第六章 输入输出系统

一、教学目的要求：

1．掌握输入输出系统的基本功能

2．掌握 IO 设备和设备控制器的概念

3．掌握 IO 控制的四种方式

4．掌握缓冲管理

5．了解设备独立性以及设备驱动程序

6．掌握设备分配

二、内容分析：

1．概述：I/O 设备的类型非常多，且彼此间在多方面都有差异，比如它们接收和产生数据的速度，传输方向、粒度、数据的表示形式及可靠性等方面。为了对这些不同的设备进行控制，通常都是为它们配置相应的设备控制器，用户可以通过调用设备的控制器（麻烦剪辑为正确的）的相关命令来控制设备，而不用去关心与设备相关的实现细节。

2．教学重点：

1）IO 控制方式

2）设备独立性

3）缓冲管理

4）设备分配

3．教学难点：

1）IO 控制方式

2）设备独立性

I/O 系统的基本概念

I/O 系统是计算机系统中一个重要的组成部分。一般包括输入、 输出和存储设备及相应的设备控制器，在大、中型机系统中，还有 I/O 通道或 I/O 处理机。

I/O 系统的功能

1、隐藏物理设备的细节

I/O 设备的类型非常多，且彼此间在多方面都有差异，比如它们接收和产生数据的速度，传输方向、粒度、数据的表示形式及可靠性等方面。为了对这些不同的设备进行控制，通常都是为它们配置相应的设备控制器，用户可以通过调用设备的控制器的相关命令来控制设备，而不用去关心与设备相关的实现细节。

2、与设备的无关性

隐藏物理设备的细节，在早期的 OS 中就已实现，它可方便用户对设备的使用。与设备的无关性是在较晚时才实现的，这是在隐藏物理设备细节的基础上实现的。

3、提高处理机和 I/O 设备的利用率

在一般的系统中，许多 I/O 设备间是相互独立的，能够并行操作，在处理机与设备之间也能并行操

作。为此，一方面要求处理机能快速响应用户的 I/O 请求，使 I/O 设备尽快地运行起来；另一方面也应尽量减少在每个 I/O 设备运行时处理机的干预时间。

4、对 I/O 设备进行控制

对 I/O 设备进行控制是驱动程序的功能。目前对 I/O 设备有四种控制方式：① 采用轮询的可编程 I/O 方式；② 采用中断的可编程 I/O 方式；③ 直接存储器访问方式；④ I/O 通道方式。

5、确保对设备的正确共享

从设备的共享属性上，可将系统中的设备分为如下两类：

（1）独占设备，进程应互斥地访问这类设备，即系统一旦把这类设备分配给了某进程后，便由该进程独占，直至用完释放。典型的独占设备有打印机、磁带机等。系统在对独占设备进行分配时，还应考虑到分配的安全性。

（2）共享设备，是指在一段时间内允许多个进程同时访问的设备。典型的共享设备是磁盘，当有多个进程需对磁盘执行读、写操作时，可以交叉进行，不会影响到读、写的正确性。

6、错误处理

大多数的设备都包括了较多的机械和电气部分，运行时容易出现错误和故障。从处理的角度，可将错误分为临时性错误和持久性错误。对于临时性错误，可通过重试操作来纠正，只有在发生了持久性错误时，才需要向上层报告，请求高层软件解决。

I/O 设备的类型

I/O 设备的类型繁多，从 OS 观点看，其重要的性能指标有: 设备使用特性、数据传输 速率、数据的传输单位、设备共享属性等。因而可从不同角度对它们进行分类。

1) 按设备的使用特性分类 按设备的使用特性，可将设备分为两类。第一类是存储设备，也称外存或后备存储器、 辅助存储器，是计算机系统用以存储信息的主要设备。该类设备存取速度较内存慢，但容 量比内存大得多，相对价格也便宜。第二类就是输入/输出设备，又具体可分为输入设备、 输出设备和交互式设备。输入设备用来接收外部信息，如键盘、鼠标、扫描仪、视频摄像、 各类传感器等。输出设备是用于将计算机加工处理后的信息送向外部的设备，如打印机、 绘图仪、显示器、数字视频显示设备、音响输出设备等。

2) 按传输速率分类 按传输速度的高低，可将 I/O 设备分为三类。第一类是低速设备，这是指其传输速率仅 为每秒钟几个字节至数百个字节的一类设备。属于低速设备的典型设备有键盘、鼠标器，语音的输入和输出等设备。第二类是中速设备，这是指其传输速率在每秒钟数千个字节至数十万个字节

的一类设备。典型的中速设备有行式打印机、 激光打印机等。第三类是高速设备，这是指其传输速率在数百个千字节至千兆字节的一类设备。典型的高速设备有磁带机、磁盘机、光盘机等

3）按设备的共享属性分类 这种分类方式可将 I/O 设备分为如下三类：

(1) 独占设备。这是指在一段时间内只允许一个用户(进程)访问的设备，即临界资源。 因而，对多个并发进程而言，应互斥地访问这类设备。系统一旦把这类设备分配给了某进程后，便由该进程独占，直至用完释放。应当注意，独占设备的分配有可能引起进程 死锁。

(2) 共享设备。这是指在一段时间内允许多个进程同时访问的设备。当然，对于每一时 刻而言，该类设备仍然只允许一个进程访问。显然，共享设备必须是可寻址的和可随机访问的设备。典型的共享设备是磁盘。对共享设备不仅可获得良好的设备利用率，而且它也是实现文件系统和数据库系统的物质基础。

(3) 虚拟设备。这是指通过虚拟技术将一台独占设备变换为若干台逻辑设备，供若干个 用户(进程)同时使用。

接下来我们学习设备控制器

设备控制器是计算机中的一个实体，其主要职责是控制一个或多个 I/O 设备，以实现 I/O 设备和计算机之间的数据交换。它是 CPU 与 I/O 设备之间的接口，它接收从 CPU 发来的命令，并去控制 I/O 设备工作，以使处理机从繁杂的设备控制事务中解脱出来。

