第22章 操作系统

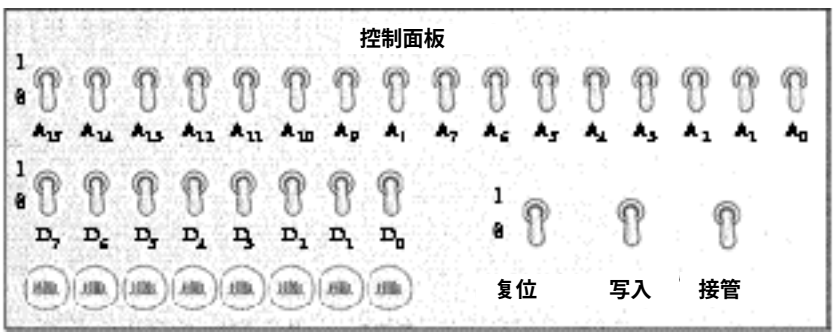
一直以来，我们似乎在组装着——至少在想像中——一台完整的计算机。它有一个微处理器、一些随机访问存储器、一个键盘、一个视频显示器和一个磁盘驱动器。当所有硬件各就各位以后，我们全神贯注于开关，给它加电，带给它生命。也许这样的描述会在你的脑海里唤起Victor Frankenstein 装配怪物时的情景，或者想起Geppetto正在制造将要命名为匹诺槽的木偶。

但我们还缺少一些东西，既不是惊人的力量，也不是良好的愿望。继续进行下去，打开新计算机电源，然后告诉我们你看到了什么？

当阴极射线管发热以后，屏幕上显示的是一些整齐排列但又是随机的 ASCII码字符。这正如预期的那样，当电源断开时，半导体存储器的内容会丢失；当给它加电时，它处在随机的不可预料的状态。同样，为微处理器构建的所有 RAM中的内容也是随机的，微处理器把这些随机的字节当作机器代码来执行。这样不会引起任何坏的情况发生，但是，也没有什么意义。

这里缺少的正是软件。当微处理器加电或复位时，它执行内存中某个地址里存放的机器代码。对8080来说，这个地址是0000h。对正确设计的计算机来说，加电时，该地址处应该有一个机器代码指令（很可能是多个指令中的第一条）。

机器代码指令又是怎样放到内存的那个地方的呢？在新设计的计算机中，把软件放到合适地方的处理过程可能是最令人费解的。要理解它，先从一个控制面板着手。该控制面板与第16章讲到的用来写入字节到随机访问存储器然后再读出的控制面板相似：



与以前的控制面板不同的是，这个控制面板有一个标明为复位的开关，这个开关连到微处理器的复位输入。只要这个开关是闭合的，处理器就什么也不做；当断开这个开关后，微处理器开始执行机器码。

控制面板的使用方法是：复位开关置 ON，复位微处理器，中止执行机器码；接管开关置ON，则接收总线上的地址信号和数据信号。这时，可以使用 A_0～A_{15}开关输入16位的存储器地址。标为D_0～D_7的灯用来显示该地址的8位内容。要写入一个新的字节到相应的地址，则应在D_0～D_7开关上设置该字节，然后把写入开关先拨到 ON再拨到OFF。完成了向内存中写入相应

字节以后，把接管开关设置为OFF，复位开关设置为OFF，则微处理器开始执行程序。

这就是如何向刚刚从头建成的计算机中输入第一个机器码程序的过程，不用说，这是很费事的。

又是什么改变了这一切，使得人们乐于在视频显示器前查看自己程序的执行结果呢？在上一章中已经讲到，只显示字符的视频显示器有1KB的随机访问存储器用来存放25行，每行40个字符的ASCII码。程序把内容写入到该存储器中，方法与写入到计算机中其他存储器中的方法一样。

然而，把程序的输出显示到视频显示器并不是那么简单。例如，如果一段程序，执行结果是4Bh，则不能简单地把这个值写入视频显示器的存储器中。如果这样做，屏幕上将会看到的是字符K，因为该字符对应的ASCII码是4Bh。正确的是应写两个ASCII码字符到显示器：34h（是4的ASCII码）和42h（是B的ASCII码）。8位的计算结果每半个字节是一个十六进制数字，该数字必须通过对应的ASCII码来显示。

