Tripos—The Roots of AmigaDOS

Metacomco is the British company behind AmigaDOS

BY DICK POUNTAIN

A question that must be puzzling many people in U.S. computer circles is, "What is Metacomco?" When Commodore announced its spectacular Amiga computer, much of the U.S. press failed to point out (and possibly did not know) that the advanced operating system AmigaDOS was in fact written by a small British software house called Metacomco. (For more information on the Amiga, see "The Amiga Personal Computer" by Gregg Williams, Jon Edwards, and Phillip Robinson, August 1985 BYTE, page 83.)

Metacomco is based in Bristol, England, a city that is beginning to rival Cambridge as our potential computing capital (it also houses TDI-Pinnacle, INMOS, and others). Metacomco was founded in 1981 by Derek Budge and Bill Meakin and now employs about 25 people, mainly programmers and other technical staff.

The company's first product was a portable BASIC interpreter written in BCPL, the forerunner of C, which is taught and used extensively at Cambridge University. This interpreter was ported to the 8086 processor and shortly afterward was sold to Digital Research Inc., which still markets its descendant as Personal BASIC. This U.S. link became very important to Metacomco, for the royalties provided a steady source of income during the crucial early years and helped the company establish an office in California, which kept Metacomco in touch with the U.S. computer scene.

In 1983 Dr. Tim King, a Cambridge computer scientist, was engaged by the company as a consultant, and Metacomco's emphasis switched to the 68000 processor, with which King had been working since the first samples came out in 1981. The company produced a series of development tools, also written in BCPL, including a full-screen editor, a macro assembler, and a linking loader. At that time there was no clearly established standard operating system for the 68000, so the next step was to write one. Subsequently, Tripos was born.

The Tripos operating system was based on a multitasking kernel developed as a doctoral thesis project at Cambridge in 1976. ("Tripos" was the name given to the three-legged stools that students sat on in the old days when taking their examinations and has since become the colloquial name for the Cambridge final examinations.) King, then working at Bath University, took the kernel written for a DEC PDP-11 and made it into a full 32-bit multitasking operating system for the Sage microcomputer (which was new at that time). Tripos is BCPL-based in the same way that UNIX is C-based, and it has many innovative features that I will discuss.

Metacomco had also purchased the rights to Cambridge LISP, a powerful LISP interpreter/compiler originally developed for the IBM 370 and then ported to the 68000 at Cambridge. Metacomco produced versions for the ill-fated CP/M 68K and then for Tripos. Reduce 3, a symbolic math system written in LISP, was added to produce a Sage-based workstation that was sold to research institutions in various countries. Customers included SORD in Japan and Bristol neighbor INMOS, who used BCPL for the first stage of bootstrapping its Occam compiler onto the 68000, using Sage computers running Tripos.

In 1984, Tim King joined Metacomco full-time as Research Director, and Sinclair Research launched the QL. Initially, the QL lacked a serious software-development environment, and Metacomco was able to quickly port its development tools, including the BCPL compiler, to it. The company has since extended the range to include an ISO (International Organization for Standardization)-validated Pascal computer, and it markets these products directly, rather than via the manufacturer, largely by mail order.

November 1984 is the crucial date in the AmigaDOS story. Metacomco visited Amiga

(continued)

Dick Pountain is a technical author and software consultant living in London, England. He can be contacted c/o BYTE, POB 372, Hancock, NH 03449.

Corporation (which was still in the midst of finalizing its purchase by Commodore) to discuss the sale of Metacomco's 68000 Pascal compiler for Amiga's new Lorraine machine, as it was then called. During these discussions it was revealed that the Amiga operating system (OS) was way behind schedule and causing some concern. Amiga's stipulations for the Lorraine OS were that it should be multitasking, should support both synchronous and asynchronous I/O, and that the I/O should be stream-based and hardware-independent. Metacomco was already marketing just such an operating system, Tripos, running on the 68000. Amiga agreed to consider Tripos as insurance, in case its already-commissioned system didn't work out.

In February 1985 Metacomco was given the go-ahead, and Tripos was ported to the Lorraine in three weeks flat, thanks to its BCPL portability (although the kernel is written in 68000 code for efficiency). King recalls that when he demonstrated it at the end of February, he turned from the screen to find the whole Amiga staff gathered around applauding; the hardware had suddenly become a real computer. The existing OS was dumped, and the job of turning Tripos into AmigaDOS began.

