CCPMBYTE.WS4 (= Concurrent CP/M article in BYTE magazine)

- "Concurrent CP/M"
  Joe Guzaitis
  BYTE, November 1983, p.257

(Retyped by Emmanuel ROCHE.)

"By permitting a 16-bit microcomputer to execute several processes that seem to occur simultaneously, this Operating System efficiently uses computer and operator resources"

A growing sentiment at Digital Research can be expressed as:

CCP/M : 16 :: CP/M : 8

that is, Concurrent CP/M is to 16-bit microcomputers as CP/M is to 8-bit machines. Bold stuff. But not really, when you consider that CP/M (Control Program for Microcomputers) has come to dominate the 8-bit market.

But what exactly is concurrency, the major enhancement of this Operating System? Concurrency does not allow two processes to occur at the same time in the same place, but it does permit many processes to occur sequentially in round-robin fashion in infinitesimal time slices, so that they seem to occur simultaneously in the same place. Therefore, although most systems spend a lot of time waiting for input from a person or process, Concurrent CP/M permits a computer to perform a task while waiting for input from another process.

Multitasking, multiprogramming, and concurrency allow as much of a system's resources as possible to perform useful work for as much of its operating time as possible. Concurrency increases throughput, which in turn results in increased efficiency and cost-effectiveness.

16-bit Advantages

Concurrent CP/M has the potential of stimulating the 16-bit microcomputer market the way Visicalc stimulated the early 8-bit field -- by giving the world a powerful example of a microcomputer's capabilities.

Let us face it: 16-bit computers are not inherently faster or more versatile than 8-bit machines. In fact, an 8-bit computer can often run rings around a 16-bit machine. In addition, a wider variety of applications software is available for 8-bit computers than for 16-bit machines. Why spend the extra money for this new technology?

There are two good reasons. The first is memory. Getting an "OUT OF MEMORY" message in the middle of a program is a frustrating experience that nearly every computer user will encounter eventually. But this problem is not insurmountable; there is usually a way to work around memory limitations.

A better reason to choose a 16-bit machine is concurrency. Its large memory requirements make its use within an 8-bit architecture impractical. Concurrent CP/M takes up as much as 90K bytes; 256K bytes are actually needed to make it useful.

How Concurrency Works

To understand how concurrency is possible, we can look at our work habits, which resemble a type of concurrent processing. For example, as I sit here at my word processor typing away, I break momentarily to jot down an appointment on my calendar, go back to typing, break away again to use my calculator, return to the keyboard, stop to look up a word in the dictionary, then go back to typing, all the while waiting for a phone call.

Breaks can be self-generated, such as those made to check a word in the dictionary, or they can be imposed from the outside. We work in an interrupt-driven manner, allowing phone calls, messages, or fellow workers' inquiries to tear us from the task at hand. Many users of Concurrent CP/M say that the operating system seems like a natural extension of the way they work because it enables them to switch among tasks without losing the thread of any of them.

Because it provides the capability for processes to seemingly execute simultaneously, Concurrent CP/M increases processing efficiency much the way online processing proved more efficient than batch processing. In batch processing, similar types of data are accumulated over a period of time and processed in one run. Online processing, on the other hand, allows a computer to appear to handle many sources of input simultaneously, then usually returns to the task's origin. Batch processing works serially; online processing allows another task to begin before the first is completed, and it appears to handle both processes at the same time.

Similarly, single-tasking operating system must process sequentially, and multitasking systems such as Concurrent CP/M rapidly go from one process to another, appearing to perform many tasks at once. And, whereas single-tasking systems left the operator idle much of the time, waiting for a process to be completed, Concurrent CP/M has the machine waiting for the operator, ready to do more work. Concurrent processing involves one user at a time, who feeds various types of input into the processor via several virtual consoles, whereas online processing provides for many users at many consoles, all feeding into a central computer.

How Concurrency Looks to the User

The concept of virtual consoles helps some users understand concurrent processing, but confuses others. The computer can be thought of as having only one actual console (the terminal) but several virtual consoles -- equivalent consoles that can also interact with the central processor. The terminal can monitor one process at a time. A concurrent operating system allows a user to go from one process to another, switching to various virtual consoles to

monitor different processes.

This procedure is analogous to the way a television user can switch from one channel to another, sequentially viewing several programs. Both the television and Concurrent CP/M permit screen switching. Use of a computer differs from that of a television, though, because a computer allows a user to interact with its programs, whereas a television does not (we will ignore those few cable-TV experiments that permit user participation).

Another way to think of concurrency is to picture a computer operator sitting among several computers, each running a different applications program. By swiveling around, the operator can interact with each application -- use the output from one process to inform another, print one letter while writing another, and compile one program while editing another and debugging a third. With Concurrent CP/M, swiveling is replaced by a keystroke, which summons the program you want to monitor to the terminal screen.

Processes and Data Modes in CP/M

In Concurrent CP/M, we talk of processes more than programs. In this environment, a program is a static piece of code, and a process is what is executed. Whenever a program is loaded into memory, a process is created that involves code from the program, the operating system, and housekeeping data that indicates, for example, which virtual console to use. The operating system monitors the process, not the program.

There are two modes in which console output generated by a process can be handled: dynamic and buffered. Whatever task you have selected to be in the foreground directs its output to the console screen, and you monitor the virtual console assigned to that selected process on the terminal. You must set each virtual console to either dynamic or buffered mode, so that the system knows how to handle console output in your absence.

However, a process not being monitored on the screen is considered to be in the background, and its output is not monitored. In dynamic mode, when you select a virtual console, you do not see the procedure as it happened; instead, you see the net results. For instance, if your word processor was performing a search-and-replace procedure in a lengthy file, you would return to see the strings replaced but would have missed the replacements as they occurred.

Output is handled differently in buffered mode. To return to our TV analogy, buffered mode works as though you had a videotape recorder connected to a channel you are not viewing, recording everything that was going on in your absence. When you return to that virtual console, it replays all the updates that happened on that console while you were away in the sequence and context in which they occurred.

Depending on the implementation, information on which mode you are in is usually available on the status line at the bottom of the screen. The status line also typically tells which virtual console is being displayed and the name of the process running, and may also include information such as time of

day, printer assigned to that console, and disk drive in use. As you switch screens, the status line changes, providing information for the next virtual console you want to monitor.

Shared Files

Another feature that Concurrent CP/M provides is a shared file structure. By using BDOS calls, programs can open files in one of three modes: locked, read-only, and unlocked. Two or more concurrent processes can access the same file; that access is controlled by the file-access mode.