当然，也可以写一段小的子程序来完成这种转换。下面的一段8080汇编语言程序用来把十六进制数中的一位转换成对应的ASCII码（假定包含的十六进制数范围从00h～0Fh）：

```
NibbleToAscii: CMP A,0Ah  ;Check if it's a letter or number （判断是数字还是字母）
               JC Number
               ADD A,37h   ;A to F converted to 41h to 46h（把A～F转换成41h～46h）
               RET
Number:        ADD A,30h   ; 0 to 9 converted to 30h to 39h(把0～9转换成30h～39h)
               RET
```

下面的子程序调用NibbleToAscii两次，把累加器A中的一个字节转换成两个ASCII码数字，并放在寄存器B和C中：

```
ByteToAscii:      PUSH   PSW       ;Save accumulator(保存A)
                  RRC              ;Rotate A right 4 times...（A右移4次...）
                  RRC
                  RRC
                  RRC
                                   ;...to get high-order nibble(取高半字节)
                  CALL NibbleToAscii;Convert to ASCII code(转换成ASCII码)
                  MOV  B,A         ;Move result to register B（结果放入寄存器B）
                  POP   PSW        ;Get original A back（取出原来的A）
                  AND  A,0Fh        ;Get low-order nibble（取低半字节）
                  CALL NibbleToAscii   ;Convert to ASCII code（转换成ASCII码）
                  MOV  C,A         ;Move result to register C（结果放入寄存器C）
                  RET
```

这些子程序使得可以在视频显示器中按十六进制来显示一个字节。如果要转换成十进制，再做一些工作即可。此过程与把十六进制数转换成十进制数的方法非常相似——用10来除几次即可。

记住，还没有把这些汇编语言程序输入到内存中。也许，你已经把它们写到了纸上并且转换成了机器码，然后再输入到内存中。这种“手工汇编”是第24章要讲的内容。

尽管控制面板不需要许多硬件，但却不容易使用。它所采用的输入/输出方法是最坏的方法。既然聪明到可以从零开始来制造自己的计算机，却还用数字0和1来作为按键，的确令人

汗颜。那么首先要做的是去掉控制面板。

当然要用键盘来作为按键。前面讲过计算机键盘的构造是只要按下一个键，就会产生一个对微处理器的中断信号。计算机中的中断控制芯片使得微处理器响应中断，执行一条 RST 指令。假设这是一条RST 1指令，这条指令使得微处理器在堆栈中保存当前程序计数器的值并跳转到地址0008h处。从这个地址开始，可以输入一些代码（用控制面板）。这些代码称为键盘处理程序。

为了使一切都正常工作，还需要一些代码在微处理器复位时执行，这些代码叫初始化程序。初始化程序首先设置堆栈指针，使得堆栈分配到内存的有效区域，然后，把视频显示存储器的每一个字节设置为十六进制数 20h，即ASCII码的空格，这样就可以去掉屏幕上的随机字符。初始化程序用 OUT（Output）指令设置光标的位置（光标是视频显示器上的下划线，指示了新输入的字符将要显示的位置）到第 1行第1列。下一条指令为 EI，即中断允许，该指令使得微处理器可以响应键盘中断。在此之后是 HLT指令，它停止微处理器的工作。

这就是初始化程序的工作。从这时起，由于执行了 HLT指令，计算机很可能处于停机状态。能够把计算机从停机状态唤起的事件仅有来自于控制面板的复位信号或从键盘来的中断信号。

无论何时在键盘上按下一个键，中断信号都使得微处理器从初始化程序最后的 HLT语句跳转到键盘处理程序。键盘处理程序用 IN（Input）指令来确定按下的键，然后根据按下的键来执行一些动作（即键盘处理程序处理每一个按键），接着执行一条RET（Return）指令，最后又回到HLT语句等待另一个键盘中断。

不论按下的是字符、数字还是标点符号，键盘处理程序使用键盘扫描码，结合 Shift键是否被按下，来确定合适的 ASCII码。然后将ASCII码写到视频显示存储器中光标的位置。这个过程称为回显键到显示器。光标位置增加并移到刚才显示的字符后面的空格处。由此，可以在键盘上敲入一串字符并显示在屏幕上。