Fortunately for Metacomco, there was a remarkably close fit between Tripos's internal structure and Amiga's planned software architecture. Tripos is conceptually organized on classic OS lines, with a scheduler, a message-passing system, and a set of device drivers. Amiga's programmers already had ROM (read-only memory) routines to do the jobs of scheduling and message passing and the crucial device drivers for the very special custom chips, the Copper and the Blitter, which handle the graphics, animation, and sound. (For more information on the custom chips, see the interview with Jay Miner entitled "The Amiga's Custom Graphics Chips" conducted by Phillip Robinson, November 1985 BYTE, page 169.) The story might have ended right there had these drivers needed to be written from scratch, given that these were new and unknown custom parts and were probably only partly debugged at the time. The people at Metacomco integrated these parts with the disk-file I/O system, console-text I/O, printer I/O, and command-line processor from Tripos to make AmigaDOS.

The Amiga staff produced the icons/windows front end called Intuition that sits on top of AmigaDOS; we have Metacomco to thank, though, for insisting that an underlying CLI (command-line interface) be always available as a programmers' interface and for more experienced users.

The relationship between Commodore-Amiga and Metacomco has now become quite close. Metacomco's Pascal, LISP, and a much-modified BASIC are all running on the Amiga. The BASIC story is rather complicated in itself. Amiga had already commissioned Microsoft for a version of its much-delayed Macintosh BASIC to be put onto the machine. At the launch in July, however, it was Metacomco's ABASIC that was seen by the press, though certain "ambiguities" may have led people to think it was Microsoft's. At the time of this writing, the language that finally got shipped with the machine still appeared rather vague. [*Editor's note: We have since learned that* ABASIC, *which started out as* Metacomco's, *will become* Microsoft's.] Metacomco is currently working on enhancing ABASIC to permit procedures with parameters, optional line numbers, and full compilation; the present version is structurally still at the Microsoft version 5.2 level. It does, however, have some astonishingly powerful Amiga hardware support commands, such as TRANSLATE and NARRATE, which respectively convert an ASCII string into a phoneme string and then speak it. All the power of the custom chips is accessible through high-level BASIC statements rather than through PEEKs and POKEs.

TRIPOS/AMIGADOS

The Tripos operating system has some features that are not usually found in

(continued)

microcomputer operating systems, particularly in the area of disk-file organization, and these have been inherited by AmigaDOS. Many of these advanced ideas stem from Tripos's origins in a computer science research project; there is much emphasis on doing things the way they should be done, rather than kludging around the way the last guy did it.

Tripos is based on the concepts of multiple tasks and message passing. When an application task is started, it finds a number of other tasks already running. In particular, there will be one for each peripheral device that it needs to talk to, though some of these tasks sleep until awakened by a demand for service from another task. A debugger task also runs continuously in the kernel, which is a great boon to the programmer. An application's environment, greatly simplifed, might look like figure 1.

Every peripheral device is served by its own task. All tasks run concurrently or, strictly speaking, pseudocurrently, since there is only one central processing unit, and the application gets the resources it needs by sending and receiving messages. If a program needs 200 bytes of disk storage, it might send a message to file task 1 requesting this. The file task has its own cache buffer, and it will proceed to get a new block into the cache by in turn communicating with the disk device, which has its own track buffer so that whole tracks are read in at one time. When the file task has the block, it sends a message to the application that the store is now available.

