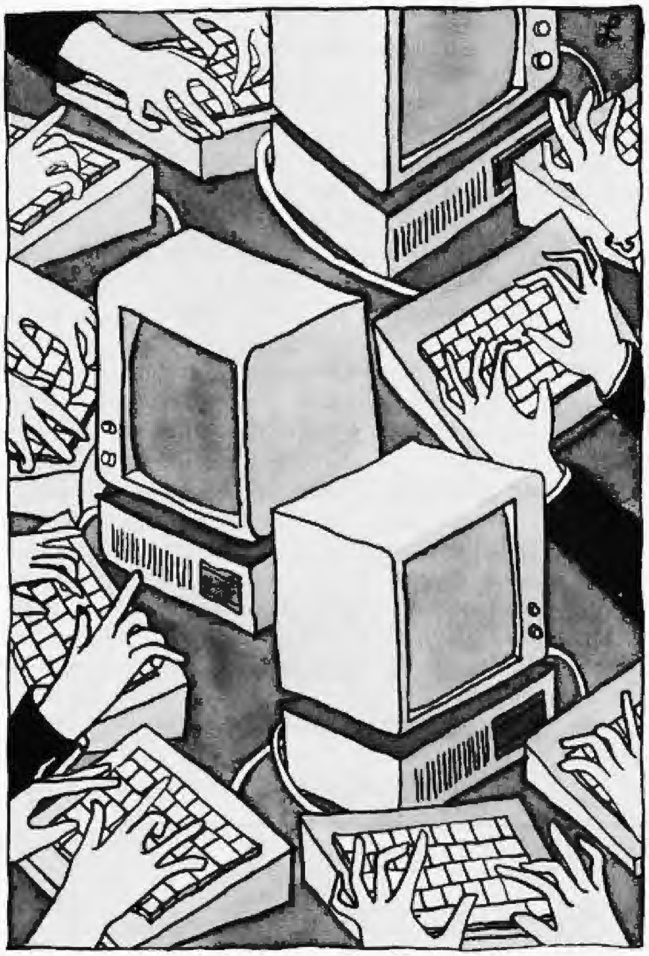
PICK, COHERENT, AND THEOS

Multiuser systems on the IBM PC XT

BY MARC J. ROCHKIND



PICK, COHERENT, AND THEOS are all multiuser, multitasking operating systems for the IBM PC XT. [*Editor's note*: THEOS *is the new name for the* Oasis *operating system. In this article,* THEOS *refers to* Oasis86. *The name was changed this year.*] None of them can run MS-DOS programs, so you can't use them with the familiar spreadsheet, word-processing, and other packages for which most people buy XTs. These operating systems are intended for more specialized purposes. Pick and THEOS are designed as a basis for multiuser database applications. Coherent is more general in principle (it's a UNIX clone), but its most appropriate use is for program development in C.

Because these operating systems are so different from each other and are designed for different purposes, I won't compare them feature for feature. Instead, I'll discuss them separately and evaluate them only in terms of their intended uses.

I tested release 1.3 of Pick (from Pick Systems, 1691 Browning, Irvine, CA 92714, (714) 261-7425), release 2.3.43 of Coherent (from Mark Williams Co., 1430 West Wrightwood Ave., Chicago, IL 60614, (312) 472-6659), and release 7.0 of THEOS (from THEOS Software Corp., Suite 100, 201 Lafayette Circle, Lafayette, CA 94549, (415) 283-4290). Pick and Coherent sell for $495 (including programming language). THEOS sells for $595 without a language; C or BASIC costs about $400 more. None of these operating systems are copy-protected.

I tested the systems on my IBM PC XT, which consists entirely of IBM parts (even the memory board). It has a 10-megabyte hard disk, a 360K-byte floppy, 512K bytes of RAM (random-access read/write memory), and a monochrome display. I used a Zenith Z-19 terminal (at 9600 bps) to exercise the multiuser features of each operating system. My printer is connected to the parallel port.

All three systems supposedly work on many other hardware configurations besides a plain-vanilla XT, but I didn't try them on any IBM-compatibles or with other kinds of hard disks.

These operating systems are too complex for me to claim to have tested them thoroughly. In the few days I spent with each, I tested about 10 percent of their functions. With such a sampling, a defect gets emphasized more than it should in view of the overall capabilities of the product.