如果按下的键是Backspace（对应的ASCII码是08h），则键盘处理程序删除最后写入到视频显示存储器中的字符，（删除字符是很简单的一件事，只需写入 ASCII码 20h —— 空格字符 —— 到某一内存位置。）然后把光标移回一格。

人们通常在键盘上敲入一行字符（需要改正错误时可用 Backspace键），然后敲入Return(回车)键，回车键在计算机键盘上通常标为 Enter。与在电子打字机上敲 Return键表明已经准备好开始输入下一行一样，在计算机中敲 Enter键表明打字者已经完成了一行文字的键入。

键盘处理程序在处理 Return或Enter键（对应的ASCII码为0Dh）的时候，视频显示存储器的这一行字符被解释成对计算机的一个命令，也就是说，键盘处理程序要去做的一些事情。键盘处理程序中包含有命令处理程序用来解释命令，例如三个命令：W、D和R。

如果字符行以W开始，该命令意味着 Write（写入）一些字节到内存中。假设敲入到屏幕上的行如下面这样：

```
W  1020  35  4F  78  23  9B  AC  67
```

这个命令指示命令处理程序把十六进制数 35、4F等写入到地址 1020h开始的内存中。为了完成这项工作，键盘处理程序需要将 ASCII码转换成字节 —— 前面示范的那个变换的反变换。

如果字符行以D开头，该命令意味着 Display（显示）内存中的一些字节。假使敲入到屏

幕上的行如下面这样：

```
D   1030
```

命令处理程序将会显示从内存地址 1030h开始的存放在内存中的 11个字节（之所以为 11，是因为在40个字符宽的显示器上，在与上面命令同一行的地址后面能显示的字符数为 11）。可以用Display命令来查看内存中的内容。

如果字符行以R开头，该命令意味着Run（运行），如下的命令：

```
R   1000
```

意味着“运行从地址 1000h处开始存储的程序”。命令处理程序把 1000h存到寄存器对 HL中，然后执行指令 PCHL，即把寄存器对 HL的值装入程序计数器，也就是跳转到该地址处执行程序。

采用键盘处理程序和命令处理程序进行工作是一个重要的里程碑。有了它，无需再用什么控制面板，从键盘输入容易、迅速且效果良好。

当然，还有问题。当电源断电时，输入的所有代码会丢失。正因为如此，可能要把这些新代码存到只读存储器，即 ROM中。上一章曾讲到了一个 ROM芯片里存有所有用来在屏幕上显示ASCII字符的点阵模式。假定所用的芯片在制造时已经配置有这些数据，则你也可以在家里自己编程ROM芯片。可编程只读存储器（PROM)芯片只可以编程一次；可擦除可编程只读存储器（EPROM）芯片即可以编程，而且它在紫外光的照射下擦除所有的信息后还可以重新再进行编程。

前面讲过，RAM板连到DIP开关，DIP开关允许设定 RAM板的开始地址。如果使用的是8080系统，初始时一个 RAM板地址应设置成 0000h。如果还有 ROM，则 ROM的地址应为0000h，而RAM板可以连到更高的地址。

命令处理程序的创建是一个重要的里程碑，不仅因为它对输入到内存中的字节提供了较快的解释，而且使计算机现在成为交互式的了。当从键盘上敲入一些东西后，计算机就会做出响应，并在屏幕上显示出来。

一旦有了ROM中的命令处理程序，就可以开始试着从内存中写入数据到磁盘驱动器（可能是对应于磁盘扇区大小的块），并且把数据读回到内存。把程序和数据存放在磁盘上比存放在RAM中要安全得多（后者如果电源出故障它们会丢失），也比存放在ROM中要灵活得多。

也许应该加入一些命令到命令处理程序，如用 S命令来表示存储：

```
S   2080  2  15  3
```

这个命令表示从地址2080h处开始的内存块将要存放到磁盘的第 2面，第15磁道，第3扇区（内存块的大小根据磁盘扇区的大小确定）。同样，也可以加入一个Load命令：

```
L  2080  2  15  3
```