设备控制器具有以下基本功能

第 1 个功能， 接收和识别命令：就是 CPU 可以向控制器发送多种不同的命令，设备控制器应能接收并识别这些命令。

第 2 个功能， 数据交换，这是指实现 CPU 与控制器之间、控制器与设备之间的数据交换。

第 3 个功能 ，标识和报告设备的状态：控制器应记下设备的状态供 CPU 了解。

第 4 个功能 ，地址识别：就像内存中的每一个单元都有一个地址一样，系统中的每一个设备也都有一个地址， 而设备控制器又必须能够识别它所控制的每个设备的地址。

第 5 个功能 ，数据缓冲区：由于 I/O 设备的速率较低而 CPU 和内存的速率却很高，故在控制器中必须设置一缓冲器。

第 6 个功能，差错控制：设备控制器还兼管对由 I/O 设备传送来的数据进行差错检测

设备控制器的组成

由于设备控制器位于 CPU 与设备之间，它既要与 CPU 通信，又要与设备通信，还应具有按照 CPU 所发来的命令去控制设备工作的功能，因此，现有的大多数控制器都是由以下三部分组成：

第 1 部分， 设备控制器与处理机的接口。

该接口用于实现 CPU 与设备控制器之间的通信。共有三类信号线: 数据线、地址线和 控制线。数据线通常与两类寄存器相连接，第一类是数据寄存器(在控制器中可以有一个或 多个数据寄存器，用于存放从设备送来的数据(输入)或从 CPU 送来的数据(输出))；第二类 是控制/状态寄存器

第 2 部分 设备控制器与设备的接口。

在一个设备控制器上，可以连接一个或多个设备。相应地，在控制器中便有一个或多个设备接口，一个接口连接一台设备。在每个接口中都存在数据、控制和状态三种类型的信号。控制器中的 I/O 逻辑根据处理机发来的地址信号去选择一个设备接口。

第 3 部分 I/O 逻辑。

在设备控制器中的 I/O 逻辑用于实现对设备的控制。它通过一组控制线与处理机交互， 处理机利用该逻辑向控制器发送 I/O 命令； I/O 逻辑对收到的命令进行译码。每当 CPU 要启动一个设备时，一方面将启动命令发送给控制器；另一方面又同时通过地址线把地址发送给控制器，由控制器的 I/O 逻辑对收到的地址进行译码，再根据所译出的命令对所选设备进行控制。

缓冲管理

1、引入缓冲区的主要原因

（1）缓和 CPU 与 I/O 设备间速度不匹配的矛盾。

事实上，凡在数据发送速率与接收速率不同的地方，都可设置缓冲区，以缓和它们之间速率不匹配的矛盾。众所周知，CPU 的运算速率远远高于 I/O 设备的速率，如果没有缓冲区，则在打印数据时，必然会由于打印机的速度跟不上而使 CPU 停下来等待；然而在计算阶段，打印机又空闲无事。显然，如果在打印机或控制器中设置一缓冲区，用于快速暂存程序的输出数据，以后由打印机“慢慢 地”从中取出数据打印，这样，就可提高 CPU 的工作效率。类似地，在输入设备与 CPU 之 间也设置缓冲区，也可使 CPU 的工作效率得以提高。

（2）减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制。

在远程通信系统中， 如果从远方终端发来的数据仅用一位缓冲来接收，如图 1(a)所示，则必须在每收到一位 数据时便中断一次 CPU，这样，对于速率为 9.6 Kb/s 的数据通信来说，就意味着其中断 CPU 的频率也为 9.6 Kb/s，即每 100 μs 就要中断 CPU 一次，而且 CPU 必须在 100 μs 内予以响应，否则缓冲区内的数据将被冲掉。倘若设置一个具有 8 位的缓冲(移位)寄存器，如图 1(b) 所示，则可使 CPU 被中断的频率降低为原来的 1/8；若再设置一个 8 位寄存器，如图 1(c) 所示，则又可把 CPU 对中断的响应时间放宽到 800 μs。

（3）提高 CPU 和 I/O 设备之间的并行性。

缓冲的引入可显著地提高 CPU 和 I/O 设备间的并行操作程度，提高系统的吞吐量和设备的利用率。例如，在 CPU 和打印机之间设置了缓冲区后，便可使 CPU 与打印机并行工作。

（4）解决数据粒度不匹配的问题。

2、单缓冲

在单缓冲情况下，每当用户进程发出一个 I/O 请求时，操作系统便在主存中为之分配一个缓冲区，如图 2 所示。在块设备输入时，假定从磁盘把一块数据输入到缓冲区的时间为 T，操作系统将该缓冲区中的数据传送到用户区的时间为 M，而 CPU 对这一块数据处理(计算)的时间为 C。由于 T 和 C 是可以并行的(见图 2)，当 T>C 时，系统对每一块数据的处理时间为 M+T，反之则为 M+C。

3、双缓冲

为了加快输入和输出速度，提高设备利用率，人们又引入了双缓冲区机制，也称为缓冲对换(Buffer Swapping)。在设备输入时，先将数据送入第一缓冲区，装满后便转向第二缓冲区，此时操作系统可以从第一缓冲区中移出数据，并送入用户进程，接着由 CPU 对数据进行计算，在双缓冲时，系统处理一块数据的时间可以粗略地认为是 Max(C，T)。如果 C<T，可使块设备连续输入；如果 C>T，则可使 CPU 不必等待设备输入。 对于字符设备，若采用行输入方式，则采用双缓冲通常能消除用户的等待时间，即用户在输入完第一行之后，在 CPU 执行第一行中的命令时，用户可继续向第二缓冲区输入下一行数据。如果我们在实现两台机器之间的通信时。为了实现双向数据传输，必须在两台机器中都设置两个缓冲区，一个用作发送缓冲区，另一个用作接收缓冲区

4、循环缓冲

先来看循环缓冲的组成 循环缓冲由以下两个部分组成：

第 1 部分， 多个缓冲区。在循环缓冲中包括多个缓冲区，其每个缓冲区的大小相同。作为输入的多缓冲区可分为三种类型：用于装输入数据的空缓冲区 R、已装满数据的缓冲区 G 以及计算进程正在使用的现行工作缓冲区 C，如图 3 所示。

