而我们有必要记住，每种语言传递参数的方式稍微有些不同。下面的PSYSSLP是一个汇编语言程序，它把编译程序的参数传送方式转换为PAX的方式。

表2.1.3—1 Pascal下使用系统原语的例子

```
PROGRAM EXEC_TEST_1 (INPUT, OUTPUT) ;
PROCEDURE PSYSSLP (MILLISECONDS; INTEGER) ; EXTERN;
    (PSYSSLP is PASCAL Interface routino to SYYLP)
BEGIN
        (Loop forever)
       WHILE (1)
       BEGIN
              (sleep about a socond using interface routine)
             PSYSSLP (1000) ;
              (indicate that a seoond has slapsod)
             WRITELN ( 'Another second has elapsed' ) ;
       END;
END.
```

这是一个简单的程序。十分显然，无限循环中每次有一秒钟的睡眠状态，然后写屏幕。下面这个汇编程序基本上也执行同一功能。

表2.1.3—2　汇编语言下使用系统原语的例子

```
title EXEC_TEST_1 System Executive Test Program 1
; …DEFINE EXTERNAL PROCEDURES……
;
extrn SYSSLP: far                    ; system executive SYSSLP
extrn CONSO: far                     ; CONSO is a member
                                     ; ofthe PAX-PC driver
                                     ; provided with PAX
; …DEFINE DATA SEGMENT……
;
dseg segment para                    ; Data Segment
message      db'Another second has elaped' ; 0DH, 0AH, '$'
dseg ends
; …DEFINE CODE SEGMENT……
;
cseg segment para                    ; Code segment
     assume cs: cseg, ds: dseg
main proc    far
       mov ax, dseg                  ; align data segment
       mov ds, ax                    ;
                                     ;
main_1: mov ax, 1000                 ;
        call SYSSLP                  ; sleep about a second,
        call  CONCO                  ; use console string output
                                     ;
        mov dx, offset message       ;
                                     ; procedure,
        jmp main_1                   ;
main endp
cseg ends
    end
```

2.1.4　队列

PAX另外提供了5个系统原语来控制讯号排队（见2.1.2.3）。PAX最多允许32个队列同时打开。每个队列长度可为1到64个双字长登记项，采用先进先出的方式管理。

队列完全由PAX管理并可通过如下一些调用被软件开发者利用

SYQACC	无条件的从一个队列接受一个讯号
SYQCRE	建立一个队列
SYQINQ	完成一个队列的查询
SYQPND	等待来自一个队列的讯号
SYQPST	发送讯号到一个队列

SYQACC无条件地从一个队列接受一个讯号或者返回队列空错误而不做作业切换；SYQCRE建立一个新队列，只要有足够的空间；SYQINQ返回一个特定队列的查询状态，该状态包括队列中当前的项目数和队列中先出双字长的拷贝（即先出登记项并未从队列移走）；SYQPND挂起等待来自一个特定队列的讯号，并能自动地在一个时间隔后再作检查，如果这一时间过后仍未收到任何讯号，那么SYQPND将返回一个队列空错误；SYQPST发送讯号到一个队列或回答该队列已满。

2.2 PAX个人计算机驱动模块

PAX软件包包括一组为IBM PC系列（PC、PC/XT和PC/AT）开发的设备驱动模块。这些设备驱动模块是一些速度上高度优化的汇编程序。在大多数情况下，这些程序在机器层上运行而不用请求BIOS或DOS。

在这样的低层次上运行既有优点也有缺点。缺点之一是来自PC系列的物理结构可能改变这一事实，这样，设备驱动模块可能在下一代计算机上失效。

然而使用“机器层次”的程序也有一些优点。这些程序的执行速度比类似的DOS程序快许多倍，并且在许多应用场合这一速度不是多余而是必要的。此外，这些程序的构造还允许开发者决定哪个是最适合于他的应用。例如，需要屏幕更新的作业不需要调用特殊的设备驱动模块而可轻易地请求BIOS来代替。

下面描述了设备驱动模块的一般特性。

2.2.1 异步通讯驱动模块

初看，PC软件的BIOS层仿佛提供了具有接受能力的异步通讯驱动模块。然而，BIOS“异步”通讯程序的一个严重缺陷是串行输入不是由中断驱动。不管查询方式设计得如何好，总不能与缓冲的、中断驱动的串行输入方式相比。

提供给PC串行口的PAX设备驱动模块是：

COMCI
COMCO
COMINI

COMCI (Character Input) 是一个允许应用作业从中断缓冲器中读一个字符的例程。缓冲器是个1024字节长、由COMxIS(对应的中断服务程序)支持的环形阵列；