该命令把该扇区的内容从磁盘送回到内存中。

当然，还需要保留存放的地方的记录，可以用手边的本和铅笔来记录。一定要小心不要把保存在某个地址的代码重载到内存的另一个地址，这样做就别指望它能正常工作。所有的Jump和Call指令将会出错，因为它们标识的是原来的地址。同样，如果一个程序比磁盘扇区的大小要大，则需要把它存放到几个扇区。磁盘中有些扇区可能被其他程序或数据占用了，有些扇区还是空的，因而存放长程序的扇区在磁盘上可能是不连续的。

这样，你可能就会发现手工记录哪些东西存放到哪些地方的工作是相当多的，正因为如

此，就需要有一个文件系统。

文件系统是指在磁盘存储器中按文件来组织数据的方法。文件是存放在一个或多个扇区中相关数据的集合。更重要的是，每个文件有一个文件名作为标识，便于记住文件中包含的内容。可以把磁盘看成类似于文件柜，里面的每一个文件都有一个标志用来表示文件的名称。

文件系统通常是称作操作系统的较大软件集合的一部分。本章构造的键盘处理程序和命令处理程序也肯定包含在操作系统中。先不考虑其漫长的演化过程，让我们看一下真正的操作系统是在干什么，又是如何工作的。

回顾历史，最重要的8位微处理器操作系统是CP/M，是Gary Kildall（出生于1942年）在20世纪70年代中期为Intel 8080微处理器而写的，他后来创立了DRI（digital research incorporated）公司。

CP/M存放在磁盘中。早期CP/M最常用的存储介质是单面8英寸磁盘，有77个磁道，每道26个扇区，每扇区128个字节（总共256 256字节），磁盘的头两个磁道包含有CP/M。下面将简单地描述CP/M是如何从磁盘装入到计算机内存中的。

CP/M盘中余下的75个磁道用来存储文件。CP/M的文件系统虽然很简单，但却满足两个基本的要求：首先，磁盘中的每个文件有一个名字作为标识，这个名字也存在磁盘中。其实，CP/M用来读取文件所需的全部信息都与文件一起存放在磁盘中；第二，文件在磁盘中并不占据连续的扇区。由于经常创建和删除不同大小的文件，因而磁盘上的剩余空间都是碎片。文件系统具有把大文件存放在不连续扇区的这种能力是非常有用的。

用来存放文件的75个磁道按分配块进行分组，每一个分配块有8个扇区，即1024字节。磁盘中共有243个分配块，编号从0～242。

开始的两个分配块（共2048字节）用作目录区。目录区是磁盘中的一个特殊区域，用来存放磁盘中每一个文件的名称和一些主要信息。存在磁盘中的每一个文件需要一个32字节长的目录项。因为目录区总共只有2048字节，因而磁盘能够存放2048/32，即64个文件。

每一个32字节的目录项包含有以下信息：

字节	含义
0	通常设为0
1～8	文件名
9～11	文件类型
12	文件扩展
13～14	保留（设置为0）
15	最后一块的扇区数
16～31	磁盘存储表

目录项的第一个字节只在文件系统可供两个或更多人同时共享时使用。在CP/M中，该字节通常设置为0，与第13、14字节一样。

在CP/M中，每个文件的文件名由两部分组成，第一部分称作文件名，最多有8个字符，存放在目录项的第1～8字节；第二部分是文件类型，最多有3个字符，存放在第9～11字节。有几个标准的文件类型，如：TXT表示文本文件（即文件中只包含ASCII码），COM（Command的简称）表示文件内容是8080机器码指令或程序。定义文件时，这两部分由点隔开，如：

```
MYLETTER.TXT
CALC.COM
```

这种文件命令的方式习惯上称为8.3，表明点前最多有8个字符，点后最多有3个字符。

目录项中的磁盘存储表标明了该文件所存放的分配块。假设磁盘存储表的前 4项分别为14h、15h、07h和23h，其余均为0，则表明该文件占用4个分配块，即4KB的空间。文件实际上可能要短一些。目录项的第15字节标明在最后一个分配块中实际用到了多少个128字节的扇区。