The locked mode is the default one. In that mode, a file can be opened only if no other process has that file open already. Once opened in locked mode, the file must be closed before any other process can open, access, or delete it. (An extended lock feature allows a process to keep the file locked after it is closed.)

If a file was opened in read-only mode, no process can write to it, but any process can read from it. But, if a file was opened in unlocked mode, it can be read from or written to by any process.

For a process to access either a read-only or unlocked file, it must open the file in that mode. Record locks are also available in unlocked file mode, to deny access to individual records within an otherwise unlocked file.

Advanced Features

As more software vendors realize the power of concurrency, applications programs will share common data structures that allow the packages to work interactively. Shared files gives us a hint of what is possible. Other features that lend themselves to the interactive environment Concurrent CP/M affords are queue management, and priority setting.

A queue, a line of items waiting for the processor's attention, is a way for one concurrent application to communicate with another. In other words, a process on one virtual console can be made to share data with a process on a different virtual console. Because queues operate entirely in RAM, they work quickly and efficiently. Queues can be created, opened, closed, and deleted just as disk files can, and you can read or write to them on a conditional or unconditional basis. The data structures of the programs must be compatible, however, to allow for queue management.

Another advanced feature that concurrency permits is priority setting. Specifically, it allows you to set a priority level on each process, so that important processes are not hindered by lesser ones. Because a system's processes all share the same central processor, they affect each other's operation. For instance, if your modem is attached to one console and is receiving data, you want to ensure that the data is not slowed down by work you are performing on another console. Moreover, because data integrity and telephone charges are involved, the task receiving the data demands top

priority. Less important tasks can run more slowly.

To ensure that the more crucial task gets preferential handling, you need not use such tactics as postponing "saves" as you work in your word processor, or stopping the compiler while data is being sent or received. The priority-setting capability lets you assign the reception of data priority over other processes. If the modem is using bits-per-second (bps) rates above 1200, other processes may slow down when the modem is receiving or sending data. A lower bps rate, however, should cause no problem.

Priority setting will probably be a standard feature of applications packages designed to run under Concurrent CP/M. Until those packages are available, however, it must be accomplished via a system-function call.

Another advanced capability that is also implemented through a system-function call is process detachment, which allows certain processes that need not be monitored, such as print spooling, to be detached from a virtual console and run unattended, thus freeing a virtual console for other tasks. Concurrent CP/M also provides the program logic for other features that do not actually reside in the operating system. Until they are made available in software packages, though, the only way to get them is to program them yourself. Those packages should also encourage software designers to standardize user interfaces, because when users can rapidly switch back and forth among programs, the differences between software packages can affect operator efficiency.

Additional Benefits

Because printing can take a great deal of time and use little of the processor's power, many people invest in a hardware or software spooler, which allows printing to operate as a background task while another task is carried out in the foreground.

With concurrency, a spooler is unnecessary, because the operating system allows you to print a file from one virtual console while working on several others. Moreover, each virtual console can be assigned to a different printer, so you can print several files, each from a different console, on the same or different printers, while working with other programs. If two files are trying to print a file on the same printer, the first to begin printing "owns" the printer, and the other one must wait until the first is finished. During that time, all activity on the waiting console is suspended.

Communications is another task for wich concurrency will prove useful. Linking many microcomputers in your organization can increase the efficiency of each operator, because it makes available such features as shared files, shared resources, and electronic mail. CP/NET and Concurrent CP/M permit each computer to share files and other resources (such as printers and disk drives) with other computers in the network.

The next level of utility is having several virtual consoles running the same or different programs at the same time. Running the same programs can be of help to writers or reporters, for instance, who may be working on several

articles or stories at the same time. As an idea strikes you for story two while you are in the middle of story one, merely hit a key and type some notes in that story file. To non-writers, this feature may seem unnecessary, but I assure you it is an efficient way to work. Flashes of inspiration are best recorded quickly.

This feature would also be helpful to a financial analyst who might have several spreadsheets running side by side in different currencies, and who might want to use the same base-line data and generates figures in pound, franc, mark, and yen denominations. By switching screens and entering common base-line data, the appropriate currency spreads can be generated instantly.

Theoretical and Realistic Limits

The number of virtual consoles that may someday be supported by a system depends ultimately on the memory available. Let us imagine we manufacture computers. Knowing that 8086/8088 systems provide as much as 1 megabyte of memory and that Concurrent CP/M can use as much as 90K bytes (supporting four virtual consoles with full-screen buffers), we have about 900K bytes to work with. By dividing that value by the number of applications programs that are to run concurrently, we can determine how much memory we can use for each application program.

Taking another approach, we could divide 900K bytes by an estimated average of how much memory each application (including files) will require, to see how many virtual consoles we could expect to have in our system. This result is still only a rough estimate, because the operating system must grow when the number of virtual consoles increases beyond four if additional screen buffers are added.

Sixteen-bit microprocessors other than the 8086/8088 have even more memory. Motorola's 68000 provides up to 16 megabytes of RAM, and the 80286 from Intel furnishes much more than that. Clearly, with such abundant memory, tomorrow's machines will be able to handle many consoles, as well as highly sophisticated integrated applications packages.

Two to eight virtual consoles will probably be offered in the first wave of Concurrent CP/M implementations. Four will probably be the average number. After the first wave, manufacturers may find themselves in a race to add consoles to get the attention of increasingly adept users.

Concurrent CP/M supports up to 16 logical disk drives -- separate floppy drives or several virtual drives on a hard disk, or combinations of the two. Any virtual console can log on to any disk drive to access programs or files.

And as do other Digital Research operating systems, each disk drive supports as many as 16 user numbers (areas), numbered 0 through 15. These areas are partitions within the file system's environment for grouping files. Files that are to be accessed by any or all user numbers on the drive are placed in user number 0 and given the system attribute. Otherwise, you must be working in the user number to access files within it.

Concurrent CP/M does have some limitations. Because disks are frequently shared by processes on different virtual consoles, you must be careful not to have an open file on a disk you are removing. In many implementations, you will be able to tell this from the status line.

Occasionally, you will come across a program that requires a lot of memory. Certain spreadsheets, debuggers, and assemblers fit into this category. If they are loaded first, they could use all available memory and prevent you from loading other programs. It is wise, therefore, to load these last, so that they can use only what memory is left.