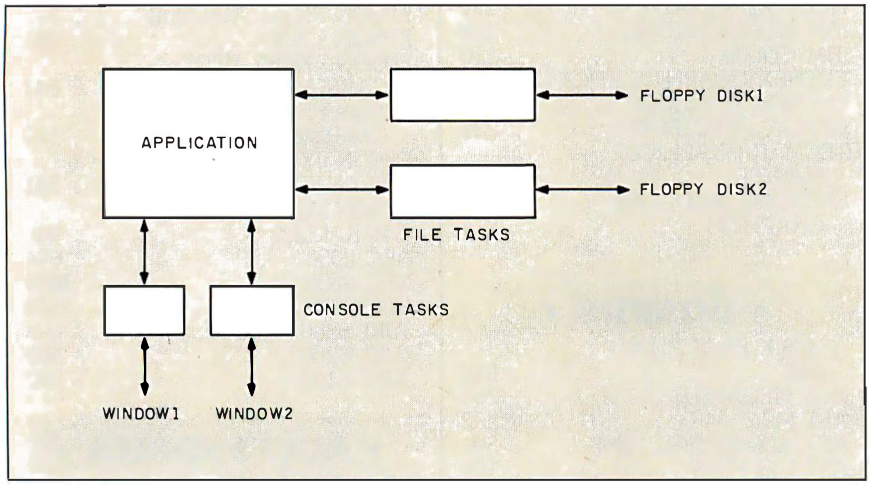

Figure 1: *An example of a simplified application's environment.*

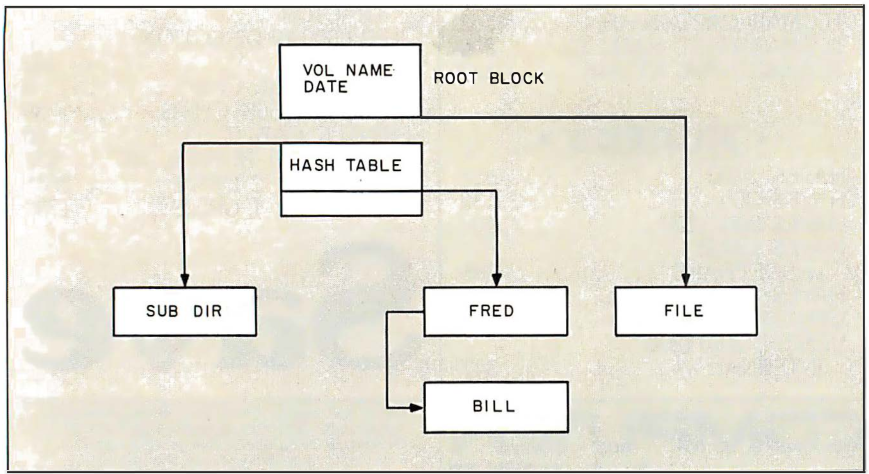

Figure 2: *An example of hash collision. Extension blocks are chained onto the pointed-to block, and the collision is resolved by string matching.*

One consequence of this structure is that, unlike simpler systems such as CP/M and PC-DOS, it's possible for disk activity to occur at seemingly random times, without the user doing anything to provide it; this is quite spooky until you get used to it.

The only limit on the number of tasks that can run is the memory available; it is not a virtual-memory system, but code sharing is used to minimize the memory requirement during multiple invocations of similar tasks. Tasks can be given priorities, and any task can be executed in the background from the command line by typing RUN <taskname>. The CLI is itself a task, and multiple CLIs can be spawned if desired.

The message-passing interface is quite similar to that in UNIX and is identical for all devices and applications; it includes messages like Open, Close, Read, Write, and Seek.

FILE STRUCTURE

It's in the area of disk-file structure that Tripos is truly radical. For starters, there is no directory track on a Tripos disk, and indeed no directory in the usual sense of a table of filenames. Instead, Tripos uses a *root block*, which is placed in the center of the disk surface rather than on track 0 as is usual.

The root block contains the volume name of the disk and the date of creation and last modification. Following this is a *hash table*, via which file or subdirectory names get turned into block numbers. Each block so pointed to can be a directory or a file, leading to a hierarchical directory structure like that in UNIX or PC-DOS 2. In the case of hash collision (perhaps "Fred" and "Bill" both hash to the same block number), extension blocks are chained onto the pointed-to block, and the collision is resolved by string matching (see figure 2).

(continued)

Subdirectories have the same structure as the root block, while file headers have a filename, date, and a table of the data-block numbers that constitute the file. The size of the block is fixed (512K bytes in AmigaDOS, 1024K in Sage Tripos), and when a file header runs out of space for its block table, it merely chains on an extension block.

To optimize speed of data access, file headers and subdirectories are allocated inward from the root block, while data blocks are allocated outward, so that consecutive blocks can be kept close together (see figure 3).

This whole scheme has several beneficial consequences, compared to more conventional operating systems. There are no arbitrary limits on anything; files are governed only by the physical storage capacity of the medium. All files are automatically random-access. Moreover, there is no distinction between binary and ASCII files, as files do not need to contain any special control characters like ^Z for end-of-file. All files are the same, just blocks of "stuff."

There is more, however. All the blocks that make up a file contain pointers to the next block in line (enabling efficient sequential access) and also a *back pointer* to their header block. The inclusion of these features means that even if a file header gets completely mangled, it can be reconstructed by the reading pointers in the data blocks; the individual data know their own identity (see figure 4).

Metacomco also has a "disk doctor" program that can reconstruct a disk, both files and directories, from almost any state of damage short of total data loss, and it can do it automatically. This is a very significant step forward in mass-storage security compared to PC-DOS, where the corruption of a directory track can lead to leaps from high buildings.

The only penalties paid, as tradeoffs for all the advantages, are that directory listing and file renaming are slower than in conventional systems, because there is no single place to go to look for filenames; the whole tree structure needs to be traversed to find the names. Metacomco is currently considering caching the directory structure to alleviate this problem, but from my limited experience of the Amiga, it doesn't seem too bad anyway; it's not much slower than an IBM PC by the time the latter's disk-access and screen-updating speeds have taken their toll.