第 1 部分， 多个指针。作为输入的缓冲区可设置三个指针：用于指示计算进程下一个可用缓冲 区 G 的指针 Nextg、指示输入进程下次可用的空缓冲区 R 的指针 Nexti，以及用于指示计算 进程正在使用的缓冲区 C 的指针 Current。

5、循环缓冲区的使用

计算进程和输入进程可利用下述两个过程来使用循环缓冲区。

第 1 个过程， Getbuf 过程。当计算进程要使用缓冲区中的数据时，可调用 Getbuf 过程。该过程将由指针 Nextg 所指示的缓冲区提供给进程使用，相应地，须把它改为现行工作缓冲区， 并令 Current 指针指向该缓冲区的第一个单元，同时将 Nextg 移向下一个 G 缓冲区。类似地 ， 每当输入进程要使用空缓冲区来装入数据时，也调用 Getbuf 过程，由该过程将指针 Nexti 所指示的缓冲区提供给输入

进程使用，同时将 Nexti 指针移向下一个 R 缓冲区。

第 2 个过程，Releasebuf 过程。当计算进程把 C 缓冲区中的数据提取完毕时，便调用 Releasebuf 过程，将缓冲区 C 释放。此时，把该缓冲区由当前(现行)工作缓冲区 C 改为空缓冲区 R。类似地，当输入进程把缓冲区装满时，也应调用 Releasebuf 过程，将该缓冲区释放，并改为 G 缓冲区。

在使用循环缓冲区时计算和输入进程是并发执行的，因而会引发进程之间的同步问题，相应地，指针 Nexti 和指针 Nextg 将不断地沿着顺时针方向移动，这样就可能出现下述两种情况：

第 1 种情况，Nexti 指针追赶上 Nextg 指针，此时输入进程应阻塞，直到计算进程把某个缓冲区中的数据全部提取完，使之成为空缓冲区 R，并调用 Releasebuf 过程将它释放时，才将输入进程唤醒。这种情况被称为系统受计算限制。

第 2 种情况 Nextg 指针追赶上 Nexti 指针，计算进程只能阻塞，直至输入进程又装满某个缓冲区，并调用 Releasebuf 过程 将它释放时，才去唤醒计算进程。这种情况被称为系统受 I/O 限制。这样我们就解决同步问题。

6、缓冲池

上述的缓冲区仅适用于某特定的 I/O 进程和计算进程，因而它们属于专用缓冲。当系统较大时，将会有许多这样的循环缓冲，这不仅要消耗大量的内存空间，而且其利用率不高。 为了提高缓冲区的利用率，目前广泛流行公用缓冲池(Buffer Pool)，在池中设置了多个可供 若干个进程共享的缓冲区。1. 缓冲池的组成 对于既可用于输入又可用于输出的公用缓冲池，其中至少应含有以下三种类型的缓冲区: ① 空(闲)缓冲区； ② 装满输入数据的缓冲区； ③ 装满输出数据的缓冲区。 为了管理上的方便，可将相同类型的缓冲区链成一个队列，于是可形成以下三个队列: (1) 空缓冲队列 emq。这是由空缓冲区所链成的队列。其队首指针 F(emq)和队尾指针 L(emq)分别指向该队列的首缓冲区和尾缓冲区。 (2) 输入队列 inq。这是由装满输入数据的缓冲区所链成的队列。其队首指针 F(inq)和 队尾指针 L(inq)分别指向该队列的首缓冲区和尾缓冲区。 (3) 输出队列 outq。这是由装满输出数据的缓冲区所链成的队列。其队首指针 F(outq) 和 队尾指针 L(outq)分别指向该队列的首缓冲区和尾缓冲区。 除了上述三个队列外，还应具有四种工作缓冲区：① 用于收容输入数据的工作缓冲区； ② 用于提取输入数据的工作缓冲区；③ 用于收容输出数据的工作缓冲区；④ 用于提取输出数据的工作缓冲区。

为使诸进程能互斥地访问缓冲池队列，可为每一队列设置一个互斥信号量 MS(type)。 此外，为了保证诸进程同步地使用缓冲区，又为每个缓冲队列设置了一个资源信号量 RS(type)，并利用 Getbuf 过程和 Putbuf 过程 来对缓冲池进行操作。

缓冲区有以下四种工作方式

第 1 种方式，收容输入。在输入进程需要输入数据时，便调用 Getbuf(emq)过程，从空缓冲队列 emq 的队首摘下一空缓冲区，把它作为收容输入工作缓冲区 hin。然后，把数据输入其中， 装满后再调用 Putbuf(inq，hin)过程，将该缓冲区挂在输入队列 inq 上。

第 2 种方式， 提取输入。当计算进程需要输入数据时，调用 Getbuf(inq)过程，从输入队列 inq 的队首取得一个缓冲区，作为提取输入工作缓冲区(sin)，计算进程从中提取数据。计算进程用 完该数据后，再调用 Putbuf(emq，sin)过程，将该缓冲区挂到空缓冲队列 emq 上。

第 3 种方式，收容输出。当计算进程需要输出时，调用 Getbuf(emq)过程从空缓冲队列 emq 的队 首取得一个空缓冲区，作为收容输出工作缓冲区 hout。当其中装满输出数据后，又调用 Putbuf(outq，hout)过程，将该缓冲区挂在 outq 末尾。

第 4 种方式，提取输出。由输出进程调用 Getbuf(outq)过程，从输出队列的队首取得一装满输出 数据的缓冲区，作为提取输出工作缓冲区 sout。在数据提取完后，再调用 Putbuf(emq，sout) 过程，将该缓冲区挂在空缓冲队列末尾。

IO 控制方式

随着计算机技术的发展，I/O 控制方式也在不断地发展。在早期的计算机系统中，是采 用程序 I/O 方式；当在系统中引入中断机制后，I/O 方式便发展为中断驱动方式；此后，随 着 DMA 控制器的出现，又使 I/O 方式在传输单位上发生了变化，即从以字节为单位的传输 扩大到以数据块为单位进行转