Certain applications programs create temporary files during their operation that never appear in the directory. For that reason, if you load several programs from the same drive, they should be loaded in different user numbers to prevent the process on one console from overwriting the temporary file of a proces on another.

Concurrency on the IBM PC

The most popular implementation of Concurrent CP/M thus far is on the IBM Personal Computer. The PC is designed to support four virtual consoles with a minimum 256K bytes. Because the PC version of the operating system requires 90K bytes (with all four screen buffers used), you really would not want to run the system with less than 256K bytes.

A PC running Concurrent CP/M requires at least two disk drives. To load the system, the boot disk must be placed in drive A and a system disk in drive B. When the system is running, the boot disk is removed and applications programs are loaded from drive A. On the XT hard-disk version of the PC, the system can be automatically booted from hard disk when the power is turned on.

The system supports both serial and parallel printers, the number of which is determined by the number of printer cards installed, either in the main motherboard, or in an expansion interface. Both color and monochrome monitors can also be used with Concurrent CP/M.

Other Machines That Can Run Concurrent CP/M

The list of OEMs (Original Equipment Manufacturers) signed up for Concurrent CP/M is a lengthy one and is growing longer every day. It includes Digital Equipment Corp., Texas Instruments, National Cash Register, Fujitsu, Nippon Electric, Olympia, Eagle, Corona, Commodore, MADD, Vector Graphic, and Toshiba.

Computer systems using Concurrent CP/M may differ; they will probably boot differently, support different subsets of the CCP/M utility superset, or have a different status line. Most of the initial hardware implementations will support two to eight virtual consoles, and some OEMs will also provide unique hardware enhancements that will later build upon the operating system's inherent power.

Popular Application Combinations

One of the beauties of concurrency is that it becomes more useful as the operator becomes more adept. It is also immediately useful, even to the novice. A typical novice might, for example, run only one applications program and use another console to run system utilities. It is helpful to a beginner to be able to have the disk directory on one virtual console and the HELP utility on another, so that while he learns how to use the system, useful reference tools are always on line, only a keystroke away.

For those who make intense use of a particular applications program, it can be useful to have several versions of that program on the computer at one time. Such a setup would permit you to jump from one process to another without having to save, unload, and load another file. Managers can thus have several department's budgets on line on different virtual consoles, for instance, to permit quick comparisons of the impact of a percentage change on each.

More popular applications configurations will combine programs that will be more powerful to a user when run concurrently, rather than serially. Consider the programmer who can simultaneously run a debugger, an editor, and a compiler or assembler. As the debugger turns up bugs on one virtual console, the programmer can switch to another console and begin editing the program immediately, while on a third console the compiler works on a program that had been debugged earlier that day. After each edit, the programmer can then switch back to the first console, find the next bug, switch back to the editor, and continue in that manner until all the required tasks are completed. What used to be a long tedious linear process thus becomes an interactive one, eliminating much idle time.

Similarly, consider the busy project manager, who may have a word processor on one virtual console, a spreadsheet on another, a database-management program on a third, and the fourth connected to a modem awaiting a call. When the data is phoned in, it is stored in a file that can be shared by any of the other processes. It can be entered into the database or used by the spreadsheet as input for other projections, which may then be entered into the report being written on the word processor. Moreover, the data can be made available to different processes in a fraction of the time and by fewer people than it would have taken otherwise.

Consider the secretary who is connected to a network and has a word processor on one virtual console, a critical-path schedule on another, and an appointment calendar on a third. That secretary can receive input and transmit output to a large number of sources efficiently and, more important, be more up to date each time information is sent out than was ever possible before.

The Future of Concurrency

Concurrent CP/M is having an impact on software developers. Integrated software packages represent the first step in the development cycle of a new

generation of software, and other enhancements are appearing. For example, it has already become possible to interact with processes on several virtual consoles by means of dynamic windowing. As you work on one console, you can use one or more windows, of whatever size you specify, to show you what is going on in real time in other consoles. Furthermore, you can log on to any console being monitored and send input to it. A programmer can thus see which bugs are turning up on the debugger without ever having to leave the editor and simultaneously see how the compiler is running without having to log on to its virtual console.

Similarly, a project manager can use dynamic windowing to monitor data being received by a modem through a window in his word processor without having to switch screens. Furthermore, the manager can also work on those consoles because they are dynamic (i.e., it is possible to interact with them). In other words, if he presses the function key to log on to console 3 and has customized the window so that he can see enough output, the manager can work right there without switching screens, while also monitoring several other consoles. It may take some effort to customize each window to be able to see the crucial screen output needed, but the results can be impressive. Going back to the TV analogy, it is like having a small window in the corner of your TV screen showing you what is happening on the news while you are watching MASH. When a commercial comes up during MASH, you can always switch the big screen to the news and put the MASH channel in the window to wait for that commercial to end.

The hardware implications of concurrent processing are not as easy to speculate about. Because many machines handle concurrency well, it may be some time before we see hardware designed around concurrent processing. However, features that are desirable for this environment include the hard disk, which can alleviate file-storage problems; multiple floppy drives, for those who want to eliminate shared drives; and larger monitor screens to allow additional and bigger windows.

Conclusion

Three concepts can be used to summarize the effects of concurrency: synergy, holism, and heuristics. Synergy is the total effect of separate processes working together. It describes the cooperative action that single-user Concurrent CP/M permits.

Holism is the tendecy in nature to produce larger organisms from ordered groupings of smaller organisms. It is exemplified by people exploring the manifold possibilities that 16-bit computing technology represents and applying it to their needs.

Finally, heuristics, the principle of discovery as it applies to learning, will be practiced as computer users and designers discover the capabilities of concurrency. Concurrent processing will exert a powerful influence on the development of hardware and software, and the user interfaces to both.

Computer users have become more aware of how human thinking differs from the way a computer "thinks", and are not as easily impressed by computers as they

once were. Users now want enhancements that are extensions of the way they work; they don't want to be forced to adjust to the way a computer works. Concurrency is such an enhancement. It is an idea whose time has come.

EOF

CCPMDDJ.WS4 (= Concurrent CP/M article in DDJ magazine)

- "16-Bit Software Toolbox"
  Ray Duncan
  DDJ, Vol.8, No.2, February 1983, p.18

(Retyped by Emmanuel ROCHE.)

