FROM THE MAILBOX

The Origins of DOS

DOS Creator Gives His View of Relationship Between CP/M, MS-DOS

*The following letter comes from Tim Paterson, one of the two engineers who created MS-DOS for Microsoft back in 1981. Mr. Patterson is responding to comments made by author John Wharton in his article on Gary Kildall (see **081003.PDF**). Although this letter is longer than those that we usually print, we hope that you will find it of historical interest.*

Dear Editor:

John Wharton's tribute to Gary Kildall is full of well-deserved praise for Mr. Kildall's contribution to the industry. Unfortunately, Mr. Wharton also got briefly sidetracked onto other issues that he demonstrated he knows nothing about. Through his occasional pot-shots in the past, he has made clear his contempt for MS-DOS and its author (me), and everyone is entitled to an opinion. But to present such distortions—even completely made-up stories—about MS-DOS as facts is irresponsible.

Let's start with the characterization of MS-DOS as "an unauthorized 'quick and dirty' knockoff of CP/M from Seattle Computer Products." Back in 1980, what we now call MS-DOS was sold by Seattle Computer Products (SCP) as 86-DOS with their 8086-based computer system, one of the first to use that microprocessor. At that time, CP/M ran only on the 8080/Z80 microprocessors, although a version for the 8086 was known to be under way. Before starting development of 86-DOS, SCP had been shipping its computers since 1979 and was in desperate need of standard software. The uncertain outlook for CP/M-86 led to the internal development project in April 1980.

SCP was a small company with no clout in the industry. To get major software developers to port their products from the 8080/Z80 to the 8086, I decided we had to make it as easy as possible. I had already written a Z80-to-8086 source code translator (hosted on the 8080 and CP/M). My plan was that running an 8080 CP/M program through the translator would be the only work required by software developers to port the program to the 8086. In other words, the interface used by applications to request operating system services would be exactly the same as CP/M's after applying the translation rules.

So 86-DOS generally had all the same application-visible elements as CP/M—the function codes, the entry point address, part of the File Control Block layout, etc. I used the 1976 *CP/M Interface Guide* for my description of the requirements. I also provided some similar commands from the console—such as DIR, RENAME, ERASE—although any system would have such functions, regardless of name chosen.

But that's where the influence from CP/M ended. (Isn't that enough, you say?) Consider that 86-DOS used a completely different file-storage mechanism than CP/M (representing maybe 80% of the 86-DOS code). Once the functions for translation compatibility were done, I immediately added the "real" file interface—allowing the application to read or write any number of records of any size in a single request, rather than one 128-byte record at a time. I also added rudimentary built-in editing (maybe 15% of the code).

The point is, 86-DOS is completely different from CP/M inside. It is an entirely original work within the confines of providing the translation-compatible interface. Because of the completely different file storage format, none of the internal workings has any corresponding relation to anything within CP/M. I never used CP/M source or disassembly at any time while I was developing 86-DOS. It wouldn't have made sense to; there was nothing I could learn from it, since my tasks were different. And finally, if there had been a CP/M for the 8086 microprocessor, 86-DOS would never have been developed.

Contrast this scenario with some other cases of software "cloning." One example is the ROM BIOS that each IBM-compatible PC must have. The ROM is required to be fully compatible, yet IBM was unwilling to license it to others. Compaq and Phoenix Technologies were two of the first to clone the ROM with independently developed code that performed exactly the same functions and even had the same internal addresses (since some software relied on this). Another example is Digital Research's DR-DOS (now Novell DOS), a perfect functional clone of MS-DOS with improvements added. Unlike 86-DOS, each of these clones performs exactly the same function for the same microprocessor as the original program being cloned. 86-DOS provided wholly new functionality (a completely different, faster storage format) for a microprocessor which had no alternative.