磁盘存储表长16字节，可以容纳长达16 384字节的文件，超过16KB的文件要使用多个目录项，称为扩展。在这种情况下，第一个目录项的第12字节设置为0，第二个目录项的第12字节设置为1，依此类推。

上面提到过文本文件也称为ASCII文件，或其他类似名称。文本文件中包含有对应于字符的ASCII码（包括回车和换行代码）供人们浏览。不是文本文件的文件称为二进制文件。CP/M的COM文件为二进制文件，因而它包含8080的机器码。

假设一个文件（一个很小文件）包括三个16位数——例如，5A48h、78BFh和F510h。由这三个数字组成的二进制文件长仅为6字节：

```
48 5A  BF  78  10  F5
```

当然，这是存储多字节数的Intel格式，其中低字节在前。为Motorola处理器编写的程序则是按以下方式来创建文件：

```
5A  48  78  BF  F5  10
```

若用ASCII码文本文件存放这同样3个16位数，则由以下这些字节组成：

```
35 41 34 38 68 0D 0A 37 38 42 46 68 0D 0A 46 35 31 30 68 0D 0A
```

这些字节是数字和字符的ASCII码，每一个数由回车（0Dh）和换行（0Ah）终止。文本文件很容易显示，它们不是作为字节串,而是作为字符显示：

```
5A48h
78BFh
F510h
```

包含这3个数的ASCII码文本文件也可以由以下字节组成：

```
32 33 31 31 32 0D 0A 33 30 39 31 31 0D 0A 36 32 37 33 36 0D 0A
```

这些字节是与这3个数等效的十进制数的ASCII码：

```
23112
30911
62736
```

既然采用文本文件的目的是方便人们阅读，因而没有什么理由不用十进制而非要用十六进制。

上面提到过，CP/M自身存放在磁盘的头两个磁道。为了执行它，CP/M必须从磁盘装载到内存。使用CP/M的计算机中，ROM并不需要很多，它只需要用来存放一小段代码，称为引导程序（因为这段代码通过自举来引导操作系统的其余部分）。引导程序把磁盘最开始的128个字节的扇区装入内存并执行，这个扇区包含有把CP/M的其余部分装入内存的代码。整个这个过程称为引导操作系统。

最终，CP/M把它自己安排在RAM的最高地址区域。装载CP/M以后，整个内存组织如下所示：

该图不是按比例画的。CP/M的三个部件——基本输入/输出系统（BIOS）、基本磁盘操作系统（BDOS）和控制台命令处理程序（CCP）仅占用6KB的内存，临时程序区域（TPA）——在64KB内存的计算机中大约有58KB——初始时没有任何东西。

控制台命令处理程序等效于前面构造的命令处理程序，控制台指的是键盘和显示器。CCP在显示器上显示提示符，就像这样：

```
A>
```

提示符提示可以输入信息。在有不止一个磁盘驱动器的计算机中，A指的是第一个磁盘驱动器，CP/M从该驱动器装入。在提示符后敲入命令并按回车键，CCP就执行该命令并在屏幕上显示结果信息。命令执行完以后，CCP又显示提示符。

CCP只能识别一些命令，最重要的命令可能是：

```
DIR
```

该命令用来显示磁盘目录，即存放在磁盘中的所有文件的列表。可以用特殊字符？和*来限定显示具有某些特定名称和类型的文件，例如：

```
DIR  *.TXT
```

显示所有文本文件，而

```
DIR  A???B.*
```

显示文件名为5个字符，第一个字符为A，最后一个字符为B的所有文件。

另外一个命令是ERA，它是Erase的缩写，用来从磁盘中删除文件。例如：

```
ERA  MYLETTER.TXT
```

删除具有这个名字的文件，而：

```
ERA *.TXT
```

删除所有文本文件。删除文件意味着释放文件的目录项及文件所占用的磁盘空间。

还一个命令是REN，它是Rename的缩写，用来改变文件名。TYPE命令用来显示文本文件的内容。因为文本文件只包含有ASCII码，因而该命令还可用来浏览屏幕上的文件内容，如：

```
TYPE  MYLETTER.TXT
```