This month, I will devote the column to a review of Concurrent CP/M (CCPM), which has recently been released by Digital Research for the IBM Personal Computer. The CP/M-86 operating system was customized and marketed for the PC as an IBM software product, but apparently Digital Research was displeased with the result, and undertook to implement and sell the Concurrent version directly. This is fortunate from the standpoint of evaluating the operating system's performance, since it gives us an "official" version which has been presumably optimized and polished to the complete satisfaction of its inventors -- we don't need to concern ourselves with the hardware-dependent foibles that sometimes creep into OEM implementations. Versions of Concurrent CP/M for other 8086/88 microcomputers will be available eventually, but will probably be many months in arriving.

A Little Background Information

For those of you who are not familiar with the Digital Research family of operating systems, a brief overview may be in order. The original CP/M, now known as CP/M-80, is a single-user operating system for 8080/Z-80 microcomputers that provides hardware-independent console, printer, and file handling services for application software and program development tools. CP/M-80 has become the industry standard for 8-bit microcomputers. There are several hundreds of thousands of licensed installations, and an unknown but probably also enormous number of unlicensed copies in circulation (as well as a large number of licensed users of "CP/M compatible" operating systems such as CDOS, SDOS, and TurboDOS). The third major revision of CP/M-80 has just been released, with an improved file manager, enhanced utilities, and provision for powerful graphics extensions.

The multi-user version of CP/M-80 is known as MP/M-80, and is now in its second major release. The original version of MP/M had irksome deficiencies in performance, was difficult to configure, and included no provision for file or record locking, so that the integrity of data could not be protected when several programs were executing simultaneously. These problems were virtually eliminated in MP/M-80 version II, which is a very acceptable real-time, multi-tasking operating system; however, it is inherently limited by the speed and power of the host 8-bit microprocessor, and cannot practically support more than about four consoles in standard systems.

CP/M-86 is a translated version of CP/M-80 for the Intel 8086/88 family of microcomputers. From the user's point of view, operation of CP/M-86 is extremely similar to that of CP/M-80, with most of the same commands. Disk

allocation and file management are identical to that of CP/M-80, so that data may be easily transported between programs running on either operating system. CP/M-86 has been a fairly stable operating system since its release, and has not required any major revisions. The performance of CP/M-86 has been limited somewhat by the constraint of compatibility with the 8-bit systems, and it has run into stiff competition for the hearts of microcomputer manufacturers and users from the Microsoft counterpart "MS-DOS".

Finally, MP/M-86 is available as the upward-compatible, multi-user version of CP/M-86. This is an exceedingly powerful microcomputer operating system, readily capable of supporting up to sixteen users. It offers many additional sophisticated services to the programmer, such as queue management, dynamic memory allocation, easy installation of Resident System Processes (RSPs), multiple printer support, and task spawning. It requires a comparatively large amount of memory (256 Kbyte RAM systems are common) and is usually hard-disk based. Efficient implementations of MP/M-86 with proper interrupt handling and buffered I/O are quite complex and require several man-months of work. For this reason, it has been slow to appear on the market for the various 8086/88 microcomputers. An unusual variant called MP/M 8-16 is sold by G&G Engineering for the CompuPro 8085/8088 dual processor board. It allows the interleaved execution of programs coded for the 8080 or 8086 on the same microcomputer without any special intervention by the user.

What is Concurrent CP/M?

To quote Digital Research, "Concurrent CP/M is a single-user, multi-tasking operating system that lets you run multiple programs simultaneously by dividing tasks between virtual consoles". Inspection of the documentation, services, and utilities of CCPM reveals it to be a slightly stripped-down version of MP/M-86, with some rather novel modifications. From the operating system's perspective, there are four user consoles which can originate and interact with executing processes. From the user's point of view, there is one physical console which serves as a "window" and can be switched at will between any of the four virtual consoles.

The terminal handlers for the four virtual consoles buffer output independently; when a program writes to a virtual console that is not currently being viewed by the user, the characters are stored into a RAM swapping buffer and optionally spooled into a disk file, so that no output is lost. When that virtual console is finally selected by the user, the saved output is recalled from the swapping buffer, and displayed on the screen. Toggling the physical console between the four virtual consoles is accomplished merely by holding down "Control" and keying a number between 0 and 3 on the numeric keypad; the response by the operating system is instantaneous. While the system is running, the status line at the bottom of the screen displays the following information for the currently-selected virtual console: the name of the active task, the unit number of the attached printer, names of disk drives with any open files, the time, and the status of certain keys such as CAPS LOCK and NUM LOCK.

System Requirements

The power of Concurrent CP/M carries a hefty price tag for both hardware and software. The first requirement, of course, is that you fork over $350.00 to Digital Research for a single-user license.

Next comes the hardware. A minimum of 256 Kbytes of RAM is recommended by Digital Research; at least 192 Kbytes must be present in order to boot up the system. If more memory is available, part of it can be allocated as a "virtual disk".

Two disk drives are mandatory; double-sided drives make the system much more pleasant to use, due to the large amount of disk storage occupied by the various system utilities and the HELP text file.

Commands and Utilities

Users of CP/M-80 and CP/M-86 will find much that is familiar. The well-known commands DIR, ED, ERA, PIP, REN, STAT, SUBMIT, TYPE, and USER that hearken back to the early days on the 8080 operate in the same old tried-and-true manner.

The utilities ABORT (terminate executing task), DSKRESET (log in new diskette), ERAQ (selective file erase with query), PRINTER (select list device), SDIR (formatted directory with file sizes and attributes similar to STAT *.* or LST), SHOW (display system and disk parameters), SET (control disk and file attributes, passwords, and time stamping), and TOD (set or display system time and date) work in the same manner as their counterparts on MP/M systems.

The program development tools ASM-86 (8086/88 assembler), DDT-86 (interactive debugger), and GENCMD (create executable command files) are identical to their CP/M-86 counterparts.

Finally, there are a number of utilities which are unique to the IBM Personal Computer implementation of Concurrent CP/M. CONFIG allows the user to control the baud rate, word length, parity, and stop bits of the serial ports. DSKMAINT provides formatting and copying of single- or double-sided diskettes. FUNCTION allows the user to specify character strings for each of the programmable function keys. HELP provides interactive, on-line explanation and examples for any of the system commands. SYSDISK allows the user to specify which disk drive will be searched for a program if it is not found on the "current" or default disk. Finally, VCMODE is used to control the characteristics of each of the virtual consoles. See Table 1 below for a more condensed list of the system commands.