输，从而大大地改善了块设备的 I/O 性能；而通道的引入，又 使对 I/O 操作的组织和数据的传送都能独立地进行而无需 CPU 干预。应当指出，在 I/O 控 制方式的整个发展过程中，始终贯穿着这样一条宗旨，即尽量减少主机对 I/O 控制的干预， 把主机从繁杂的 I/O 控制事务中解脱出来，以便更多地去完成数据处理任务。

1、程序 I/O 方式

早期的计算机系统中，由于无中断机构，处理机对 I/O 设备的控制采取程序 I/O(Programmed I/O)方式，或称为忙—等待方式，即在处理机向控制器发出一条 I/O 指令启 动输入设备输入数据时，要同时把状态寄存器中的忙/闲标志 busy 置为 1，然后便不断地循 环测试 busy。当 busy=1 时，表示输入机尚未输完一个字(符)，处理机应继续对该标志进行 测试，直至 busy=0，表明输入机已将输入数据送入控制器的数据寄存器中。于是处理机将 数据寄存器中的数据取出，送入内存指定单元中，这样便完成了一个字(符)的 I/O。接着再 去启动读下一个数据，并置 busy=1。

2、中断驱动 I/O 控制方式

中断在操作系统中有着特殊重要的地位，它是多道程序得以实现的基础，没有中断，就不可能实现多道程序，因为进程之间的切换是通过中断来完成的。另一方面，中断也是设备管理的基础，为了提高处理机的利用率和实现 CPU 与 I/O 设备并行执行，也必需有中断的支持。中断处理程序是 I/O 系统中最低的一层，它是整个 I/O 系统的基础

中断处理层的主要工作有：进行进程上下文的切换，对处理中断信号源进行测试，读取设备状态和修改进程状态等。

对于为每一类设备设置一个 I/O 进程的设备处理方式，其中断处理程序的处理过程 分成以下几个步骤。

第 1 步，唤醒被阻塞的驱动(程序)进程

当中断处理程序开始执行时，首先去唤醒处于阻塞状态的驱动(程序)进程。如果是采用 了信号量机制，则可通过执行 signal 操作，将处于阻塞状态的驱动(程序)进程唤醒；在采用 信号机制时，将发送一信号给阻塞进程。

第 2 步，保护被中断进程的 CPU 环境

通常由硬件自动将处理机状态字 PSW 和程序计数器(PC)中的内容，保存在中断保留区 (栈)中，然后把被中断进程的 CPU 现场信息(即包括所有的 CPU 寄存器，如通用寄存器、段 寄存器等内容)都压入中断栈中，因为在中断处理时可能会用到这些寄存器

第 3 步，转入相应的设备处理程序

由处理机对各个中断源进行测试，以确定引起本次中断的 I/O 设备，并发送一应答信号 给发出中断请求的进程，使之消除该中断请求信号，然后将相应的设备中断处理程序的入 口地址装入到程序计数器中，使处理机转向中断处理程序

第 4 步，中断处理

对于不同的设备，有不同的中断处理程序。该程序首先从设备控制器中读出设备状态， 以判别本次中断是正常完成中断，还是异常结束中断。若是前者，中断程序便进行结束处 理；若还有命令，可再向控制器发送新的命令，进行新一轮的数据传送。若是异常结束中断，则根据发生异常的原因做相应的处理。

第 5 步，恢复被中断进程的现场

当中断处理完成以后，便可将保存在中断栈中的被中断进程的现场信息取出，并装入 到相应的寄存器中，其中包括该程序下一次要执行的指令的地址 N+1、处理机状态字 PSW， 以及各通用寄存器和段寄存器的内容。这样，当处理机再执行本程序时，便从 N+1 处开始， 最终返回到被中断的程序。

现代计算机系统中，都毫无例外地引入了中断机构，致使对 I/O 设备的控制，广泛采用中断驱动(Interrupt Driven)方式，即当某进程要启动某个 I/O 设备工作时，便由 CPU 向相应 的设备控制器发出一条 I/O 命令，然后立即返回继续执行原来的任务。设备控制器于是按照 该命令的要求去控制指定 I/O 设备。此时，CPU 与 I/O 设备并行操作。例如，在输入时，当 设备控制器收到 CPU 发来的读命令后，便去控制相应的输入设备读数据。一旦数据进入数据寄存器，控制器便通过控制线向 CPU 发

送一中断信号，由 CPU 检查输入过程中是否出错， 若无错，便向控制器发送取走数据的信号，然后再通过控制器及数据线将数据写入内存指定单元中。

3、直接存储器访问(DMA)I/O 控制方式

虽然中断驱动 I/O 比程序 I/O 方式更有效，但须注意，它仍是以字(节)为单位进行 I/O 的，每当完成一个字(节)的 I/O 时，控制器便要向 CPU 请求一次中断。换言之，采用中断 驱动 I/O 方式时的 CPU 是以字(节)为单位进行干预的。如果将这种方式用于块设备的 I/O， 显然是极其低效的。例如，为了从磁盘中读出 1 KB 的数据块，需要中断 CPU 1K 次。为了 进一步减少 CPU 对 I/O 的干预而引入了直接存储器访问方式，