SAVE命令用来把临时程序区域中的一个或多个 256字节的内存块以一个特定名称存入到磁盘中。

如果敲入一个CP/M不能识别的命令，就认为输入的是磁盘中的一个程序的名称。程序的文件类型为COM，代表命令。CCP在磁盘中查找叫这个名字的文件，如果有， CP/M把文件从磁盘装入临时程序区域，该区域从地址0100h处开始。以上就是告诉你如何运行磁盘中的文件。如果在CP/M提示符后敲入：

```
CALC
```

且如果名称为CALC.COM的文件存在于磁盘中，则CCP把该文件装入从地址0100h处开始的内存中，然后转到地址0100h处的机器码指令开始执行程序。

前面讲述了如何在内存的任一地方加入机器码指令并执行，但按磁盘文件存储的 CP/M程序必须设计成从内存的特定地址0100h处开始装入。

CP/M包括几个有用的程序，如 PIP（peripheral interchange program），即外设交换程序，用来拷贝文件。 ED是文本编辑器，用来创建和修改文本文件。像 PIP和ED这类小且用来完成简单事务的程序通常称为实用程序。如果运行 CP/M系统，可以购买一些大的应用程序，如字处理软件或计算机电子报表软件；也可以自己编制这样的软件。所有这些也都以 COM类型的文件存储。

到目前为止，已经知道了 CP/M（像许多操作系统一样）如何提供命令和实用程序以便对文件进行基本的操作。同样，也已经知道 CP/M如何把程序装载到内存并执行。作为一个操作系统，CP/M还有第三个主要功能。

在CP/M下运行的程序经常需要把输出写到视频显示器，或者从键盘上读入输入的内容，或者从磁盘读取一个文件和向磁盘中写入一个文件。但通常情况下， CP/M程序并不把程序输出直接写到视频显示存储器中；同样， CP/M程序也不访问键盘硬件看看输入了什么，它也不访问磁盘驱动器硬件去读或写磁盘的扇区。

事实上，运行在 CP/M下的程序利用 CP/M中所构建的子程序集来完成这些公共事务。这些子程序经过特别设计，从而使得程序很容易访问计算机中的硬件 —— 包括视频显示器、键盘和磁盘 —— 且程序设计员不用关心这些外设实际上是怎样进行连接的。更重要的是，在 CP/M下运行的程序不需要了解磁道、扇区，这是 CP/M的工作，它可以把文件存放到磁盘，也可以读取磁盘上的文件。

为程序提供方便访问计算机硬件的手段是操作系统的第三个主要功能。操作系统提供的这种访问手段称之为应用程序接口，即 API（application programming interface）。

在CP/M下运行的程序通过设置寄存器 C为某一特定值（叫作功能值）来使用API并执行指令：

```
CALL  5
```

例如，一个程序通过执行下面的指令获取从键盘上输入的键的ASCII码：

```
MOV  C，01h
CALL  5
```

累加器A中包含有输入的键的 ASCII码。同样

```
MOV  C，02h
```

```
CALL  5
```

把累加器A中的ASCII码字符写到视频显示器中光标的位置，光标移到下一个位置。

如果程序中要创建一个文件，则把寄存器对 DE设置为包含有文件名所在的内存区域的地址，然后执行以下代码：

```
MOV  C，16h
CALL  5
```

此例中，CALL5指令使CP/M在磁盘上创建一个空文件。程序可以利用其他功能向文件写入，最后关闭文件，意味着文件已经使用完毕。该程序和其他程序以后可打开文件并读取文件内容。

CALL5到底能做什么呢？在内存 0005h位置由CP/M设置了一条JMP（Jump）指令，该指令跳转到CP/M基本磁盘操作系统（BDOS）所在的位置。这个区域包含有一些子程序用来完成CP/M的每一项功能。 BDOS正如它的名字一样，基本作用是维护磁盘上的文件系统。通常BDOS必须利用CP/M基本输入/输出系统（BIOS）中的子程序，而 BIOS可实现对像键盘、视频显示器以及磁盘驱动器这样的硬件的访问。实际上， BIOS是CP/M中唯一需要了解计算机硬件的部分。 CCP利用BDOS的功能来实现自己功能，那些 CP/M提供的实用程序也是如此。