Referring again to Mr. Wharton's article, he states that "these protocols [for memory allocation, file sharing, etc.] were removed [from 86/MS-DOS], since Microsoft programmers didn't understand why they were needed." This is utter nonsense. The fact is, neither 86-DOS nor CP/M 2.2 (the 8080/Z80 version of that day) had any facilities "for memory allocation, file sharing, process switching, and peripheral management." As single-task sys-

tems, they generally had no need for it. Both simply gave all available memory to the single running task—there was no other task to switch to or share files with.

For reasons unrelated to 86-DOS, I left SCP and went to work for Microsoft just in time to put the finishing touches on the adaptation of 86-DOS to the IBM PC (May 1981). I joined Bob O'Rear, who had done all the hard work of getting 86-DOS up and running on the IBM machine. Bob and I must be the programmers that Mr. Wharton refers to, since we were the only ones working on it. I'm pretty sure I didn't hack anything out of my program because I didn't understand it.

Finally, there's the discussion of machine independence. The original CP/M was a bit weak on this point, because it assumed a specific disk format: 77 tracks, 26 sectors, one head, 128 bytes per sector. CP/M 2.2 generalized this into a table-driven approach, although it still assumed 128-byte sectors. The basic problem with CP/M was that it had no internal buffering—the application program was required to read/write files in physical 128-byte sectors. 86-DOS did have internal buffering and separated the logical record of the application from the physical sector on the disk, allowing either to be any size.

To quote from an 86-DOS manual from late 1980: "In order to provide the user with maximum flexibility, the disk and simple device I/O handlers of 86-DOS are a separate subsystem which may be configured for virtually any real hardware." This was put to the test, because the 86-DOS that ran on SCP's S-100 Bus 8086 computer using 8-inch floppy disks and a serial terminal for I/O was exactly the same binary as used on IBM's 8088 with 5-inch disks and memory-mapped video. Other early users of MS-DOS included Zenith and Sirius (later Victor), each of whose computers were unique.

BIOS (for Basic Input/Output System) was the name given to CP/M's hardware dependent layer so that the BDOS (Basic Disk Operating System) and all applications could be hardware independent. But it is not true that "Microsoft lifted the term 'BIOS' for MS-DOS but wrote the software to be machine-dependent anyway." IBM used the term BIOS to refer to ROMs resident in their machine. I used the term I/O System (never abbreviated) to describe the hardware-dependent layer of 86-DOS, and Microsoft even keeps this layer in a separate file on the disk (IO.SYS) from the machine-independent code (MSDOS.SYS).

Responsible journalists check their facts. I don't know if Mr. Wharton is just so gullible that he believes any anti-Microsoft story he hears or if he just makes up the stories himself. To get some of the basics, he could start by reading *Gates* by Manes and Andrews, or *Hard Drive* by Wallace and Erickson. Then, as authors of both books did, just ask me if there are any questions.

—Tim Paterson, Microsoft

Mr. Wharton replies:

I couldn't agree more with Mr. Paterson's observation that responsible journalists check their facts. In the 42 hours available from when I began writing the Kildall piece until I had to deliver a publishable draft to production, I managed to run the column past several of Gary's closest friends and work associates dating back to his consulting days for Intel. They found several errors and omissions; all were corrected before publication. I stand by the piece as printed.

Mr. Paterson is clearly disturbed by my characterization of 86-DOS as an "unauthorized 'quick and dirty' knock-off of CP/M." Yet Mr. Paterson readily admits to having "cloned" the software from a CP/M specification document that was, incidentally, copyrighted and marked "proprietary to Digital Research." His conversion was certainly quick; Mr. Paterson's stated goal was to finish his code before DRI could finish theirs. And it was dirty: 86-DOS supported just 27 of the 37 OS calls implemented by CP/M at the time. In fact, according to the books *Gates* and *Hard Drive,* Mr. Paterson named the first version of his software QDOS, for Quick and Dirty OS.