Table 1. Concurrent CP/M Commands and Utilities

CMD Name	Action
ABORT	Stops execution of a task
ASM86	8086/88 assembler

CONFIG Sets operating parameters for serial ports
DDT86 8086/88 interactive debugger
DIR Display disk directory
DSKMAINT Initialize or copy disks
DSKRESET Log in diskette
ED Line editor
ERA Erase file(s)
ERAQ Erase file(s) with query
FUNCTION Program special function keys
GENCMD Create executable file from assembler output
HELP On-line assistance for system commands
PIP Transfer and manipulate files, among other things
PRINTER Select or display attached printer number
REN Rename file(s)
SDIR Sorted directory listing with many options
SET Controls password protection and file attributes
SHOW Display disk and system parameters
STAT Display file information, assign attributes
SUBMIT Batch processing utility
SYSDISK Designate the "system" disk drive
TOD Set or display time and date
TYPE Display contents of a text file
USER Select or display the current user number
VCMODE Set mode and buffer size for virtual consoles

File Handling

Concurrent CP/M incorporates the Digital Research BDOS (Basic Disk Operating System) version 3, which is also embedded in MP/M version II and supports many advanced file management options. This BDOS apparently will become the standard for all of the DRI operating systems, including the new CP/M-80 version 3.0 (also known as CP/M Plus).

On the command level, diskettes and individual files can be protected by passwords. Time stamping for the creation, update, or last access of a file is supported. In addition, files may be marked with the attributes SYSTEM, READ-ONLY, and ARCHIVED.

On the programming level, executive service calls available to application programs provide full support for file and record locking. If a file is opened in "unlocked" mode by a program, it remains accesssible to any other task which is also willing to open it in "unlocked" mode. A file can be opened and shared in "read-only" mode by any number of concurrent processes. If a program wishes exclusive access to a file, it can open it in "locked" mode; the operating system will deny any requests by other tasks to open the same file.

When two or more processes share a file in "unlocked" mode, both having the privilege of write access, there is a potential danger of loss of information. Consider the following situation: task #1 reads a record from the disk into memory. By coincidence, task #2 reads the same record immediately thereafter. Task #1 makes some changes to the record and writes it back to disk. Task #2 makes some different modifications to the data and writes it to the disk.

Depending on which task happens to get serviced for its disk write request first, one or the other set of changes will be permanently lost. In addition, one of the tasks may take some erroneous action based on the contents of the disk record which was not yet updated by the other task.

The capability called "record-locking" is designed to forestall such disasters in a multi-tasking environment. When a program issues a read request for a record, it can optionally specify that the record should be "locked", that is, the record is made unavailable for updating by any other process until it is released by the program that issued the "lock" request.

This allows a record to be safely modified by one program without fear of interference by other tasks that may be executing simultaneously. As an alternative to file and record locking (which consumes system resources and can slow down the system), a "Test and Write Record" function is provided, which verifies that the disk copy of the record is unchanged before honoring a request to update it.

Additional file security is provided by extensive checksum verification of disk directory entries and all active file control blocks. This is designed to prevent any compromise of data integrity by a task which has crashed, run out of control, or is performing disk access in a non-standard manner. It should be noted that some techniques of file access which were legal under earlier versions of CP/M, especially those involving direct manipulation of the File Control Block (FCB) or simultaneous access to different file "extents", are intercepted and aborted by Concurrent CP/M and the other operating systems which use the third generation BDOS.

Even with the safeguards described above, concurrent access to multiple files by more than one program is tricky business. It requires careful analysis and a programmer experienced with multi-tasking operating system in order to be done safely. For example, it is quite easy to have two programs get into a "race condition" where each has locked a record that the other needs, and thus neither one can continue to execute -- system resources are tied up irrevocably, and usually the operating system will slow down and "die".

Executive Services

Concurrent CP/M offers a multitude of function calls for use by the application programmer; a complete list of the Concurrent CP/M operating system services is given in Table 2.

Table 2. Concurrent CP/M Function Calls

Where I have designated console functions as "compatible" with CP/M-86 or MP/M, I am ignoring issues of "virtual" versus "physical" consoles. Where I have called file and record functions "compatible" with CP/M, I am assuming an application running as the sole task, and disregarding problems of file or record sharing/locking.

Function	Action

0	%	System reset -- terminate calling process
1	+	Console input
2	+	Console output
3	#	Raw console input
4	#	Raw console output
5	+	Output to defaut list device
6	*	Direct console I/O
7	#	Get I/O byte (not implemented)
8	#	Set I/O byte (not implemented)
9	+	Print string
10	+	Buffered console input
11	+	Read console status
12	%	Return BDOS version number
13	%	Reset disk system
14	+	Select disk
15	+	Open file
16	+	Close file
17	+	Search for first directory match
18	+	Search for next directory match
19	+	Delete file
20	+	Read sequential
21	+	Write sequential
22	+	Make file
23	+	Rename file
24	+	Return log-in vector
25	+	Return current disk
26	+	Set memory address for disk transfer
27	+	Get address of disk allocation table
28	+	Write-protect disk
29	+	Get disk readn-only vector
30	*	Set file attributes
31	+	Get address of Disk Parameter Block
32	+	Get/Set user code
33	#	Read random record
34	#	Write random record
35	+	Compute file size
36	+	Set random record
37	+	Reset disk drive (to not logged-in state)
38	#	Access drive (place on lock list)
39	#	Free drive (remove from lock list)
40	*	Write random with zero fill
41	#	Test and write record
42	#	Lock record
43	#	Unlock record
44	#	Set multi-sector transfer count
45	#	Set BDOS error mode
46	#	Get amount of free disk space
47	#	Chain to program
48	#	Flush buffers
49		(Not inplemented)
50	*	Direct BIOS call
51	*	Set segment address for disk transfers
52	*	Return segment address for disk transfers
53	*	Get largest available memory region

```
 54 *      Allocate maximum memory available at absolute address
 55 *      Allocate memory segment
 56 *      Allocate memory segment at absolute address
 57 *      Free memory segment
 58 *      Free all memory segments
 59 *      Program load
100 #      Set directory label
101 #      Return directory lable
102 #      Read extended file contol block
103 #      Write extended file control block
104 #      Set date and time
105 #      Get date and time
106 #      Set default password for file access
107 #      Return serial number
128 #      Memory segment allocation
129 #      Memory segment allocation
130 #      Free memory segment
131 #      Poll device (test logical interrupt flag)
132 #      Wait for a system flag
133 #      Set a system flag
134 #      Create system queue
135 #      Open queue
136 #      Delete queue
137 #      Read message from system queue
138 #      Conditionally read message from queue
139 #      Write message into system queue
140 #      Conditionally write message into queue
141 #      Delay for a specified number of clock ticks
142 #      Call system dispatcher
143 #      Terminate calling process
144 #      Create (spawn) a process
145 #      Set priority of calling process
146 #      Attach console
147 #      Detach console
148 #      Set default console number
149 #      Assign console to another process
150 #      Interpret and execute a command line
151 #      Call function in resident procedure library
152 #      Parse filename
153 #      Return number of default console
154 #      Get address of System Data Area
155 #      Get date and time
156 #      Return address of process descriptor
157 #      Abort specified process
158 #      Attach list device
159 #      Detach list device
160 #      Select default list device
161 #      Conditionally attach list device
162 #      Conditionally attach console device
163 #      Return Concurrent CP/M version number
164 #      Return number of defaut list devices
```

