Microsoft's Disk Operating System

CS-550-1: Operating Systems
Fall 2003
Dominic Swayne

Table of Contents

An Overview:

***History and design environment*.** Microsoft's Disk Operating System (MS-DOS) was one of several operating systems developed in the early 1980s to support the commercial release of 16-bit personal computers. The purpose of this paper is to explore Microsoft's Disk Operating System, describe some of the important aspects of the software, and develop a better understanding of the software.

First known as 86-DOS, it was developed in about 6 weeks by Tim Paterson of Seattle Computer Products (SCP). The OS was designed to operate on the company's own 16-bit personal computers running the Intel 8086 microprocessor. (Paterson, 1983a)

When 86-DOS was developed, Digital Research's Control Program for Microcomputers (CP/M-80) was the operating system most commonly in use for 8-bit personal computers. (Flynn, McHoes, 1996) CP/M-80 was supported by a wide range of application software. Although developed for the new 16-bit environment of the PC, Paterson designed 86-DOS so that it was very similar to the 8-bit CP/M-80. Using translation rules published by Intel, it was relatively easy to port existing application programs to 86-DOS.

International Business Machines (IBM) was working on the hardware design for their entrance to the personal computer market using Intel's 8086 processor. Searching for an operating system to support their new product, IBM sought proposals from two of the leading OS software companies at the time: Digital Research and Softech. (Flynn, McHoes, 1996) Wanting to compete but lacking an operating system of their own, Microsoft worked out a licensing deal with SCP, allowing Microsoft to market 86-DOS under the Microsoft name – paying a royalty fee to SCP. In July of 1981, Microsoft purchased the rights to 86-DOS, renaming it MS-DOS. Later that year IBM released its first PC with MS-DOS (called PC-DOS on the IBM machines) as its primary operating system. (UNESCO, 1999)

Success by Design Paterson designed the operating system with **three specific goals**. First, make it easy to translate existing applications designed for the CP/M-80 operating system and running on the Z80 8-bit processor into applications that would run in 86-DOS on the 8086 16-bit processor. Designing 86-DOS so that existing applications could easily be ported to it was very significant to its commercial viability. However, what ultimately secured the commercial viability of MS-DOS was IBM's release of its first Personal Computer in the fall of 1981 – shipping them with MS-DOS as the primary Original Equipment from the Manufacturer (OEM) operating system. (Paterson, 1983a)

The second design goal is typical of most modern operating systems; speed and efficiency. Paterson designed 86-DOS so that the maximum system memory would be available to applications, minimized disk transfers, improved the speed of disk transfers, and decreased OS compute time. (Paterson, 1983a)

The final design goal was that the OS be written in assembly language. This was a practical matter rather than a philosophical decision. According to Paterson, "The only software development tools available to Seattle Computer at the time were an assembler that ran on the Z80 under CP/M and a monitor/debugger that fit into a 2K-byte EPROM (erasable programmable read-only memory). Both these tools had been developed in house." (Paterson, 1983a)

The first version of DOS used by IBM was written in 4000 lines of assembly-language source code and required 8k of memory. (Stallings, 1995)

From its start as Version 1.0 in 1981, MS-DOS underwent significant revisions until it was ultimately retired as a separate product line in 1994 with the introduction of Windows 95 (though Windows still ran on top of DOS for several more versions of Windows). In 1989, DOS accounted for 61 percent of the PC operating systems shipped worldwide and had an installed base of over 45 million. DOS was running more personal computers than all other operating systems combined. (Duncan *et al.*, 1990)

Advancing the state-of-the-art. Though it was not elegant and didn't employ any revolutionary new concepts, Microsoft's Disk Operating System advanced the adoption of technology in many ways. MS-DOS was the first large-scale commercial operating system that functioned on the new 16-bit Personal Computer hardware (Intel's 8086 processor). It was the first general purpose operating system designed to run on multiple hardware configurations of the personal computer (running Intel's x86 family of processors). Finally, largely due to IBM's endorsement, MS-DOS created an industry standard for general-purpose operating systems. (Flynn, McHoes, 1996)

Limitations. By design, DOS is a simple operating system with backwards compatibility. Therein lies one of its greatest weakness. DOS manages jobs from a single user, sequentially on a single processor. The original version could address a maximum of 1MB of memory, user programs were limited to 640k, and it did not support hard drives. Although its layered design, covered in detail later, permitted DOS to grow beyond these initial limitations, it remained a 16-bit operating system even though 32-bit hardware and the ability to change processor modes on the fly (see EMS page 4) became widely available with the introduction of the Intel 386 processor.