And Paterson's work was most definitely not authorized. In 1980, the jury was still out concerning the conditions under which it was permissible for one program to appropriate the calling conventions, look, and feel of another. In the end, IBM spent more to head off a copyright-infringement lawsuit from DRI than it spent to acquire the rights for MS-DOS in the first place. I doubt IBM would have done so had it not felt legally exposed. Just last year, Microsoft publicly denounced Sun for promoting a "clone" of the Windows ABI that could run on SPARC workstations.

I can empathize somewhat with the bind in which SCP found itself: unable to sell its 8086 hardware for lack of software and unable to buy the software it wanted. But for Mr. Paterson to cite the unavailability of CP/M-86 as justification for appropriating the "look and feel" of a competing OS and its utilities seems to me to be analogous to telling a judge, "I needed a car, Your Honor, and the plaintiff wouldn't sell me his, so I was forced to take it."

And whereas Mr. Paterson argues that 86-DOS had to be functionally compatible with CP/M to allow CP/M-80 programs to be mechanically ported, in fact it was not. By my count, 86-DOS failed to implement at least nine of the required CP/M 2.2 function calls, altered the functions performed by two others, and "enhanced" the capabilities of several more. According to the book *Undocumented DOS* (the first edition of which was co-authored by Mr. Paterson himself), "Even in the beginning there were crucial differences between the two systems. MS-DOS did not, as widely claimed, mimic every last CP/M function call. For example, MS-DOS did not implement CP/M function 12 (0CH) to get the system version number.

Somewhat unaccountably, MS-DOS instead used (and still uses) function 0CH to read the keyboard." This one change would likely have caused most CP/M 2.2 programs to malfunction, if translated according to the procedure Mr. Paterson describes.

Mr. Paterson is absolutely correct that both 86-DOS and (eight-bit) CP/M 2.2 were single-tasking systems, and neither had facilities for memory allocation, file sharing, etc. But CP/M was the basis for a family of compatible OS products that already supported multitasking (MP/M) and networking (CP/Net) functions. The protocols needed to support these additional features had all been defined by DRI long before 86-DOS was developed. CP/M-86 itself supported memory allocation, 8086 memory segmentation, and multitasking capabilities sufficient for printer spooling. 86-DOS and MS-DOS did not.

As to machine independence—although complete 86-DOS documentation is somewhat hard to come by these days, it's my recollection that both 86-DOS and MS-DOS imposed implicit constraints on memory layouts as well as disk size and configuration, constraints not imposed by CP/M-86.

Mr. Paterson points out that for me to know Microsoft's thought processes, I would have had to talk to him or Bob O'Rear at the time. Well, as it happens, I did. In August of 1981, soon after Microsoft had acquired full rights to 86-DOS, Bill Gates visited Santa Clara in an effort to persuade Intel to abandon a joint development project with DRI and endorse MS-DOS instead. It was I—the Intel applications engineer then responsible for iRMX-86 and other 16-bit operating systems—who was assigned the task of performing a technical evaluation of the 86-DOS software. It was I who first informed Gates that the software he just bought was not, in fact, fully compatible with CP/M 2.2. At the time I had the distinct impression that, until then, he'd thought the entire OS had been cloned. (This visit was recounted briefly in the book *Gates*, by the way, which also lists my name as a source.)

First impressions die hard: at the time, the only documentation that existed for MS-DOS was an "86-DOS Programmer's Reference," on the cover of which the word "Programer" was printed in bold one-inch type—with just one "m." In the weeks that followed, I spoke repeatedly by phone with Tim, Bob, and their cohorts, trying to understand their work. I even made a pilgrimage to Bellevue and spent a day with them in their offices. It was a Monday in September, I believe.

The strong impression I drew 13 years ago was that Microsoft programmers were untrained, undisciplined, and content merely to replicate other people's ideas, and that they did not seem to appreciate the importance of defining operating systems and user interfaces with an eye to the future. In the end it was this latter vision, I feel, that set Gary Kildall so far apart from his peers. ♦