OVERVIEW OF PICK

Pick is a special-purpose operating system designed to run the Pick database. In fact, when you use Pick you do not encounter the usual boundary

(continued)

Marc J. Rochkind is president of Rochkind Software Corp. (3080 Valmont Rd., Boulder, CO 80301), *which markets the business programming language* RIDE. *He is the author of* Advanced UNIX Programming (Prentice-Hall, 1985).

ILLUSTRATED BY LYNNE BUSCHMAN

between operating system and database system. For example, Pick doesn't have an operating-system command to list the files in a directory. Instead, you use a data-management facility to list the entries in your account dictionary. Pick terminology is different, too: Commands are called verbs, records are called items, fields are called attributes, and so on.

For anyone used to more traditional operating systems, the Pick approach seems strange. But after a day or so, I got used to it.

To me, the most remarkable aspect of Pick is that it tends to have just one of each function or feature rather than several. It has only one text editor that edits source files, memos, data records, and dictionary entries. There's only one file type for data; an item can be a data record, in which case the fields have their usual meaning, or it can be a source program or document, in which case each field corresponds to a text line. Its uniformity of treatment and simplicity of design make the Pick system easy to understand once you master the basic concepts. They undoubtedly contribute to the efficiency and reliability of Pick as well.

Internally, Pick implements a virtual computer in two senses. First, data and programs are accessed via a virtual-memory system that divides real memory into a large number of 512-byte frames. Data and pieces of program are moved between disk and real memory as needed. Second, programs are compiled into Pick assembly language, which is not 8088 code but the code for a virtual computer implemented by an interpreter deep within the Pick kernel. This interpretation is necessary for the virtual memory to work because the XT lacks virtual-memory hardware. In addition, the interpreter protects users from interfering with one another. This is a problem with other multiuser operating systems that run on the XT because the machine has no memory protection of its own.

The Pick database is structured hierarchically as shown in table 1. For further information on Pick file structures, see "The Pick Operating System, Part 1: Information Management" by Rick Cook and John Brandon, October 1984 BYTE, page 177.

The flexibility of items, attributes, values, and subvalues lets you organize data differently with Pick than in many other database systems. For example, in a medical office database you might have a patient record and an additional detail record for each visit. However, Pick can keep all the patient's data in a single item. It might have an attribute called, for example, Visit, and you could add an additional value with each visit. This organization is particularly easy to handle from within Pick programs because the entire item is read and written as a unit.

You don't have to specify sizes when you allocate a file or define an attribute. Attributes have a length that is used for printing reports, but that length does not limit the amount of data that you can insert. In fact, you can insert data into records without even defining the attributes. All good database systems allow some flexibility in handling data, but few go to the extremes of Pick.

The Pick database supports record locking but not transactions (a transaction is a sequence of database updates that are to be kept atomic). If a program aborts before a transaction is completed, the partial changes remain intact. There's also no transaction log to restore the database to its current state if a disk fails. You must make backup copies of the database at regular intervals, and if the database is lost, you will lose all updates since the last backup. This naive approach to database recovery is standard for microcomputers, but I expected more from Pick.

The Pick system comes with several major subsystems. The command processor (shell) is called TCL (terminal-control language). It's nothing special—it just processes commands a line at a time.

ED is a line editor, not a screen editor. You edit using line numbers, and you can't see your changes in context unless you issue a list command. After inserting or deleting a line, you can't do much else without getting a SEQN? message from the editor. You have to issue an F command to make your updates active before you can proceed. It's bad enough that the Pick user has to struggle with a line editor, but this one is the worst I have ever used.

The PROC processor is used to collect command lines, including those for interactive commands like ED, into stored procedures. PROC can be quite elaborate; it has features for I/O (input/output), branching, terminal prompting, and so on. You can think of the PROC processor as a command-level programming language, similar in concept but not in design to the UNIX shell.

The query and report-generation

Table 1: *The Pick database hierarchy.*

System dictionary	One per system; contains account names, passwords, and other administrative information.
Master dictionary	One per account (user); contains a user-specific vocabulary of commands and the user's filenames, so it acts as a log-in directory.
File dictionary	Many per account; contains definitions of fields and relationships among them.
Data file	One or more per data dictionary; contains data.
Item	Many per file; contains data for one record, up to 32,767 bytes. Might be a data record, source program, etc. Each item has a unique ID. All Pick files are accessed by hashing.
Attribute	Many per item; corresponds to a field of a record, a line of text, etc.
Value	One or more per attribute; can be used to record multiple instances of an attribute (e.g., names of dependents).
Subvalue	One or more per value; similar to value but one level deeper.

language is called ACCESS. It has facilities for selecting, sorting, adding and counting (with control breaks), and listing—the usual abilities that you expect to find in a sophisticated database system. You can operate on only one file at a time, so the system can't do a relational join. For example, if you have an employee-organization file and a sales file, in most cases you can't use ACCESS to answer queries like, "Who are the supervisors of the salespeople who did not meet their quotas?" An exception occurs when the item IDs in one file are the same as those in another; for example, if both the employee-organization and sales files used the employee ID as their item ID, one query can create an item list and the second query can use it for selection. If you use such queries, you must take considerable care in designing the database to ensure that the item IDs are strategically chosen.