API是与设备无关的计算机硬件接口，也就是说在 CP/M下编写的程序不需要知道某一机器上键盘的工作机制、视频显示器的工作机制或读写磁盘扇区的工作机制，它只是简单地利用CP/M的功能来完成涉及到键盘、显示器和磁盘的工作。这样， CP/M程序就可以在不同的计算机上运行，而这些机器可能会用差别很大的硬件来访问外设。（所有 CP/M程序必须运行在8080微处理器上，或能执行 8080指令的处理器上，如： Intel 8085或Zilog的Z－80。）只要计算机运行CP/M，则程序就可以利用 CP/M的功能间接访问硬件。如果没有标准的 API，程序则需要针对不同类型的计算机来做不同的工作。

CP/M曾经是8080中非常流行的操作系统，至今仍具有重要的历史意义。 CP/M对其后的16位操作系统QDOS（quick and dirty operating system）有很大的影响。 QDOS是西雅图计算机产品公司（seattle computer products）的Tim Paterson为Intel的16位8086和8088芯片而编写的。 QDOS后来改名为86-DOS，由Microsoft公司注册。该操作系统被授权给 IBM以MS-DOS这个名称用于第 1代IBM PC机。尽管CP/M的16位版本（称为 CP/M－86）也可用于IBM PC，但MS-DOS很快成了标准。 MS-DOS（在IBM计算机上叫PC-DOS）也允许其他生产 IBM PC兼容机的厂商使用。

MS-DOS没有保留CP/M的文件系统，在 MS-DOS文件系统中使用的是一张叫文件分配表的表，即 FAT。这种技术最初由 Microsoft公司在 1977年采用。磁盘空间分成簇，根据磁盘空间大小，簇的大小也从 512～16 384字节不等。每个文件是簇的集合，文件的目录项只表明了文件开始的簇，FAT能够表明磁盘上每一个簇的下一簇。

MS-DOS磁盘上的目录项长32字节，采用与CP/M一样的8.3文件命名系统，只是术语有些不同：后面的3个字符称作文件扩展名而不是文件类型。 MS-DOS的目录项无需包含分配块的列表，它包含的是这样一些有用的信息，如文件最后修改的日期、时间及文件大小。

MS-DOS的早期版本在结构上很像 CP/M，但MS-DOS中不需要BIOS，因为IBM PC中已经有完整的BIOS存放在了 ROM中。 MS-DOS的命令处理程序是一个名叫 COMMAND.COM的文件。 MS-DOS的运行程序有两种：具有扩展名 COM的文件，大小不能超过 64KB；具有扩展名EXE（可执行）的较大文件。

尽管开始时 MS-DOS 支持 CALL 5 API 功能接口，但对新的程序推荐了新的接口。新的接口利用了8086的一个功能叫作软件中断，这类似于子程序调用，但程序不需要知道它正在调用的确切地址。程序通过执行指令 INT 21h 调用 MS-DOS 的 API 功能。

理论上讲，应用程序只能通过操作系统提供的接口它们来访问计算机的硬件。但对针对20世纪70年代和80年代早期的小型操作系统的应用程序而言，经常绕过操作系统，尤其是在处理视频显示器的时候。直接写入字节到视频存储器的程序比采用其他方式的程序执行速度要快。的确，对有些应用程序——例如，那些需要在显示存储器上显示图形的应用程序——操作系统是不合适的。 MS－DOS最吸引程序员的地方正是它的“反传统性”，程序员可以编写程序以达到硬件的最快速度。

正因为如此，运行在IBM PC上的流行软件常常是根据IBM PC的硬件特点编制的。机器制造商为了与IBM PC竞争也不得不沿袭这些特点。如果不这样做，则会使得这些流行软件不能运行。这些软件通常要求硬件是“ IBM PC或与IBM PC 100%兼容”。

MS-DOS 2.0版于1983年3月发布，它增强了功能来使用硬盘驱动器。虽说当时的硬盘容量很小（按今天的标准），但很快就变得大了起来。当然，硬盘越大就越能存储更多的文件，但磁盘上存储的文件越多，则找到某个文件或组织文件就变得越麻烦。