COMCO (Character Output) 是一个输出一个特定字符到串行口的例程，字符输出不是由中断驱动；

COMINI (Communication Initialization) 是允许调用者起动/再起动通讯（包

插波特率、奇偶校验等）和复位中断驱动的输入缓冲区的初始化程序。

此外，还提供了串行通道1和2的中断服务程序。因为它们可能不适合于所有的应用，人们会希望调整中断服务程序（它缓冲串行输入）和COMCI程序（它读被缓冲的输入）。

2.2.2 视频驱动模块

BIOS提供了有用的视频设备驱动模块。然而，如果需要更高的速度和更强的功能，一些特殊的输出作业就要重写。PAX视频驱动模块仅对第一屏幕（80×25字符的屏幕方式）提供了服务。在第一屏幕里，视频RAM中的两个字节（字符字节和属性字节）对应于CRT屏幕上的一个字符。

对一些视频电路板，当程序不通过同步而迫使字符直接进入视频RAM中时，会出现一种有趣的情况。例如，使用IBM彩色图形板时，一个强迫视频写的程序会在屏幕上造成“雪花”（实际是一些小白点，有时妨碍不大）。产生雪花的原因是，计算机在读或写RAM时视频控制板也同时需要这个RAM。

然而，有些视频板不论程序如何快地写视频RAM也不会产生雪花。PAX视频驱动模块提供了一个“软件开关”来让开发者指定雪花是否可接受。如果把这个“开关”断开，视频板将不会产生雪花。

视频驱动模块是：

表 2.2.2—1 视频驱动模块

程序名	解释
CONCLS	清屏幕
CONCO	单个字符输出程序
CONSO	串输出程序（支持矩形、属性和光标变化）
DRVINI	初始化视频驱动模块（每当初始化系统时请求一次）

驱动模块是可再入的并可在任意时刻被任何作业请求。

2.3 PAX软件工具包

PAX也包括一个“工具包”，这些程序用汇编语言写成，从而占用了最少的存贮和执行时间，它们可应用于广泛的终端用户系统。下面详细介绍。

2.3.1 中断向量工具

在许多PC应用系统中，当一个程序被初始化时必须设置几个中断向量，当程序结束时这些向量必须被恢复为它们的初值。PAX工具包提供了程序INSVEC和RESVEC（INStall VECtor and REStor VECtor）来处理这些繁锁的事情。

安装程序INSVEC从调用程序中取得两个参数，用来指明那个向量被安装以及该向量的新地址，老向量存入专用存贮器中，新向量被装入，也即调用程序负于记住原来的向量。

复原程序RESVEC只需要一个参数。根据这个值原向量代替了新向量。注意，嵌套地使用INSVEC和RESVEC调用（对同一向量）将会出错，从而应该避免。

2.4 PAX 个人计算机初始化作业

PAX 操作系统不例外地也是由初始化作业TASK0生成起来的。TASK0的责任是建立PAX8632所需的工作环境。这一模块的源代码放在文件TASK0.ASM中。

2.4.1 起动PAX的要求

TASK0必须装上实时中断向量从而得到PAX 系统时钟（见 INSVEC，2.3.1）；TASK0也必须起动开发者所希望的应用程序（使用系统原语SYSSTR，2.1.2.3）。

此外，TASK0还必须装上其它中断向量并完成其它的不被以后作业所承担的初始化工作。例如，如果一个系统需要中断驱动的串行I/O，那么TASK0应该装上对应的中断向量，并请求串行初始化例程，当然一个应用作业能够完成TASK0中的这些起动事项从而使得系统退出更容易实现（见下面2.4.2）。

2.4.2 通过TASK0退出PAX

退出子程序SYSEXT包含在模块TASK0中。SYSEXT是公共终止程序，当任何应用程序调用它时，它系统地复原中断向量，终止作业，把控制权交给DOS命令级。

一般说来，在起动时被装入的所有中断向量都必须恢复到它们原来的状态以保证正确的PC操作。

三、建立基于PAX的系统

以下几段描述了有效地利用PAX系统原语的作业的例子，每个算法是用高级伪代码写成的，其后跟着一个作业的描述，因而适合于移植到各种语言上。

3.1 作业示例A——读数字输入

作业A读一个数字输入口并把该口的位模式扩展为一系列的8字节信号，这些信号就能很容易地被其它应用作业读出。作业A每110m执行一次。

1. 用0ms为参数调用SYSSLP去同步系统时钟（为在以后使用 SYSPER，这一步是必要的）；
2. 读一个数字输入口；
3. 把数字输入扩展为一系列8字节信号，（或为00H或为01H），存入全局RAM；
4. 为周期性的请求，以110ms为参数调用SYSPER；
5. 重新执行步骤2。