Pick has a rudimentary text formatter called RUNOFF. When used with ED, it makes a good 1960s-style word-processing system.

Finally, the one and only programming language for Pick is PICK/BASIC. This language bears little resemblance to the awkward Microsoft BASIC. It's actually a fine little language.

Among the major features in PICK/BASIC that aren't in Microsoft BASIC are typeless variables, strings up to 32,767 bytes long, subroutines with arguments, built-in database functions, multiline structured control statements, and no line numbers.

You handle a database item by reading it into or writing it from a string variable. You can reference attributes, values, and subvalues by treating the string as a three-dimensional array. For example, if R holds a data item, then R<4,3,2> is a reference to the second subvalue of the third value of the fourth attribute. Such an expression can appear on either side of an assignment statement.

INSTALLING AND USING PICK

Installation of Pick went smoothly. I booted the XT with the first disk, then followed the on-screen directions to load the remaining four disks. I didn't even have to create a partition on my hard disk; Pick automatically found an unused stretch of cylinders and made it the Pick partition. Therefore, if you also want MS-DOS or any other operating system, you have to install Pick last.

Pick is the only multiuser system I have ever seen that tells you how to wire the cable for remote terminals. Not only that, but the wiring diagram was correct. However, the list of supported terminals is short and strange (e.g., the VT-100 isn't listed).

The problem with the installation instructions is that they stop after telling you to insert the last disk. I was presented with a log-in prompt and had no idea how to respond. (I tried MARC and it didn't work.) After searching the manual for 20 minutes, I learned that I should log in as SYSPROG (Pick takes only uppercase). Then I was able to create an account for MARC with the CREATE-ACCOUNT verb.

Next I created a file named PGM and wrote a small PICK/BASIC program (painfully, with ED) into the item TST1. When I tried to compile it with the BASIC command, I received a complaint about an invalid source format. After more searching in the manual, I found a note in the file-structure section to the effect that in order to create a program file, I must use the editor to "change the D-pointer in the master dictionary to a DC-pointer." I had only a fuzzy idea of what this meant, but I was able to use ED to edit the master dictionary, find a D, and change it to a DC. I still don't know what this was all about, but it worked. I think the Pick people should have used their PROC language to supply a procedure called CREATE-PROGRAM-FILE.

In the long run, problems like these don't matter. You get used to them and you forget how silly they are. But they frighten you when you're getting started and need all the confidence you can muster.

You can boot Pick off the XT's hard disk. When it first comes up, it spends several minutes doing something called "verifying system nodes," during which time you can't use the XT's keyboard. However, the remote terminals are activated and you can use them right away. I don't know if modifying the database while it is being checked is safe.

Pick can't read or write MS-DOS floppies, nor can it access the MS-DOS partition on the hard disk. To get to MS-DOS, you have to shut down Pick and boot MS-DOS. To get back, you have to reboot Pick and wait for it to verify those system nodes.

I only used Pick for a few days, and I never tried anything really sophisticated. But it didn't crash, and I didn't find any bugs. My feeling is that it is very solid.

PICK DOCUMENTATION

The Pick documentation is well written and full of helpful examples. From what I could tell, it's also accurate and complete. The manual has a good index, but it's hidden in the system maintenance section where novices are sure not to find it. A tiny tutorial section covers so little of Pick that I didn't find it useful.

The Pick Systems people teach courses that show you how to use Pick effectively. The courses last a week and cost about $900 each. Since Pick is so different from traditional operating systems, these courses might be worthwhile.

PICK PERFORMANCE

To get some feel for Pick's performance, I compared it to Revelation running under MS-DOS. Revelation (a product of Cosmos Inc., 19530 Pacific Highway S, Seattle, WA 98188, (206) 824-9942) is a single-user database system patterned on Pick.