The Structures of MS-DOS

Internal Structure of DOS. Designed in layers, the operating system has three essential components: the BIOS module, the kernel, and the command processor.

The Basic Input/Output System (BIOS) is specific to each hardware system and is provided by the hardware manufacturer. The BIOS contains drivers for the basic hardware: the display, keyboard, line printer, clocks, and boot device, and is normally supplied by the hardware manufacturer. The most primitive components of the BIOS, called the resident portion, is stored in Read Only Memory (ROM) and loaded into main memory when the system is initialized. Other device drivers are also loaded from the CONFIG.SYS file during initialization. The kernel communicates with the device drivers through I/O request packets. These requests are translated into the correct command forms for the various hardware components by the drivers. (UNESCO, 1999)

The DOS kernel is the intermediate layer which provides mandatory operating system services and functions that are hardware independent. These functions include: file and record management, memory management, character device I/O, spawning other programs, and access to the real-time clock. (UNESCO, 1999) Being hardware independent, the MS-DOS kernel is nearly identical on every system, regardless of the manufacturer. The kernel is stored on the disk in the MSDOS.SYS file, which has hidden and system attributes. The kernel provides no default error handling. It traps errors through a vector that must be set by the command processor. (Paterson, 1983a)

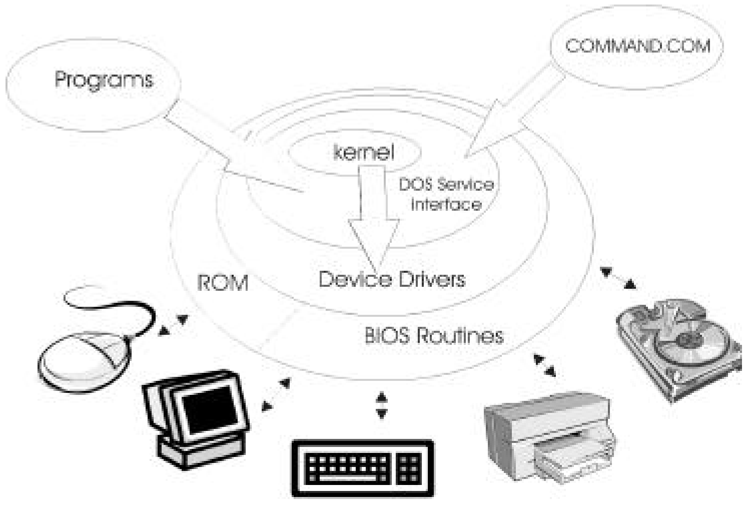

Figure 1. DOS modules. (UNESCO, 1999) Modified version of diagram

The Command Processor provides the MS-DOS user interface. The processor is located in the COMMAND.COM file. It displays the DOS prompt, accepts user input and executes the functions in order. The command processor is also responsible for setting the vector traps and providing error responses. While the kernel is required, proprietary and cannot be modified, the standard command processor is not required. Any user or vendor can implement DOS with their own command processor. A unique feature of the default command processor is that it is split into two pieces known as transient and resident. While OEM MS-DOS's command processor behaves this way, this is not a "*necessary*" feature and is not implement by some vendors choosing to implement their own command processor. (Paterson, 1983a)

As the name implies, the resident portion is the critical section that remains resident in main memory at all times. The resident portion is loaded into the low memory area just above the kernel.

The transient portion is primarily responsible for interpreting user commands. It is loaded into high memory where, if necessary, it can be overlaid by application programs. With newer versions of DOS installed on systems with more memory, this feature is not as important since main memory sizes have increased significantly and later versions of DOS are able to address more main memory. (Paterson, 1983a) One example of commands stored in the transient portion is the format command. Since this command cannot run while a user application is running, there is no useful purpose in keeping this code in resident memory.

Memory management.

The System Memory.