直接存储器访问(DMA)I/O 控制方式的 特点是：

（1）数据传输的基本单位是数据块，即在 CPU 与 I/O 设备之间，每次传送至少一个数 据块；

（2）所传送的数据是从设备直接送入内存的，或者相反；

（3）仅在传送一个或多个数据块的开始和结束时，才需 CPU 干预，整块数据的传送是 在控制器的控制下完成的。

DMA 控制器由三部分组成

（1）主机与 DMA 控制器的接口

（2）DMA 控制器与块设备的接口

（3）I/O 控制逻辑

DMA 工作过程

以从磁盘读入数据为例，来说明 DMA 方式的工作流程。当 CPU 要从磁盘读入一数据块时，便向磁盘控制器发送一条读命令。该命令被送到其中的命令寄存器(CR)中。同 时，还须发送本次要将数据读入的内存起始目标地址，该地址被送入内存地址寄存器（MAR） 中；本次要读数据的字(节)数则送入数据计数器(DC)中，还须将磁盘中的源地址直接送至 DMA 控制器的 I/O 控制逻辑上。然后，启动 DMA 控制器进行数据传送，以后，CPU 便可 去处理其它任务。此后，整个数据传送过程便由 DMA 控制器进行控制。当 DMA 控制器已 从磁盘中读入一个字(节)的数据并送入数据寄存器(DR)后，再挪用一个存储器周期，将该字 (节)传送到 MAR 所指示的内存单元中。接着便对 MAR 内容加 1，将 DC 内容减 1。若减 1 后 DC 内容不为 0，表示传送未完，便继续传送下一个字(节)；否则，由 DMA 控制器发出 中断请求。

4、I/O 通道简介和 I/O 通道控制方式

I/O 通道是一种特殊的处理机，它具有执行 I/O 指令的能力，并通过执行通道 (I/O)程序来控制 I/O 操作。它具有两个特点：

（1) 其指令类型单一，这是由于通道硬件比较简单，其所 能 执 行的命令主要局限于与 I/O 操作 有关的指令；

（2)通道没有自己的内存，通道所执行的通道程序是放在主机的内存中的，换言之，是通道与 CPU 共享内存

I/O 通道的主要目的是为了建立独立的 I/O 操作，不仅使数据的传送能独立于 CPU， 而且也希望有关对 I/O 操作的组织、 管理及其结束处理尽量独立，以保证 CPU 有更多的时 间去进行数据处理；或者说，其目的是使一些原来由 CPU 处理的 I/O 任务转由通道来承担， 从而把 CPU 从繁杂的 I/O 任务中解脱出来。

I/O 通道有三种类型：分别是：字节多路通道、数组选择通道、数组多路通道

虽然 DMA 方式比起中断方式来已经显著地减少了 CPU 的干预，即已由以字(节)为单 位的干预减少到以数据块为单位的干预，但 CPU 每发出一条 I/O 指令，也只能去读(或写) 一个连续的数据块。而当我们需要一次去读多个数据块且将它们分别传送到不同的内存区 域，或者相反时，则须由 CPU 分别发出多条 I/O 指令及进行多次中断处理才能完成。 I/O 通道方式是 DMA 方式的发展，它可进一步减少 CPU 的干预，即把对一个数据块的 读(或写)为单位的干预减少为对一组数据块的读(或写)及有关的控制和管理为单位的干预。 同时，又可实现 CPU、通道和 I/O 设备三者的并行操作，从而更有效地提高整个系统的资源利用率。例如，当 CPU 要完成一组相关的读(或写)操作及有关控制时，只需向

I/O 通道 发送一条 I/O 指令，以给出其所要执行的通道程序的首址和要访问的 I/O 设备，通道接到该指令后，通过执行通道程序便可完成 CPU 指定的 I/O 任务。

设备独立性概念及实现过程

为了提高 OS 的可适应性和可扩展性，在现代 OS 中都毫无例外地实现了设备独立性 (Device Independence)，也称为设备无关性。其基本含义是: 应用程序独立于具体使用的物 理设备。为了实现设备独立性而引入了逻辑设备和物理设备这两个概念。在应用程序中， 使用逻辑设备名称来请求使用某类设备；而系统在实际执行时，还必须使用物理设备名称。 因此，系统须具有将逻辑设备名称转换为某物理设备名称的功能，这非常类似于存储器管理中所介绍的逻辑地址和物理地址的概念。在应用程序中所使用的是逻辑地址，而系统在分配和使用内存时，必须使用物理地址。

设备独立性具有以下两个优点：

（1）设备分配时的灵活性 当应用程序(进程)以物理设备名称来请求使用指定的某台设备时，如果该设备已经分配 给其他进程或正在检修，而此时尽管还有几台其它的相同设备正在空闲，该进程却仍阻塞。但若进程能以逻辑设备名称来请求某类设备时，系统可立即将该类设备中的任一台分配给 进程，仅当所有此类设备已全部分配完毕时，进程才会阻塞。

（2）易于实现 I/O 重定向 所谓 I/O 重定向，是指用于 I/O 操作的设备可以更换(即重定向)，而不必改变应用程序。 例如，我们在调试一个应用程序时，可将程序的所有输出送往屏幕显示；而在程序调试完后，如需正式将程序的运行结果打印出来，此时便须将 I/O 重定向的数据结构——逻辑设备表中的显示终端改为打印机，而不必修改应用程序。I/O 重定向功能具有很大的实用价值， 现已被广泛地引入到各类 OS 中。

为了实现设备独立性，必须再在驱动 程序之上设置一层软件，称为设备独立性软件。

设备独立性软件的主要功能可分为以下两个方面：

第 1 方面， 执行所有设备的公有操作。这些公有操作包括:

① 对独立设备的分配与回收；

② 将逻辑设备名映射为物理设备名，进一步可以找到相应物理设备的驱动程序；

③ 对设备进行保护，禁止用户直接访问设备；

④ 缓冲管理，即对字符设备和块设备的缓冲区进行有效的管理，以提高 I/O 的效率；

⑤ 差错控制，由于在 I/O 操作中的绝大多数错误都与设备无关，故主要由设备驱动程序 处理，而设备独立性软件只处理那些设备驱动程序无法处理的错误；

⑥ 提供独立于设备的逻辑块，不同类型的设备信息交换单位是不同的，读取和传输 速率也各不相同，如字符型设备以单个字符为单位，块设备是以一个数据块为单位，即使 同一类型的设备，其信息交换单位大小也是有差异的，如不同磁盘由于扇区大小的不同， 可能造成数据块大小的不一致，因此设备独立性软件应负责隐藏这些差异，对逻辑设备使 用并向高层软件提供大小统一的逻辑数据块。

第 2 方面，向用户层(或文件层)软件提供统一接口。无论何种设备，它们向用户所提供的接口 应该是相同的。例如，对各种设备的读操作，在应用程序中都使用 read；而对各种设备的 写操作，也都使用 write。