I ran a program that writes 3000 records of 150 bytes each to a file that is initially empty. [*Editor's note: The benchmark programs used in this article are available for downloading from* BYTEnet *Listings. Call* (617) 861-9774 *before* November 1. *Thereafter, call* (617) 861-9764.] Each record has a 20-byte ID. When I created the file, I followed the recommendations in the Pick

(continued)

The important features of Pick are very well designed.

manual and determined that there should be 503 hash buckets. It took Pick 8 minutes and 20 seconds to run this program, and the resulting file consisted of 531,000 bytes.

Revelation ran the identical program (except for a small syntax change to the OPEN statement) in 21 minutes and 58 seconds and created a file of 1,016,832 bytes. This file size includes lots of space for additional records, so it can't be compared to the file size for Pick.

The times don't tell the complete story. During the Pick run, the disk light came on only now and then at the beginning, and then it blinked on and off steadily. During the Revelation run, the light stayed mostly on, and I could hear the disk seeking constantly. My disk had never had such a sustained workout.

Don't interpret these results to mean that Pick is three times faster than Revelation/MS-DOS. They're based on only one program. I'm sure Pick is faster, but exactly how much I don't know.

Also, bear in mind that Revelation has advantages over Pick: It's well integrated into the MS-DOS environment (you can read and write MS-DOS files, and you can execute MS-DOS commands from within Revelation), and many of its subsystems are better designed (there's a screen editor, for instance). It might make sense to develop your application under Revelation and then move it to Pick when it's ready for production use.

CONCLUSIONS ON PICK

Many of the Pick operating system's less important features are badly designed, but its important features are very well designed, particularly its file structures and the PICK/BASIC language. Pick is simple and powerful, and it seems to be efficient and reliable, too. It does exactly what it was designed to do.

Pick merits careful consideration if you are planning to use XTs to run dedicated database applications. Because it works well as a multiuser system, it's probably the most cost-effective way to use an XT.

OVERVIEW OF COHERENT

Coherent appears to be nearly a clone of UNIX Version 7, an older release of UNIX that has since been replaced by System III and System V. I write "appears to be" because the Coherent manual doesn't say it is based on UNIX. The failure to mention UNIX has a practical disadvantage: No advice is given on how to port Coherent programs to the various UNIX versions, something that many Coherent programmers will want to do.

As a UNIX clone, Coherent is amazingly complete. It includes even advanced features like yacc (a parser generator) and awk (a report-generation language), but it lacks many commands that are part of Version 7. Some major commands that are missing include f77 (FORTRAN), bas (BASIC), troff (typesetter formatter), eqn (equation processor), tbl (table formatter), lint (C checker), uucp (file-transfer program), and plot (plotting program). There are also 19 other missing commands.

On the other hand, Coherent includes about 20 commands not present in Version 7, including kermit, which substitutes for cu and uucp, and dos, which allows reading and writing of MS-DOS floppies (but not the MS-DOS hard-disk partition).

Two screen editors, trout and elle, are based on EMACS. The difference between them wasn't clear to me (I used trout), but I was told by a technical-support person at Mark Williams that trout is easier to use and elle is more robust. In my opinion, either editor is far superior to the UNIX editor vi because they avoid the command-mode/insert-mode problems that make vi a pain to use. For UNIX old-timers who want to get started in a hurry, the ed line editor is there, too.

Some Coherent commands have the same name as their UNIX counterparts, but they are not equivalent. For example, the Coherent nroff is much less powerful than the real thing. Of the 77 requests in the Version 7 nroff, only 31 are present in Coherent (the most useful 31, however).

Coherent has all the Version 7 system calls except nice (which sets a process's priority), and they seem to be used in the same way. It should be easy to port C programs between Coherent and UNIX Version 7.

INSTALLING AND USING COHERENT

Coherent's installation procedure is much less automated than Pick's. I was asked to make lots of decisions about the sizes of file systems without knowing exactly what the impact of my decisions would be. I was warned that the root partition of my hard disk would be overwritten, but the manual didn't define that term.

Information about leaving disk space for an MS-DOS partition is at the end of a rather long installation section in the manual, where you might not see it until too late. You must install Coherent first, leaving some space for MS-DOS, and then use the MS-DOS FDISK command to create the MS-DOS partition. If you already have MS-DOS installed, you have to calculate cylinder numbers carefully when you install Coherent. You can't boot Coherent off the hard disk; you need to use a boot floppy.