The lowest addressable memory contains a table with 256 interrupt vectors. DOS uses three types of interrupts: internal hardware, external hardware, and software interrupts. An example of an event that would generate an internal hardware interrupt is an attempt to divide by zero. External hardware interrupts are typically generated by peripheral device controllers and math co-processors. Since both of these interrupts types are essentially hard-wired into the system, neither of these may be modified. Software interrupts can be generated by either application programs or the operating system. Interrupts are synchronized by the OS in two steps. First, the OS places the contents of the Program Status Word (PSW), the code segment register, and the instruction pointer register on a stack. Second, it disables the interrupt system until the working interrupt is resolved. The CPU then uses the 8-bit number placed on the system bus by the interrupt device to resume operations and re-activates the interrupt system.

Reserved for BIOS	1MB
Unused	640k
Command Transient	
Program memory	
Command Resident	
DOS Kernel	1k
I/O System	
Interrupt Vectors	0k

Figure 2, DOS Memory Map

The DOS kernel loads itself into the low memory, just above the space occupied by the interrupt vectors and the IO system, starting at 1k. Memory addressing initially used a 16-bit word with direct access from the address register. Starting with the 80286 systems, a segment-offset scheme was used to determine the exact physical location of the data or instructions. While each segment is 64 bytes long, they begin only 16 bytes away from the previous. The DOS **High Memory** is actually a bug in the OS that allows programmers to use the 16 bytes within the 1MB limit to reference a segment of memory above the 1MB limit.

Expanded memory (EMS) implementation was another reason DOS programmers never exploited the protected mode of the 80286 processor. EMS implemented a memory page frame within the 1MB limit, but above the 640kb TPA. Using the page frame, the system could address up to 16MB of memory (on special memory boards). Additional memory management schemes were adopted after the introduction of Microsoft Windows, but they are beyond the scope of this paper.

Discrimination and behavior characteristics by program type.

MS-DOS recognizes three different types of program files, each having different internal file formats: .COM, .EXE, and .BAT.

Beginning with version 2 of MS-DOS, the operating system began supporting dynamic allocation, modification, and release of main memory blocks. The amount of memory allocated depended on several factors.

Significantly, the language used to write an application is a determining factor in the ability of DOS to dynamically allocate memory. If the program is written in C or assembly language, DOS will dynamically allocate memory. Programs written in any other language supported by DOS get their memory allocation at process creation time. For these processes, the type of file and the size of the transient program area (TPA) at the time of process creation determines memory allocation. Files with the .COM extension are pure binary and limited to 64k in size (application and data), but are given the entire Transition Program Area (TPA) regardless of whether they need it or not. (Flynn, McHoes, 1996) Program files with an .EXE extension include a header which specifies the minimum and maximum memo ry required to run the program. If the maximum memory is available, DOS allocates it to the process. If the minimum is not available, the process is not allowed to start. Files with a .BAT extension (batch files) are text files with a list of commands to be executed sequentially by the command processor. Since batch files are actually lists of commands and programs to execute, their limits on memory are a factor of the programs and commands they call.

With the exception of the .COM programs, there is no limit to the number of programs that can be loaded in the TPA. The limit is size, not quantity.

Through version 3.3, the memory management subsystem used the first fit algorithm exclusively and a linked list of memory blocks. Subsequently, the best-fit and last-fit algorithms were added as selectable options for memory management.

A memory block can be as small as 16-bytes or as large as the memory available. The first 5-bytes of each block contain descriptive data about the file and its size. The first byte indicates if this is the last block (90h) or not (77h). Bytes 1 and 2 includes a busy indicator (0 indicates busy) and the pointer to the Program Segment Prefix. The final two bytes indicate the number of paragraphs contained in the block. The code:

```
9000000006h
```

Indicates that this is the last block of memory (90), it is busy (00), its pointer to the PSP is (00), and the block contains 6 paragraphs of memory (each paragraph is 16-bytes).

When the operating system receives a memory request it searches the linked list for a contiguous block of memory that meets the requirements of the requesting process. If an adequate block exists, the process is assigned to that block. An error is generated if the linked list is broken and the system requires a reboot to recover. If two adjacent blocks become free, they are merged together and linked to the list of free blocks. A well designed program will release blocks when they are no longer needed by its processes. A poorly designed program will not release blocks until the program and its processes terminate.