那么设备独立性是如何实现的呢？它主要是通过逻辑设备名到物理设备名的映射来实现

为了实现设备的独立性，系统必须设置一张逻辑设备表(LUT，Logical Unit Table)，用 于将应用程序中所使用的逻辑设备名映射为物理设备名。在该表的每个表目中包含了三项： 逻辑设备名、物理设备名和设备驱动程序的入口地址。当进程用逻辑设备名请求分配 I/O 设备时，系统为它分配相应的物理设备，并在 LUT 上建立一个表目，填上应用程序中使用的逻辑设备名和系统分配的物理设备名，以及该设备驱动程序的入口地址。当以后进程再利用该逻辑设备名请求 I/O 操作时，系统通过查找 LUT，便可找到物理设备和驱动程序。

关于 LUT 的设置，可以是整个系统中只设置一张 LUT，也可以为每个用户设置一张 LUT。

设备驱动程序

设备驱动程序通常又称为设备处理程序，它是 I/O 进程与设备控制器之间的通信程序， 又由于它常以进程的形式存在，故以后就简称之为设备驱动进程。其主要任务是接收上层 软件发来的抽象 I/O 要求，如 read 或 write 命令，在把它转换为具体要求后，发送给设备控 制器，启动设备去执行；此外，它也将由设备控制器发来的信号传送给上层软件。由于驱动程序与硬件密切相关，故应为每一类设备配置一种驱动程序。

为了实现 I/O 进程与设备控制器之间的通信，设备驱动程序应具有以下功能：

（1）接收由设备独立性软件发来的命令和参数，并将命令中的抽象要求转换为具体要求，例如，将磁盘块号转换为磁盘的盘面、磁道号及扇区号。

（2）检查用户 I/O 请求的合法性，了解 I/O 设备的状态，传递有关参数，设置设备的工作方式。

（3）发出 I/O 命令。如果设备空闲，便立即启动 I/O 设备去完成指定的 I/O 操作；如果设备处于忙碌状态，则将请求者的请求块挂在设备队列上等待。

（4）及时响应由控制器或通道发来的中断请求，并根据其中断类型调用相应的中断处理程序进行处理。

（5）对于设置有通道的计算机系统，驱动程序还应能够根据用户的 I/O 请求，自动地构成通道程序

为了实现以上功能在不同的操作系统中所采用的设备处理方式并不完全相同。根据在设备处理时是否设置进程，以及设置什么样的进程而把设备处理方式分成以下三类：

（1）为每一类设备设置一个进程，专门用于执行这类设备的 I/O 操作。比如，为所有的 交互式终端设置一个交互式终端进程；又如，为同一类型的打印机设置一个打印进程。

（2）在整个系统中设置一个 I/O 进程，专门用于执行系统中所有各类设备的 I/O 操作。 也可以设置一个输入进程和一个输出进程，分别处理系统中所有各类设备的输入或输出 操作。

（3）不设置专门的设备处理进程，而只为各类设备设置相应的设备处理程序(模块)，供 用户进程或系统进程调用

通过以上的学习我们可以得到设备驱动程序有如下特点：

（1）驱动程序主要是指在请求 I/O 的进程与设备控制器之间的一个通信和转换程序。它 将进程的 I/O 请求经过转换后，传送给控制器；又把控制器中所记录的设备状态和 I/O 操作 完成情况及时地反映给请求 I/O 的进程。

（2）驱动程序与设备控制器和 I/O 设备的硬件特性紧密相关，因而对不同类型的设备应 配置不同的驱动程序。例如，可以为相同的多个终端设置一个终端驱动程序，但有时即使 是同一类型的设备，由于其生产厂家不同，它们也可能并不完全兼容，此时也须为它们配 置不同的驱动程序。

（3）驱动程序与 I/O 设备所采用的 I/O 控制方式紧密相关。常用的 I/O 控制方式是中断 驱动和 DMA 方式，这两种方式的驱动程序明显不同，因 为后者应按数组方式启动设备及进 行中断处理。

（4）由于驱动程序与硬件紧密相关，因而其中的一部分必须用汇编语言书写。目前有很 多驱动程序的基本部分，已经固化在 ROM 中。

（5）驱动程序应允许可重入。一个正在运行的驱动程序常会在一次调用完成前被再次调用。例如，网络驱动程序正在处理一个到来的数据包时，另一个数据包可能到达。

（6）驱动程序不允许系统调用。但是为了满足其与内核其它部分的交互，可以允许对某 些内核过程的调用，如通过调用内核过程来分配和释放内存页面作为缓冲区，以及调用其 它过程来管理 MMU 定时器、DMA 控制器、中断控制器等。

设备驱动程序的主要任务是启动指定设备。但在启动之前，还必须完成必要的准备工 作，如检测设备状态是否为“忙”等。在完成所有的准备工作后，才最后向设备控制器发 送一条启动命令。 以下是设备驱动程序的处理过程。

第 1 步，将抽象要求转换为具体要求 通常在每个设备控制器中都含有若干个寄存器，分别用于暂存命令、数据和参数等。 由于用户及上层软件对设备控制器的具体情况毫无了解，因而只能向它发出抽象的要求(命令)，但这些命令无法传送给设备控制器。因此，就需要将这些抽象要求转换为具体要求。例如，将抽象要求中的盘块号转换为磁盘的盘面、 磁道号及扇区。这一转换工作只能由驱动程序来完

成，因为 在 OS 中只有驱动程序才同时了解抽象要求和设备控制器中的寄存器情况；也只有它才知道命令、 数据和参数应分别送往哪个寄存器。

第 2 步，检查 I/O 请求的合法性 对于任何输入设备，都是只能完成一组特定的功能，若该设备不支持这次的 I/O 请求， 则认为这次 I/O 请求非法。例如，用户试图请求从打印机输入数据，显然系统应予以拒绝。 此外，还有些设备如磁盘和终端，它们虽然都是既可读又可写的，但若在打开这些设备时规定的是读，则用户的写请求必然被拒绝。