I doubt that most people who aren't XT and MS-DOS experts will be able to successfully install Coherent along with MS-DOS. But if you don't care about MS-DOS, you can just do the installation blindly and Coherent will take over the whole disk with suitable default values for the various file systems. The installation instructions don't tell you how to wire a cable for a terminal, but three wires seem to be enough (pin 1 straight through and 2 crossed with 3). Only VT-52 and Z-19 terminals are supported. There is no terminal-capabilities facility to support more terminals.

Although Coherent is a multiuser system, you might not want to use it that way, at least for program development. The XT has no memory protection, and it's easy for one user to bring down the whole system. However, if you're running only debugged programs, multiuser access should be safe enough.

You have to be careful when shutting down Coherent. You must kill the init process and then issue a sync command to flush the buffers to disk. Coherent has no shutdown command, as do many other UNIX implementations, but you can write your own.

I could read MS-DOS files off a floppy disk easily enough once I figured out the proper name for the floppy device file. There are different names for single- and double-sided disks, and for 8 and 9 sectors per track. These names are not given in the dos command write-up, but in a separate write-up for fd.

If you get stuck—and you probably will—you can call a toll-free number to get help. The support people were too busy to talk to me when I called, but they called back within a few hours and the person who called knew what he was talking about.

After I installed Coherent, I played around with it awhile and became convinced that it's close enough to UNIX to qualify as a clone. If you sat a UNIX expert down at the keyboard without telling him or her that Coherent was running, he or she would think it was the real thing.

Coherent didn't crash during the few days I used it, but I did have trouble accessing it from a terminal. The terminal would lock up after a while. From the console, I logged in as a superuser and killed the program I was running at the terminal; then the terminal came alive again. However, one time the shell was running at the terminal and I couldn't kill it (I sent it a true kill signal, not a software termination signal). When you can't kill a process, it means the kernel has a bug. I don't think the problem was with my hardware, because I used the same terminal with Pick for many hours with no problems at all.

Coherent Documentation

Coherent comes with two fat binders full of beautifully typeset UNIX-style documentation. There are individual manual pages for commands, subroutines, system calls, and device files. Several major subsystems (e.g., nroff and trout) also have their own manuals.

However, there is no manual on C and hence no information about sizes of types, signed/unsigned arithmetic, register variables, and assembler interfacing. A separate MWC86 user's manual is referenced, but it isn't supplied with Coherent.

Most manual sections have separate indexes, and an index also covers the individual manual pages. The manual seems reasonably complete, accurate, and well written, but it would be unintelligible to anyone who doesn't already know UNIX. Fortunately, many textbooks on UNIX can fill that gap.

Coherent Performance

I ran five benchmarks to compare the efficiency of Coherent to PC/IX (a UNIX System III product from IBM). Some of these benchmarks were also run on MS-DOS and THEOS (see table 2).

I had a problem with the program I used for benchmarks 4 and 5. This is a 2000-line, three-file C program that implements a B-tree access method. The Coherent C compiler failed to compile two of the three source files. These files had previously compiled successfully on five different C compilers or computers ranging from the XT to the VAX-11/780. One file caused a fatal compiler error with the message "no match, op = 65" and a dozen or so lines of debugging information that looked like a parse tree, one of the more interesting error messages I've seen. The other file caused the fatal error "more than 20 stores."

I called the technical-support number and was told that in the first case a type cast I was using wasn't handled by the compiler, and in the second case the compiler ran out of registers. After changing the source a little, I was able to compile the files, and the resulting program ran correctly without further incident.

Since most users of Coherent are likely to want it for C program development, the compiler's inability to handle perfectly legal C programs is

(continued)

Table 2: *Benchmarks comparing* THEOS, *Coherent*, PC/IX (UNIX *System* III), *and* MS-DOS. *Times are in minutes and seconds. User time is the* CPU *(central processing unit) time spent executing instructions in the program itself; system time is the* CPU *time spent executing instructions in the operating system kernel on behalf of the program; real time is the total elapsed (wall clock) time.*

Test		THEOS	Coherent	PC/IX	MS-DOS
Shell program	User	—	19:24	3:58	—
(benchmark 1)	System	—	3:25	1:24	—
	Real	—	28:33	9:22	—
Random I/O	User	—	1:44	0:39	—
(benchmark 2)	System	—	0:20	1:28	—
	Real	2:55	2:35	2:28	3:24
Sequential I/O	User	—	4:42	1:40	—
(benchmark 3)	System	—	1:05	0:59	—
	Real	8:57	7:22	3:20	7:07
C compile and	User	—	3:52	6:32	—
link (benchmark 4)	System	—	0:32	0:33	—
	Real	—	5:20	8:05	7:18
B-tree install	User	—	0:52	0:43	—
and fetch	System	—	0:19	0:20	—
(benchmark 5)	Real	—	1:43	1:27	2:47

Inquiry 171

IBM / PC

CROSS ASSEMBLERS

We've been selling these industrial-quality assemblers to the development system market since 1978. They are now available for the IBM PC.