Processor Modes and Memory.

The original IMB PCs were only able to address 1MB of main memory. The original version of MS-DOS was designed with the same limitation, thereby limiting DOS applications to the 640k that DOS makes available. With the introduction of the Intel 286 processor, the system was capable of addressing more than 1MB of main memory. As each new processor was designed, care was taken to ensure it would still be compatible with the original 8086 processor. The backwards compatible 8086 mode is now called the ***real mode***. The real mode subjects the new processor to the original memory MS-DOS size limits, but does take advantage of the increased speed of the processor. (Kozierok, 2001a)

Beginning with the IBM-AT system using Intel's 80286 chip, the ***protected mode*** was introduced and the processor could address 16MB of RAM. Although DOS did not support many of the features available with the new protected mode, the hardware could now address all of the systems memory – eliminating the previous 1MB limit, multi-tasking, virtual memory, and a 32-bit access to memory and device drivers. The key difference in the architecture of the 286 processor and the new protected mode was the address scheme. While the 8086 accessed memory directly from the address register (multiplied by 16 to get the base address), the 286 introduced a descriptor that used four 16-bit words. Using the characteristics of the new descriptor, several capabilities were added: indirection and segment addressing, four privilege levels, and a table indicator to distinguish between local and global memory.

Although protected mode was available with Intel's 80286 processor, changing modes in either direction required rebooting the system. As a result of the difficulty in switching modes, and the desire to retain legacy support, MS-DOS was never enhanced to take advantage of the protected mode so few application software developers did either. With the development of the 80386 and 80486 processors, the system could switch modes without rebooting making the use of the protected mode much more attractive. In addition to faster clock speeds, the memory descriptors used two 32-bit words rather than the four 16-bit words used in the 80286, making them more efficient. However, most software changes came as a result of DOS extenders and Microsoft's introduction of their Windows OS. The memory management and mode switching was handled by the Windows portion of the OS and imposed on DOS.

Conclusion.

Given the environment in which it was developed and the sophistication of the development tools available at the time, MS-DOS was an adequate operating system. Despite its shortcomings, it established a standard that increased the pace and presence of technology in society. The fact that DOS survived from 1981 through 1994 make it the most durable operating system in the history of the personal computer and ensure its notoriety. If there's a lesson to be learned with respect to Microsoft and DOS, it is that it's more important to have money, market-timing, market-share and broad adoption by major manufacturers than it is to write a superior operating system.

Bibliography

Duncan, R., Petzold, C., Baker, S., Schulman, A., Davis, S., Nelson, R., & Moote, R. (1990). *Extending DOS*. Reading, MA: Addison-Wesley Publishing Company, Inc. QA76.76.063E955 1990; ISBN 0-201-55053-9.

Flynn, I. & McHoes, A., (1996). *Understanding Operating Systems*. Boston: PWS PublishingCompany. QA76.76.063F598 1996; 96-27705; 005.4’3—dc20; ISBN 0-534-95093-0.

Kozierok, Charles (2001a). “The PC Guide.” URL: http://www.pcguide.com/ref/cpu/arch/int/modesReal-c.html

Kozierok, Charles (2001b). “The PC Guide.” URL: http://www.pcguide.com/ref/cpu/arch/int/modesProtected-c.html

Paterson, Tim (1983a). “An Inside Look at MS-DOS.” URL: http://www.patersontech.com/Dos/Byte/History.html

Paterson, Tim (1983b). “A Short History of MS-DOS.” URL: http://www.patersontech.com/Dos/Byte/History.html

Paterson, Tim (1981c). “DOS.” URL: http://www.patersontech.com/Dos/Encyclo.htm

Stallings, William (1995). Operating Systems. Englewood Cliff, NJ: Prentic-Hall, Inc. QA76.76.063S733 1995; 94-41175; 005.4’3—dc20; ISBN0-02-415493-8.

United Nations Educational, Scientific and Cultural Organization (UNESCO) (1999). “Disk Operating System.” URL: http://www.netnam.vn/unescocourse/os/22.htm

Wyatt, A., Tiley, E., & Paisley, J. (1996). *Using MS-DOS 6.2, Special Edition*. Indianapolis: Que Corporation. 93-86565; ISBN 1-56529-646-x.