第 3 步，读出和检查设备的状态 在启动某个设备进行 I/O 操作时，其前提条件应是该设备正处于空闲状态。因此在启动设备之前，要从设备控制器的状态寄存器中，读出设备的状态。例如，为了向某设备写入数据，此前应先检查该设备是否处于接收就绪状态，仅当它处于接收就绪状态时，才能启动其设备控制器，否则只能等待。

第 4 步， 传送必要的参数 对于许多设备，特别是块设备，除必须向其控制器发出启动命令外，还需传送必要的参数。例如在启动磁盘进行读/写之前，应先将本次要传送的字节数和数据应到达的主存始址，送入控制器的相应寄存器中。

第 5 步，工作方式的设置 有些设备可具有多种工作方式，典型情况是利用 RS-232 接口进行异步通信。在启动该接口之前，应先按通信规程设定参数：波特率、奇偶校验方式、停止位数目及数据字节长度等。

第 6 步，启动 I/O 设备。

设备分配

1、设备分配中的数据结构

在多道程序环境下，系统中的设备供所有进程共享。为防止诸进程对系统资源的无序 竞争，特规定系统设备不允许用户自行使用，必须由系统统一分配。每当进程向系统提出 I/O 请求时，只要是可能和安全的，设备分配程序便按照一定的策略，把设备分配给请求用 户(进程)。在有的系统中，为了确保在 CPU 与设备之间能进行通信，还应分配相应的控制 器和通道。为了实现设备分配，必须在系统中设置相应的数据结构。

在进行设备分配时所需的数据结构 (表格)有：设备控制表、控制器控制表、通道控制表和系统设备表等。

设备控制表(DCT)，设备控制表主要由以下几个字段组成：

（1）设备类型的字段 type：表明设备的类型

（2）设备标识字段 deviceid：设备在系统中的统一编号

（3）设备队列队首指针。凡因请求本设备而未得到满足的进程，其 PCB 都应按照一定 的策略排成一个队列，称该队列为设备请求队列或简称设备队列。其队首指针指向队首 PCB。在有的系统中还设置了队尾指针。

（4）设备状态。当设备自身正处于使用状态时，应将设备的忙/闲标志置“1”。若与该 设备相连接的控制器或通道正忙，也不能启动该设备，此时则应将设备的等待标志置“1”。

（5）与设备连接的控制器表指针。该指针指向该设备所连接的控制器的控制表。在设备 到主机之间具有多条通路的情况下，一个设备将与多个控制器相连接。此时，在 DCT 中还 应设置多个控制器表指针。

（6）重复执行次数。由于外部设备在传送数据时，较易发生数据传送错误，因而在许多 系统中，如果发生传送错误，并不立即认为传送失败，而是令它重新传送，并由系统规定 设备在工作中发生错误时应重复执行的次数

控制器控制表(COCT)主要由以下几个字段组成：

控制器标识符：controllerid 、控制器状态：忙/闲、与控制器连接的通道表指针、控制器队列的队首指针、控制器队列的队尾指针

通道控制表(CHCT)主要由以下几个字段组成：

通道标识符：channelid 、通道状态：忙/闲、与通道连接的控制器表首址、通道队列的队首指针、

通道队列的队尾指针

2、设备分配时应考虑的因素

为了使系统有条不紊地工作，系统在分配设备时，应考虑这样几个因素

第 1 个因素，设备的固有属性

设备的固有属性可分成三种: 对它们应采取不同的分配策略。

（1）独占设备的分配策略。即将一个设备分配给某进程后， 便由该进程独占，直至该进程完成或释放该设备，然后，系统才能再将该设备分配给其他 进程使用。这种分配策略的缺点是，设备得不到充分利用，而且还可能引起死锁。

（2）共享设备的分配策略。对于共享设备，可同时分配给多个进程使用，此时须注意对这些进程访 问该设备的先后次序进行合理的调度。

（3） 可虚拟设备的分配策略。由于可虚拟设备是指一台物理设备在采用虚拟技术后，可变成多台逻 辑上的所谓虚拟设备，因而说，一台可虚拟设备是可共享的设备，可以将它同时分配给多 个进程使用，并对这些访问该(物理)设备的先后次序进行控制。

第 2 个因素，设备分配算法

对设备进行分配的算法，与进程调度的算法有些相似之处，但前者相对简单，通常只 采用以下两种分配算法：

（1）先来先服务。当有多个进程对同一设备提出 I/O 请求时，该算法是根据诸进程对某 设备提出请求的先后次序，将这些进程排成一个设备请求队列，设备分配程序总是把设备 首先分配给队首进程。

（2）优先级高者优先。在进程调度中的这种策略，是优先权高的进程优先获得处理机。 如果对这种高优先权进程所提出的 I/O 请求也赋予高优先权，显然有助于这种进程尽快完 成。在利用该算法形成设备队列时，将优先权高的进程排在设备队列前面，而对于优先级 相同的 I/O 请求，则按先来先服务原则排队。

第 3 个因素，设备分配中的安全性

（1）安全分配方式 在这种分配方式中，每当进程发出 I/O 请求后，便进入阻塞状态，直到其 I/O 操作完成 时才被唤醒。

（2）不安全分配方式 在这种分配方式中，进 程 在 发 出 I/O 请求后仍继续运行，需要时又发出第二个 I/O 请求、第三个 I/O 请求等。仅当进程所请求的设备已被另一进程占用时，请求进程才进入阻塞状态。 这种分配方式的优点是，一个进程可同时操作多个设备，使进程推进迅速。其缺点是分配 不安全，因为它可能具备“请求和保持”条件，从而可能造成死锁。因此，在设备分配程 序中，还应再增加一个功能，以用于对本次的设备分配是否会发生死锁进行安全性计算， 仅当计算结果说明分配是安全的情况下才进行设备分配