FEATURES:

- ***Fully relocatable***
- ***Separate code, data, stack, memory segments***
- ***Linker included***
- ***Librarian included***
- ***Generate appropriate HEX or S-record formatted object file***
- ***Macro capability***
- ***CPM80, MPM, ISIS versions available***
- ***Conditional assembly***
- ***Cross reference***
- ***Supports manufacturer's mnemonics***
- ***Expanded list of directives***
- ***1 year free update***

Assemblers now available include:

Chip	Chip
1802/1805	NSC800
8051	F8, 3870
6500/01/02	Z8
6800/01/02	Z80
6803/08	9900/9995
6804	Z8000
6805	68000
6809	6301
6811	8048/49/50/42
8085	65C02/C102/C112

Take advantage of leading-edge technology. Get your own Relms assembler today. Use your Mastercard or order by phone: (408) 265-5411

Relational Memory Systems, Inc.
P.O. Box 6719
San Jose, California 95150
Telex: 171618

Prices subject to change without notice. Software distributor inquiries invited.

relms®

Coherent is based on an obsolete version of UNIX, *but the concepts are the same.*

disturbing. You can expect to have to mess around a little with C programs to get them to run. The technical-support people were familiar with the problems I was having, so perhaps the compiler will be improved in a later release.

For the shell program (benchmark 1), Coherent is much slower than PC/IX. I don't know why this is. The Coherent C compiler and linker was much faster than either the PC/IX compiler and linker or the Lattice C compiler and the MS-DOS linker, but since these compilers are different programs that produce different output, the time differences don't necessarily say anything about the speeds of the operating systems. Coherent is nearly as fast as PC/IX for the random I/O and B-tree tests (benchmarks 2 and 5), but these times are governed more by the hard disk than by the operating system, which can't help much on random I/O.

Clearly, Coherent is slower than PC/IX, although the difference varies and might be insignificant for your application. If you're doing program development, the fast Coherent C compiler will be a boon.

Conclusions on Coherent

If you want to learn UNIX and you have some space on your XT, you should consider buying Coherent. The price of $495 is a bargain. Many of the details are different and it's based on an obsolete version of UNIX, but the concepts—the hardest part to learn—are the same.

Coherent is probably a better C program-development environment than MS-DOS, but its C compiler needs improvement. Coherent is a weaker development environment than a complete UNIX system because it lacks tools such as lint and the Source Code Control System (SCCS).

Coherent costs half as much as PC/IX and requires less memory. Although it is less complete and runs more slowly, it's a good buy.

Overview of THEOS

Oasis8 was probably the most sophisticated operating system for Z80-based microcomputers, light-years ahead of the much better known CP/M. THEOS is an extension of Oasis8 for 8088/8086-based machines. It supports more users, more tasks, more RAM, and more disks, and it has more commands.

Unfortunately, while THEOS offers many advantages over MS-DOS, it also has to compete with operating systems moved to the PC from the other direction: minicomputers. Examples are Pick, Coherent, and, of course, UNIX. Roughly speaking, the functionality of THEOS is somewhere between MS-DOS and UNIX.

The THEOS core system consists only of what can be properly called the operating system: the kernel and utilities for configuring the system, managing the printer, manipulating files, editing text, and executing commands. Programming languages, word processors, and database systems are extra.

Each THEOS user is assigned a fixed memory partition up to 64K bytes in size. A program's data space is limited to what the partition can hold, but the instructions are located elsewhere. If two or more users are running the same program—a common occurrence for multiuser applications—they share the instructions.

A user might be running a multitasking program that consists of a main task and one or more subtasks. The tasks can communicate through shared variables and they can coordinate their access via semaphores. There are no pipes or messages.

Multitasking in THEOS is fairly restricted. A single user can't run a background task that is unrelated to a foreground task. For example, you can't compile some programs in the background while you write a report in the foreground. To do that, you need two terminals so you can pretend to be two users. Another restriction is that one user's task can't communicate with another user's task. And the command interpreter doesn't provide for multitasking at all (via a pipe operator like ¦); you set up multiple tasks only from BASIC or C.