作业A循环地（永远）读输入口然后扩展，使得读数字输入的工作对其它作业变得更容易，使用周期性调用SYSPER保证了作业A以很规则的速率工作。

3.2 作业示例B——支持一个打印排队

作业B支持一个打印排队。实质上，它将产生一个队列并为各个文件请求分配优先权。

1. 使用SYQCRE产生一个队列，把队列标识符存在全局RAM中使得其它作业也能够访问它；通过全局RAM中设置标识，通知其它作业队列已打开；

2. 利用一秒钟的暂停时间，通过请求SYQPND来确定一个请求是否已被置入队列中；

3. 如果SYQPND出错退出（无就绪信号），重新执行步骤2（换句话说，永远悬挂等待）。

4. 复制双字长讯号数据，该数据是指向存放驱动器、路径和文件说明的局部存贮区的指针；

5. 打开所指定的文件，如果DOS出错发生，重新执行步骤2；

6. 读文件并打印，完成后关闭文件。

7. 重新执行步骤2。

一般说来，其它应用作业将等待一个“打印队列就绪”（步骤1）。然后，每个希望打印一个文件的作业就可以发送一个讯号到打印队列（用全局打印队列标识符使用SYQPST）。

每个讯号仅由一个指向完整的驱动器、路径和文件名的双字长指针组成，如果文件名不合法，作业B就废弃对该讯号的打印操作并等待下一个文件说明。

3.3 作业示例C —— 响应键盘命令

作业C等待键盘输入一数据串，串结束时（回车键0DH），通过队列送给另一个作业。

1. 当前处理串置为空；

2. 使用CONCI（CONsole Character Input）确定是否一个字符已就绪；

3. 如果字符未就绪，使用系统原语SYSSLP睡眠55ms，然后重新执行步骤2；

4. 把收的字符联接在当前处理串末尾；

5. 如果该字符不是回车，从新执行步骤2；

6. 从全局RAM中获取目标队列标识符；

7. 通过系统调用SYQPST和队列ID（标识符）（步骤6）把指向串的一个双字长指针挂到队列上；

8. 如果SYQPST出错返回（也许是队列满并非队列标识符错），使用系统原语SYSSLP睡眠100ms，然后回到步骤6；

9. 回到步骤1。

本例假定了在执行时目标队列标识（步骤6）已正确地放置在全局RAM中。在一个实际应用系统中，去等待一个表明这一事件确已发生了的信号是明智的。

3.4 作业示例D和E——同步化

可以通过一个简单的方式实现两个作业的同步。作业D能向它自己的队列发送一个讯号并悬挂在作业E的队列上，作业E能悬挂在作业D的队列上并立即为自己的队列发讯。

作业D的算法形式如下：

1. 使用系统原语SYQCRE建立一个大小为1的D作业队列（高效地建立一个作业之间的邮箱）；
2. 把D作业标识存入全局RAM中；
3. 向D作业队列发送一个讯号；
4. 等待一个来向E作业队列的讯号（E的队列标识应存入全局RAM中）；
5. 执行并发进程的D作业元；
6. 返回步骤3。

类似地，作业E的算法如下：

1. 使用系统原语SYQCRE建立一个尺寸为1的E作业队列；
2. 把E作业队列的标识存入全局RAM中；
3. 等待一个来自D作业队列的讯号；
4. 向E作业队列发送一个讯号；
5. 执行并发进程的E作业元；
6. 返回步骤3。

用这种方式，一个事件或一系列事件可触发来自应用系统的一个并发响应。

四、系统原语

本节将用例子来详述每个系统原语，提示使用的注意事项。

4.1 SYSCRE——建立一个作业

SYSCRE为PAX提交一个作业。一个作业（例如初始系统的TASK0）调用SYSCRE来通知PAX一个新程序。SYSCRE不会引起作业切换，因此，它可在一个中断服务程序中使用。

输入：寄存器DX：AX包含初始代码段址：偏移量，即作业的指令指针。

输出：如果调用成功，进位标志（CF）将清除，这时，AX将含有唯一标识该作业的作业标识；如果CF被设置，AX寄存器将包含下面的出错码：

1002……作业建立出错

边缘效应：所有的标志位将被破坏；AX将被作业ID或的错误码冲掉。

如下的例子（汇编语言写成）展示了SYSCRE的典型用法。

```
mov  dx, seg NEWTASK          ; 获得新作业的段址
mov  dx, offesf NEWTASK       ; 获得新作业的偏移量
```