3、独占设备的分配程序

下面通过一个具有 I/O 通道的系统案例，来介绍设备分配过程。当某进程提出 I/O 请求后，系统的设备分配程序可按下述步骤进行设备分配。

第 1 步：分配设备

首先根据 I/O 请求中的物理设备名，查找系统设备表(SDT)，从中找出该设备的 DCT， 再根据 DCT 中的设备状态字段，可知该设备是否正忙。若忙，便将请求 I/O 进程的 PCB 挂在设备队列上；否则，便按照一定的算法来计算本次设备分配的安全性。如果不会导 致系统进入不安全状态，便将设备分配给请求进程；否则，仍将其 PCB 插入设备等待 队列。

第 2 步：分配控制器

在系统把设备分配给请求 I/O 的进程后，再到其 DCT 中找出与该设备连接的控制器的 COCT，从 COCT 的状态字段中可知该控制器是否忙碌。若忙，便将请求 I/O 进程的 PCB 挂在该控制器的等待队列上；否则，便将该控制器分配给进程。

第 3 步：分配通道

在该 COCT 中又可找到与该控制器连接的通道的 CHCT，再根据 CHCT 内的状态信息， 可知该

通道是否忙碌。若忙，便将请求 I/O 的进程挂在该通道的等待队列上；否则，将该通 道分配给进程。只有在设备、 控制器和通道三者都分配成功时，这次的设备分配才算成功。 然后，便可启动该 I/O 设备进行数据传送。

4、SPOOLing 技术

那么什么是 SPOOLing 技术呢？

为了缓和 CPU 的高速性与 I/O 设备低速性间的矛盾而引入了脱机输入、脱机输出技术。 该技术是利用专门的外围控制机，将低速 I/O 设备上的数据传送到高速磁盘上；或者相反。事实上，当系统中引入了多道程序技术后，完全可以利用其中的一道程序，来模拟脱机输入时的外围控制机功能，把低速 I/O 设备上的数据传送到高速磁盘上；再用另一道程序来模 拟脱机输出时外围控制机的功能，把数据从磁盘传送到低速输出设备上。这样，便可在主 机的直接控制下，实现脱机输入、输出功能。此时的外围操作与 CPU 对数据的处理同时进 行，我们把这种在联机情况下实现的同时外围操作称为 SPOOLing，或称为假脱机操作。

由上所述得知，SPOOLing 技术是对脱机输入、输出系统的模拟。因此 SPOOLing 系统主要有以下三部分：

（1）输入井和输出井

这是在磁盘上开辟的两个大存储空间。输入井是模拟脱机输入时 的磁盘设备，用于暂存 I/O 设备输入的数据；输出井是模拟脱机输出时的磁盘，用于暂存用 户程序的输出数据。

（2）输入缓冲区和输出缓冲区

为了缓和 CPU 和磁盘之间速度不匹配的矛盾，在内存 中要开辟两个缓冲区：输入缓冲区和输出缓冲区。输入缓冲区用于暂存由输入设备送来的 数据，以后再传送到输入井。输出缓冲区用于暂存从输出井送来的数据，以后再传送给输 出设备。

（3）输入进程 SPi 和输出进程 SPo

这里利用两个进程来模拟脱机 I/O 时的外围控制机。 其中，进程 SPi 模拟脱机输入时的外围控制机，将用户要求的数据从输入机通过输入缓冲区 再送到输入井，当 CPU 需要输入数据时，直接从输入井读入内存；进程 SPo 模拟脱机输出时的外围控制机，把用户要求输出的数据先从内存送到输出井，待输出设备空闲时，再将输出井中的数据经过输出缓冲区送到输出设备上。

案例：利用 SPOOLing 技术技术实现共享打印机的应用

当用户进程请求打印输出时，SPOOLing 系统同意为它打印输出，但并不真正立即把打印机分配给该用户进程，而 只 为 它 做 两 件 事 ：① 由输出进程在输出井中为之申请一个空闲磁盘块区，并将要打印的数据送入其中；② 输出进程再为用户进程申请一张空白的用户请求打印表，并将用户的打印要求填入其中，再将该表挂到请求打印队列上。如果还有进程要求打印输出，系统仍可接受该请求，也同样为该进程做上述两件事。 如果打印机空闲，输出进程将从请求打印队列的队首取出一张请求打印表，根据表中的要求将要打印的数据，从输出井传送到内存缓冲区，再由打印机进行打印。打印完后，输出进程再查看请求打印队列中是否还有等待打印的请求表。若有，又取出队列中的第一张表，并根据其中的要求进行打印，如此下去，直至请求打印队列为空，输出进程才将自己阻塞起来。仅当下次再有打印请求时，输出进程才被唤醒。

由以上分析可以得到 SPOOLing 系统有如下特点：

（1) 提高了 I/O 的速度

（2）将独占设备改造为共享设备

（3）实现了虚拟设备功能

三、本章总结：

I/O 系统是计算机系统中一个重要的组成部分。在该系统中包括有用于实现信息输入、 输出和存储功能的设备和相应的设备控制器。

为了缓和 CPU 与 I/O 设备速度不匹配的矛盾，提高 CPU 和 I/O 设备的并行性，在现代 操作系统中，几乎所有的 I/O 设备在与处理机交换数据时都用了缓冲区。缓冲管理的主要职 责是组织好这

些缓冲区，并提供获得和释放缓冲区的手段。

程序 I/O 方式效率最低，中断驱动 I/O 控制方式是以字(节)为单位进行 I/O 的，如果将这种方式用于块设备的 I/O，是极其低效的。DMA 方式以数据块为单位的操作。I/O 通道方式是对一组数据块进行操作，又可实现 CPU、通道和 I/O 设备三者的并行操作。

设备独立性 ，也称为设备无关性：就是指应用程序独立于具体使用的物理设备。

设备驱动程序通常又称为设备处理程序，它是 I/O 进程与设备控制器之间的通信程序。

设备分配中的数据结构，设备分配时应考虑的因素 、独占设备的分配程序、SPOOLing 技术。

在多道程序环境下，系统中的设备供所有进程共享。为防止诸进程对系统资源的无序 竞争，特规定系统设备不允许用户自行使用，必须由系统统一分配。为了实现统一分配，系统中设置了相应的数据结构，策略。