Where:

+ = Compatible with CP/M-80, CP/M-86, and MP/M
* = Compatibel with CP/M-86 and MP/M
# = Compatible with MP/M-86
% = Similar to CP/M-86 or MP/M-86, but with subtle differences

In addition to the usual functions 1-40 which are present in ordinary CP/M 2.2 systems, there are entire new repertoires of capabilities which are highly reminiscent of a powerful minicomputer real-time operating system such as DEC's RSX-11M. For example, functions 53 through 58 and 128 through 130 allow a program to dynamically request, use, and release memory from the system pool, based on requirements determined at run-time. Functions 131 through 133 allow peripheral device drivers to communicate with other tasks through use of "semaphores" or software-controlled flags.

Functions 134 through 140 allow tasks to create and maintain "queues", which can be used to pass information asynchronously between processes. Functions 146 through 149 and 158 through 162, among others, allow a given program to attach or detach itself for input and output from any of the virtual consoles or printer devices. Functions 150 and 152 assist an application program in parsing filenames and interpreting command lines. Lastly, function 144 allows a task to initiate or "spawn" other processes which can run concurrently.

An interesting observation in passing is that BDOS function 12, which fetched a simple one-byte operating system version number in CP/M-80, has been drastically extended. It now returns a sixteen-bit parameter that is broken up into a number of bit fields to specify the BDOS version, the environment (whether CP/M, Concurrent CP/M, MP/M, or any of the preceding with networking) and the CPU type. Digital Research is evidently planning well ahead here. The day is not far off when the entire DRI family of compilers and operating systems will be running on the 8080, Z-80, 8086/88, 68000, Z-8000, and 16000 microprocessors.

Noteworthy Features

As we have already seen, Concurrent CP/M contains much that is new and different from the CP/M systems that we are all accustomed to, but there are a few features which are especially welcome and should be mentioned here.

- File access has been enhanced with a special "burst mode", which allows the transfer of one to sixteen records with a single function call.

- A complete set of BDOS disk error codes are defined, and an application program may choose to handle all physical and logical disk errors. This makes it possible for editors, disk utilities, and other critical programs to handle formerly fatal events (such as the disk drive door not being closed) in a graceful way. The dreaded message "BDOS ERROR ON XX", which has been the cause of so much cursing by users and programmers alike, can now be eliminated forever.

- The new system utilities, DSKMAINT, CONFIG, and FUNCTION, are screen-

oriented, menu-driven, make extensive use of the programmable function keys and video attributes, and are practically idiot-proof. I hope that this is sign of the trend to be taken by future Digital Research products.

- A "virtual disk" capability, sometimes known as "RAMDISK" or "M-DRIVE", is built into Concurrent CP/M. If the operating system finds that the system has memory installed at the proper address, the virtual disk is automatically made accessible as drive M. It can be used to store the commonly needed system utilities such as PIP and STAT, and any temporary files created by SUBMIT, improving the perceived responsiveness of the system dramatically.

Documentation

The manuals for Concurrent CP/M demonstrate a quantum jump in usability as well as flashiness. The documentation is divided into the 204-page "Users' Guide" which describes operation of the various common system commands and utilities, and a 338-page "Programmer's Reference Guide" which covers the file system BDOS function calls, assembler, and debugger in detail. Colored text is employed to emphasize example of actual system interactions. Liberal use of tables, diagrams, and appendices make information easy to locate. The manuals are packaged together in a hardboard, three-ring binder with accompanying box cover that is slightly larger than the IBM format.

As is usual for Digital Research documentation, the manuals are aimed at a relatively sophisticated reader, but they are well organized and lucid. There are a vanishingly small number of typographical and grammatical errors. In fact, the only technical error I have found so far that might affect a programmer is in Appendix H of the "User's Guide", and involves an overlap between the video escape sequence for "Erase Entire Line" and "Insert Blank Line". Clearly, the new department Digital Research set up to enforce standards of documentation quality has made its presence felt.

What Price Concurrency?

With 320 Kbytes of main memory, I have found it possible to run three 64-Kbyte tasks simultaneously. This implies that the operating system with its various buffers requires about 128 Kbytes of RAM. The load image "CPM.SYS" on the disk drive occupies 92 Kbytes.

Naturally, operating system support for concurrent task execution entails a certain amount of overhead for servicing the system clock, maintaining various queues and lock lists, saving task contexts, and dispatching tasks based on their priority and demand for system resources. To try to assess the actual effect of this overhead on task execution time, I ran a well-known Forth prime number sieve benchmark program on the conventional IBM implementation of CP/M-86 and on Digital Research Concurrent CP/M. This is a classic "CPU-bound" program, with no disk I/O and only a small amount of terminal I/O. The program completed in 27 seconds on ordinary CP/M-86, and in 28 seconds on the

Concurrent version (with no other active tasks) -- approximately a 4% decrease in throughput. It is interesting to note that this benchmark under Microsoft MS-DOS requires only 25.7 seconds; the difference can probably be attributed to the time which is wasted updating the status line display on IBM's CP/M-86.

Then I ran the same CPU-bound benchmark on Concurrent CP/M as two simultaneous tasks. They each completed in 56 seconds, exactly twice the amount of time required when the benchmark executed as a single task. This implies that the operating system overhead to support concurrency is relatively independent of the number of processes that are active.