The multitasking printer spooling system seems to be an exception, but it isn't really. It actually runs as a separate user.

The THEOS file system is a three-level hierarchy, but it's less general than that of MS-DOS or UNIX. A filename can consist of three components separated by periods: the filename, the file type, and the library member name. Each component can be up to eight characters long. Although there is nothing that corresponds to current directory, you can set various default libraries, such as link library and macro library. If you edit a file named, for example, Memo, the editor treats it as a member of the default macro library. If you want it to be a file in its own right, you have to use periods in the name.

Each file has an owner. The owner can set the file to be accessible by everyone or only by the owner. Since you need a password to log in, this system is reasonably secure. There are separate permissions for erase, read, write, and execute.

In addition to sequential and random I/O, the file system also supports a keyed access method. It has a mechanism for record locking, but a user can lock only one record at a time in a given file. This is inappropriate for transaction processing, where you want to keep all your records locked until the transaction completes. Also, THEOS doesn't provide other facilities needed for serious database management, such as automatic rollback of aborted transactions. Deadlock is possible, but I couldn't find any mention of it in the manual. I assume that the operating system doesn't detect it, let alone resolve it.

(continued)

LETTRIX™ Improves PC Printing

With LETTRIX™ your IBM ProPrinter, Epson, Okidata, IBM Graphics, Star Gemini, or Panasonic printer can print proportionally-spaced and microjustified **letter-quality** text *directly from all software*.

Before: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef
After: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef

LETTRIX™ resides in your PC's memory between your software and printer, responding automatically to spacing, paging, margins, **boldfacing**, underlining, *italics*, $_{\text{sub}}$scripts and $^{\text{super}}$scripts. Just print as you would normally... LETTRIX takes care of the rest!

"The top of its class." -- PC PRODUCTS MAGAZINE
"A fun program that is easy to use." -- PC WEEK
"The sharpness and clarity nearly equals that of an expensive daisy wheel printer." -- THE NEW YORK TIMES
"Surprisingly low price." -- BOARDROOM REPORTS

You can design your own characters on any display, or use the twenty different typefaces included: **Broadway**, Courier, **Old English**, Art Deco, OUTLINE, ABCEBLOCK, **Engraved**, *Park Avenue*, WESTERN, SHADOW, **Roman**, BANKER, Gothic, Prestige, Folio, ORATOR, OCR-A, and the entire IBM foreign language and line-drawing character set:

$$\sum_{n=1}^{k} \alpha\phi\pi^2\theta \; \frac{(\mu+\beta)}{6.718} \approx \int_{1}^{\infty} \left[\frac{\partial^3 u}{\partial t^3} + u\,\frac{\partial u}{\partial x} \right] + \frac{Ax\cdot\delta}{\sigma\mu/\Omega}$$

Français, Español, Italiano, Deutsch, Ру́сский, Ἑλληνικός, אני אוהב להשתמש בלטריקס כי לטריקס יפה.

You can control LETTRIX features either by accessing a menu that pops-up over your current software, or by placing simple single-character commands in your text.

For fastest delivery, charge LETTRIX to your VISA or MasterCard by dialing (800) 351-4500. LETTRIX™ costs just $98.50. We pay UPS shipping.

Call and mention this magazine for a free print sample. LETTRIX requires an IBM PC, XT, AT, or compatible with 50kb of free memory, and DOS version 2.0 or greater.

This entire advertisement is an actual-size, unretouched column of text printed directly from a wordprocessing program on an EPSON FX printer controlled by LETTRIX™.

HAMMERLAB CORPORATION
(203) 776-5699
5700 ARLINGTON AVENUE
RIVERDALE, NEW YORK 10471

Inquiry 79

THEOS is *easier to use than* UNIX *or Coherent once an expert has set up the system.*

THEOS can't access MS-DOS files, although it can access Oasis8 files. The text editor is a combination screen editor and line editor, similar in concept to vi and ex. You spend most of your time in the screen editor part, but you do operations such as global changes in the line editor part. You can use many of the special keys on the IBM PC XT keyboard (such as the arrow keys), so I had an easy time learning and using the editor.

The C compiler is called Definitive C and is described in its manual as compatible with UNIX Systems III and V. However, I didn't find this to be true. The compiler failed to handle the types enum and void, and the library functions worked differently (for example, there is no fseek function). Also, you can't include a file named, for example, stdio.h, because a member of a macro library can't have a period in it. You have to use stdio instead.