MS-DOS 2.0的解决方法是采用层次文件系统，它对原有的MS-DOS文件系统做了一些小的改动。前面讲过，磁盘中有一个区域叫目录，它是一个文件列表，里面包含了有关文件存放在磁盘的什么地方的信息。在层次文件系统里，一些这样的文件可能本身就是目录，也就是说，它们是包含其他文件列表的文件，这些文件也有可能还是目录。磁盘中，这个常规的目录称为根目录，包含在其他目录里的目录称为子目录。目录（有时称文件夹）成为对相关文件进行分组的一种方法。

层次文件系统以及MS-DOS 2.0的其他一些功能是从UNIX操作系统借鉴来的。UNIX是20世纪70年代早期在贝尔实验室开发的，大部分工作由 Ken Thompson（生于1943年）和Dennis Ritchie（生于1941年）完成。这个操作系统有趣的名字是一个文字游戏：UNIX先是作为贝尔实验室为MIT和GE开发的名为Multics（表示多路复用信息和计算业务： multiplexed information and computing services）的早期操作系统的一个缺少健壮性的版本。

对设计计算机核心程序的计算机程序员来说， UNIX什么时候都是很好的操作系统。虽然大多数操作系统都是针对特定计算机的，但 UNIX是可移植的，意思是它可以运行在各种各样的计算机中。

在开发UNIX的时候，贝尔实验室还是 AT&T的一个辅助机构。为了抑制 AT&T在电话业的垄断地位， AT&T受到法庭裁决。起初， AT&T被禁止销售UNIX，公司被迫把它授权给别人。所以从1973年开始，UNIX被广泛授权给大学、公司和政府机构。 1983年，AT&T获准重返计算机业并发布了它自己的UNIX版本。

由此导致的结果就是没有单一的 UNIX版本，相反，有许多不同的版本，用不同的名称，运行在不同的计算机上并由不同的经销商销售。许多人把手伸向 UNIX，并在UNIX上留下印迹。然而，当人们在UNIX上加一些东西时，似乎仍然有一种流行的“ UNIX哲学”在引导人们。这个哲学的其中一部分是用文本文件作为公用的文件形式。许多 UNIX实用程序读取文本文件，利用它们来做一些工作，然后写入另外一个文本文件。 UNIX的实用程序可以组织起来形成一个链，然后在这些文本文件上实现不同的处理。

UNIX最初是为只一个人使用时大且昂贵的计算机而编写的。使用 UNIX的计算机通过分时技术允许多个用户同时与计算机交互操作。由于很快地在所有终端之间切换时间片， UNIX操作系统使得用户感觉计算机就像在同时为每个人服务。

并行运行多道程序的操作系统称为多任务操作系统。显然，这种操作系统比像 CP/M和MS-DOS这样的单任务操作系统要复杂得多。多任务使得文件系统复杂化，因为多个用户可能会试图同时访问同一个文件。多任务同样也影响到计算机如何为不同程序分配内存，所以需要进行内存管理。由于多道程序并行运行需要更多的内存，因而很可能计算机没有足够的内存来分配。操作系统可能需要采用虚拟内存技术，当程序不需要某些内存块时可以把它们存放在临时文件中，等需要时再读回内存。

近几年来，UNIX最令人感兴趣的发展是FSF（自由软件基金会，free software foundation）和GUN方案，它们都由 Richard Stallman建立。GUN表示“ GUN不是UNIX ”，当然，GUN不是UNIX。GUN试图与UNIX兼容但却采用了一种方式来使得软件不成为专有的。 GUN方案导致了许多与UNIX兼容的实用程序和工具，还有 Linux，它是一个与UNIX兼容的操作系统的内核。Linux的大部分程序由芬兰的Linus Torvalds完成。近几年，Linux已经变得很流行。

从20世纪80年代中期开始，操作系统最显著的发展趋势是开发大型的、成熟的操作系统，如，苹果公司的Macintosh和微软的Windows,它们结合了图形和可视化视频显示，从而使其更容易使用。本书最后一章将要描述这种趋势。