Given the multitasking nature of Tripos, and hence the unpredictable times of disk accesses, measures were necessary to manage disk changing gracefully. Accordingly, Tripos keeps a bit map of the disk usage in memory—the same idea as a PC-DOS FAT (file-allocation table)—which has a bit set for every block in use. As mentioned before, each file task keeps its own block buffer in memory. After disk activity (signaled by the usual red light) there is a three-second time-out period, after which the task automatically flushes its buffers and the updated bit map to disk. If a disk is removed during the time-out period, the bit map on disk will be marked as invalid, and when that disk is reinserted, a validation task in the kernel will automatically be invoked to rebuild the bit map. Only if the disk is removed when the red light is actually on is there any chance of losing data; Amiga and Metacomco debated long and hard about a mechanical locking scheme similar to that on the Macintosh but decided against it after observing the unpopularity of the latter scheme (with everyone except the paper-clip industry, that is).

Tripos knows all about disk volumes and can find a volume in any drive or prompt for it to be inserted as required; no messing about with default drives or logging on. It is, in fact, possible to remove a disk with a file still open, use a new disk, then be prompted by the system to replace the first disk and continue.

As in UNIX, all devices are addressed as files, with a device name replacing the volume name that would be used in a full-file spec. The device CON: may have window-size parameters attached to it, as in CON:20/20/100/100/Fred, which addresses an 80- by 80-character window called Fred. The serial port and printer can be addressed in a similar way.

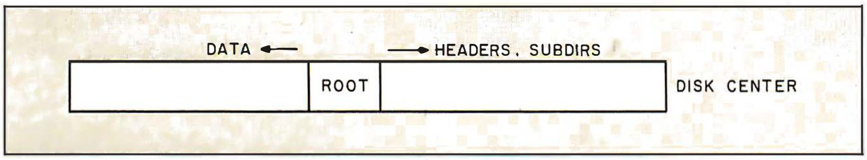

Figure 3: *File headers and subdirectories are allocated inward and data blocks allocated outward to optimize speed of data access.*

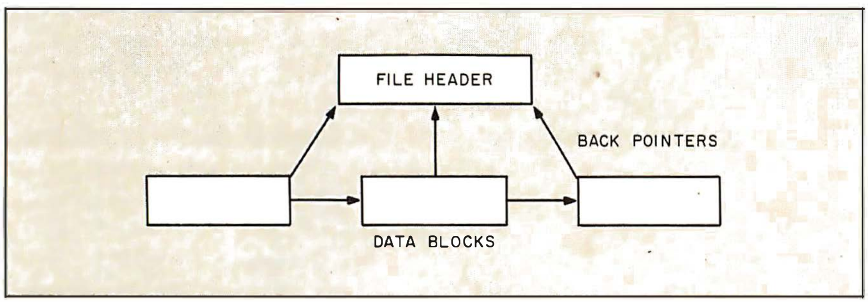

Figure 4: A *file header can be reconstructed by reading pointers in the data blocks.*

Conclusion

The relationship between Metacomco and Commodore-Amiga seems to have been mutually beneficial. The Amiga got itself a mature and capable operating system that was designed on sound, though not conservative, principles. Metacomco, on the other

(continued)

Metacomco maintains its strong links with both Cambridge and Bath universitities.

hand, gained a stronger foothold in the U.S., along with the respect of those in the U.S. computer industry who were already aware of its existence.

Whether or not the Amiga will be the world-shaker that I think it deserves to be must remain the great "wait and see" question of the year. Although some teething problems are emerging, it's likely that they'll be less serious than they might have been had a totally untried OS been adopted.

I've felt for some time that there is insufficient user awareness of just how complex the new-generation, post-Macintosh, operating systems are. The days of "patch it and hope" are gone forever, and we are now deep into the territory of heavyweight software engineering; debugging must now be considered a continuing process, and the chances of a bug-free OS at launch (or even a year later) are pretty remote. Commodore-Amiga is still debating whether or not to commit AmigaDOS to ROM in Macintosh style (first machines are being shipped with a disk-based DOS). Tim King is solidly in favor of keeping a disk-based DOS for precisely these reasons.

As for Metacomco's future plans, it is content for the moment to remain with the 68000, a processor in which the company's accumulated expertise is now paying dividends. The Atari 520ST has attracted Metacomco's attention, and its staff has already put the assembler/editor combination onto it, soon to be followed by Pascal and Lattice C. An IBM PC-based development system, complete with cross-assembler, has just been announced also. The relationship with Lattice arose when it was commissioned to put the compiler onto the Sinclair QL; Metacomco ended up marketing it.

Metacomco still maintains its strong links with both Cambridge and Bath universities (King still teaches a computer science course at Bath) and pays them royalties for work such as the original Tripos kernel. It exemplifies the slow but welcome trend toward fruitful collaboration between academia and commerce that is new to the U.K., although it has been standard practice on U.S. campuses since the beginning of the microelectronics revolution. ■

 Inquiry 151 Inquiry 378