To assess the effect of concurrent tasks on disk performance, I coded a small routine which performed 100 random reads of 1-Kbyte records in a 50-Kbyte contiguous data file. This is a truly disk-I/O-bound benchmark, since it performs almost no computation at all (except for the random number generator) and simply requests one disk access after another. On IBM's version of CP/M-86, the program completed in 75 seconds -- a surprisingly poor performance. On Concurrent CP/M, with one task running, the benchmark executed in 52 seconds, approximately a 30% improvement! However, when the system was really put to the acid test with two copies of the disk-bound benchmark running simultaneously (accessing the same file on the same drive), its performance really fell apart, and each task required about 430 seconds to complete. This seemed to be largely due to a large amount of disk "thrashing" that took place as the two tasks requested records in different extents, forcing many inspections of the disk directory.

I expect that the most common use of multi-tasking by the average user will be with one CPU-intensive program, such as a spreadsheet or word-processor executing concurrently with spooling of a file or some other peripheral-bound process. Under such circumstances, the decrease in responsiveness that can be attributed to multi-tasking is practically unnoticeable. To verify this impression, I ran the two previously described types of benchmarks simultaneously. The CPU-bound program completed in 32 seconds, and the disk-bound routine finished in 59 seconds, each showing only 14% degradation over the tasks executing alone.

A Few Small Gripes

In using Concurrent CP/M for one month, I have not found any earth-shaking bugs or omissions. There are a few minor annoyances which I have taken the liberty of itemizing below.

There is no check for the disk drive door being closed; if a diskette is not loaded into a drive when it is accessed, the system will hang indefinitely without an error message.

There is no provision for accessing the bit-mapped graphics capabilities of the PC, even at the minimal level of simply plotting points. Control of cursor position, character attributes, selective erasing, and foreground and background colors in text mode is provided by means of escape sequences -- very nice. But for unclear reasons, Digital Research did not choose the same escape sequences for screen control as were used in the IBM version of CP/M-

86, so that vendors of program packages will have to maintain and sell two separate versions.

Output to the video screen is inexplicably slow. Text appears to write on the screen at the equivalent of somewhere around 4800 baud. I realize that there is unavoidable overhead involved in buffering output for four virtual consoles. Still, on such a powerful machine with memory-mapped video to boot, one might hope for a somewhat better effective speed of display.

The state of the CAPS LOCK, NUM LOCK, and WRAP parameters is not maintained individually for each virtual console. This is a minor point, but when you are flipping back and forth between an editor and some other utility, it is a nuisance to have to toggle these keys when the operating system could just as easily keep track of them for you.

The 8086/88 assembler does not include the mnemonics for the Intel 8087 numeric coprocessor. Although they can be implemented by the user with the CODE-MACRO facility, this is a very laborious process.

The system boot file, startup batch files, and all of the commonly-used system utilities will fit nicely on a double-sided disk in drive A, making the two-diskette system distributed by Digital Research unnecessary. But, even if everything the system needs is on drive A and no auto-start batch files are present, it will still try to access drive B during the cold boot sequence. I have found no way to configure the system so that it will leave drive B alone.

Summary

Concurrent CP/M provides the power to implement applications on a desktop machine that would have been unthinkable even two or three years ago. Even for the casual user, the flexibility and conveniences that will result from the ability to execute several programs concurrently is going to cause a rapid "revolution of rising expectations" that will have to be answered by the other manufacturers of microcomputer operating systems. I rate this software "Excellent".

- "CP/M Exchange"
  Robert Blum
  DDJ, Vol.9, No.1, January 1983, p.78

(...) I recently visited the Dallas office branch of Digital Research. While there, I was fortunate enough to be given a demonstration of Concurrent CP/M with windows on what appeared to be a plain vanilla IBM PC. Not that there is anything wrong with the PC. It is a fine machine, backed microcomputers that I have had the opportunity to use.

We began by bringing up WordStar, my favorite word processing package, in task area one. We then reduced WordStar's screen by two lines, from 24 to 22, to make room for three 2-line tall, 25-character wide windows to be used by other tasks. Next, dBase II was executed in task area two. As demonstrations go, there were two submit files available to perform various utility tasks, such

as copying files from one disk to another. These were started in task areas three and four.

After typing in a short letter and saving it, I commanded WordStar to print while I used dBase II. The transition from WordStar to dBase II required only one control sequence. After telling dBase II to find some bogus data, I switched over to task area three to see how things were going there. Again, only one control sequence was necessary. The printer had stopped, so I switched back to WordStar, where I was greeted with the "no file" menu. My attention was drawn to window two, because dBase II had completed its job and was waiting for another command.

And so it went: readily jumping from one task to another, without ever losing track of what was happening in other areas. The only problem I had was keeping the machine busy. This demonstration emphasized, however, that the PC's memory-mapped video display is, to a large degree, accountable for this amazing performance. If it had been necessary to wait for an entire screen of data to be written to a terminal at 9600 or 19.2K baud when swapping tasks, much of the system's performance would have been nullified.

Concurrent CP/M is a software package that can transform a machine of modest capabilities into a real workhorse. (...)

EOF

CCPMMAM.WS4 (= Concurrent CP/M article in "Mprocs And Msystems")

- "Concurrent CP/M-86 and recent advances in operating systems"
  Howard Kornstein
  "Microprocessors and Microsystems", Vol.7, No.8, October 1983, p.391

(Retyped by Emmanuel ROCHE.)

Concurrent CP/M-86 provides a multi-window environment for executing standard application software in real time.

When the first personal computers were being designed, CP/M was proposed as an operating system to match their low cost, small memory and small disc storage. Concurrent CP/M-86 allows simultaneous execution of real-time application software. The structure of the operating system is described. The development of the user interface is sketched. Concurrent CP/M has also been extended with a graphics interface. Future trends in concurrency are suggested.

CP/M is one of the key products that is associated with the growth of the personal computer industry. When the first personal computers were being designed, Gary Kildall determined to provide an operating systems environment for the emerging low-cost personal computers. CP/M was originally scaled to match the capabilities of these early personal computers, which provided small amounts of memory space, minimal disc storage systems, and Intel 8080 or Zilog Z-80 processors. The design goals for CP/M-80 were compaction of code to conserve memory resources, provision of a robust file-handling system and, above all, portability of the operating system over a wide range of machine models.

CP/M-80 was designed as a single-user single-tasking batch-oriented operating system, and proved an ideal match of system software with the personal computer hardware available in the late 1970s. Perhaps the most significant concept that CP/M introduced was the capability for machine model independence, i.e. the operating system could be ported to a wide range of machine configurations made by different manufacturers, as long as those machines provided a minimal memory space, an Intel 8080 or Zilog Z-80 processor, a floppy disc system, and keyboard and screen functions. To accomplish this, CP/M was partitioned into machine-independent and machine-dependent parts known as the BDOS and BIOS, respectively.

As the personal computer system developed, CP/M expanded into a family of operating systems to suit different personal computer environments. Other members of this family include MP/M II and MP/M-86, respectively 8- and 16-bit multi-user multi-tasking operating systems, which allow a number of users to share a common computer system, and file and printers peripherals.

CP/NET was introduced to provide for distributed personal computing systems, where a number of workers in an office had local computing power, but wanted to share global resources, e.g. a shared database.

This year, Digital Research has introduced Concurrent CP/M-86, which represents a significant advance in the functionality and capability of a personal computer operating system. Concurrent CP/M was designed as an operating system complementing the advanced-performance personal computer, and enhancing the overall system performance for application software. Concurrent CP/M was designed especially to provide an advanced multi-window environment, allowing simultaneous execution for several real-time applications, in a way that was simple for a user to understand.

Let us explore these capabilities, and how they are implemented in Concurrent CP/M-86.

Internal structure of Concurrent CP/M

Concurrent CP/M-86 was created to provide an environment for execution of standard application software in an enhanced hardware, and an advanced user-friendly environment. To accomplish these objectives, the Concurrent CP/M-86 operating system was structured on a real-time multi-tasking operating system which has a kernel that allows 256 tasks to co-reside in a system, and which schedules task execution based on both priority pre-emption and round-robin scheduling.

A priority pre-emptive kernel is typical of many real-time operating systems which have been used in time-critical control applications, examples being DEC's RSX-11 or Intel's RMX-86. Such real-time systems allow for critically time-dependent functions to be acted on with good response time from the computer system. This allows real-time activity, such as communications processing or real-time control, to be brought into the environment of the personal computer.

Round-robin scheduling is typical of time-sharing systems, Unix being a very good example. In such systems, we want to share out a computer system among equal-priority users, giving an equal slice of computer resource to each user. This is very much the environment of commercial processing. The personal computer today needs to do a mixture of time-dependent and equal-resource processing, hence the structure of Concurrent CP/M's despatching mechanisms. The Concurrent CP/M real-time supervisor provides for process creation, process deletion, process despatching, process communications, queue management, flag management, and device control.

The operating system supervisor interfaces with 4 major software subsystems: the real-time monitor, memory pool manager, basic disc operating system, and character I/O module. The operating system supervisor can maintain a number of CP/M processes simultaneously.

It is interesting to note that any individual CP/M application can be written without any knowledge of other CP/M applications, which will co-exist with it in the concurrent environment. Fundamentally, each runs in a separate independent operating partition. The operating systems calls which are made by a concurrent application are a slightly extended set of standard CP/M operating system calls. This allows for portability of a CP/M application from a single-tasking to a multi-tasking environment, with minimal design change by

the software author.

The extended I/O system (XIOS) provides the same function as the previously mentioned BIOS of conventional CP/M. It connects the virtual operating systems environment with the real machine environment for a particular machine model of personal computer.

The things that make Concurrent CP/M user-friendly are the technical capabilities provided by the virtual console session manager and the permanently resident terminal message process. This system software provides the replication of 4 console equivalents to which an application may talk when Concurrent CP/M is running.

Concurrent CP/M-86 user interface

What we have been describing up to now is a high-performance operating system environment, allowing multiple CP/M applications to run in real-time. We now consider the CP/M-86 interfacing.

The advanced concepts of user interface provided by the operating system are especially noteworthy in Concurrent CP/M. The personal computer is characterized by having a naive computer user. He has little experience, and little interest, in the complexity of the user interface of a sophisticated operating system. His interests lie in executing important applications packages in a way which maximizes the use of his personal computer system, and makes the use of his own time on the personal computer most efficient.

There has been much progress in defining an efficient user interface for a more naive computer user. Much of this pioneering work was done by Xerox Corporation in its Xerox Star program, and has been emulated in the microcomputer world with Apple Lisa and VisiCalc VisiOn technology. The Concurrent CP/M user interface is structured on similar principles, but has been able to implement a more modern system than any of these alternatives by providing an application-independent environment. The personal computer has evolved from being a system which is used at short intervals of time during the working day into a workstation which is continuously operative throughout the working day, and is the one instrument on which all office-related activity is taking place. In essence, it provides one paperless environment as pioneered in the Xerox Star system. As users go through the working day, they tend to get involved with many different applications, first initiating them, and then suspending a particular job while a more important office activity takes place. Ideally, users would like to see some of their work continued in the background, while they concern themselves with matters of greater priority. The user interface of Concurrent CP/M allows users to accomplish this by providing for virtual computer systems. Each computer system has its own console and keyboard. Switching between these machines is done at one keystroke; thus, operators can begin word processing on one virtual screen, be interrupted and begin business model computation, then go back to the word processor later in the work session, begin a compilation on a third virtual screen, and allow it to run while going back to complete the letter on the word processor. Users never have to reload their software or back out of the context of the job that they had stopped looking at on their physical console.

The 4 or more screens that are part of the Concurrent CP/M user interface can be presented as separate screens available at a keystroke, or can be overlapped or coresident windows which simultaneously show work in progress on the various CP/M applications running on the system.

Graphics interface for Concurrent CP/M

The modern personal computer typically provides graphics capabilities in its hardware. To provide a graphics interface to applications packages which may want to exploit good visual presentation, Concurrent CP/M has been further developed to allow for graphics extensions which are totally machine-independent. This is accomplished by modelling operating systems extensions on standards defined in the Virtual Device Interface pioneered as part of the Graphics Kernel System proposed by the Internation Standards Organization.

The GSX graphics systems extensions allow applications packages with graphics capability to be machine-independent. They can obtain graphic input from digitizing tablets, selector functions, or pointer devices such as the mouse or joystick and light pen.

Graphics output can be provided to the graphics screen of the personal computer, or to various graphics plotters or printers. To accomplish machine-independent graphics, the graphics system extensions are partitioned in much the same way as CP/M itself. The GSX system consists of a machine-independent and a machine-dependent subsection. The machine-independent system is known as the graphics disc operating system (GDOS). The machine-dependent system is the graphics I/O system (GIOS). The configuration of the GIOS is carried out by a particular personal computer manufacturer.

Future trends in concurrency

The capabilities of Concurrent CP/M are established in its real-time kernel. Based on the real-time capabilities of Concurrent CP/M, this product can naturally evolve into a high-performance networking operating system, or a distributed multi-user system. Such products will be the natural derivatives of this new generation of personal computer operating systems.

EOF