Aside from these irritating incompatibilities, the compiler worked. One nice touch is that it pauses after displaying an error message so you can read the message when you get back from the refrigerator. Then you can kill the compilation if you think you've seen enough. The linker is also special: It flashes the object module names on the screen as it finds them in the library. Niceties like these might not be important, but they conveyed a feeling of quality and attention to detail that enhanced my confidence in the system.

A regular BASIC and a graphics VBASIC are available at extra cost. VBASIC uses a virtual-device interface to support a variety of output devices. I did not evaluate either BASIC in detail.

INSTALLING AND USING THEOS

I made a mistake in installing THEOS: I read the one-page installation instructions, followed them, and then got stuck. Although the system was installed, I didn't have the foggiest idea how to use it.

What I should have done is spend several hours studying the 58-page introductory manual first. Nuggets of essential information are spread throughout this manual, and you really need to know about them before you can get THEOS up and running.

For example, to get the printer working you have to know that its physical name is centlp. The manual doesn't tell you this, but it does tell you where to find a file that lists the physical names. Then you need to know how the ATTACH command works. There's a section titled "Attaching Printer Devices," but it only discusses general principles. You are directed instead to the chapter titled "Using the Printer Spooler." There isn't any chapter with that title, but there is one called "Using a Printer." After more chasing around, you finally realize that it would have been much easier to read the manual sequentially from the beginning, memorizing as much as possible. Then at least you would know where to find information.

You can boot THEOS from the hard disk, but it was difficult to figure out from the manual how to do it. You use a disk utility with an option titled "write boot sectors."

It was also very difficult to figure out how to install a second terminal, but once I did, the actual installation went smoothly. Again, if I were an expert on physical and logical devices, attaches, and class codes, it would have been a breeze. There was no wiring diagram for the serial cable, but the same cable I used for Coherent worked.

Ignoring the documentation problems, I would say that THEOS is easier to use than UNIX or Coherent once an expert has set up the system. This is partially because THEOS has fewer commands, with fewer options on each command, and partially because of more use of menus and prompts. THEOS doesn't try to be as clever as UNIX does.

THEOS DOCUMENTATION

The sentences and paragraphs in the documentation are written well enough, but there are major problems with the manual's organization. First, it has too little redundancy. For example, instead of listing the terminal class codes in the subsection titled "Console Class Codes" in the section titled "Attaching Console Terminals," the manual refers you to the chapter titled "Console Terminal Class Codes." No such chapter exists. I experienced far too many wild-goose chases to track down cross-references.

The manual is also too verbose. Long treatises on the philosophy of multitasking are mixed in with essential installation information. Since the cross-references are messed up and there's no index, you have to read the manual sequentially. This takes twice as long as it should because you have to wade through lots of stuff that you could have read later at your leisure.

Finally, the target audience for the manual is ill-defined. One paragraph discusses head load delay for a hard disk, while another explains what is meant by the word "abbreviation," complete with an example. Not only that, but the explanation of head load delay precedes the explanation of how to install a printer.

In summary, the manual does not present information in a logical order, has too many cross-references (many of them erroneous), mixes nonessential and essential information, and has no index. But after reading the whole manual, I was able to understand how THEOS works and I had very little trouble installing and using it.

THEOS PERFORMANCE

I didn't run the B-tree program (benchmark 5) on THEOS, because it couldn't read the source code from my MS-DOS floppy, and I didn't have time to work out a serial port connection to another computer. Instead, I tested just the random I/O and sequential I/O programs.

Based on these tests, it seems that random I/O is faster than MS-DOS but slightly slower than Coherent and PC/IX. In sequential I/O, THEOS was the slowest of the four operating systems by a wide margin. It's difficult to extrapolate from just two programs, but it's safe to say that the THEOS file system is in the same ballpark as MS-DOS, which is slow. Since THEOS has indexed access methods built in, they are likely to be fast, although I ran no test to prove this.

CONCLUSIONS ON THEOS

THEOS is well designed and worked the way it should. It's easier to use and understand than UNIX, but it also has many fewer features. Of course, if you are implementing a specific application, you don't care about the number of features; you care about whether the features you need are present. You might well find that THEOS is a perfect match for your requirements.

Compared to other operating systems for the XT, the price for THEOS seems too high. It costs twice as much as Pick or Coherent, and, with the C compiler, it costs even more than PC/IX. ■

A PROTOTYPE FOR YOUR PROTOTYPES

Methods

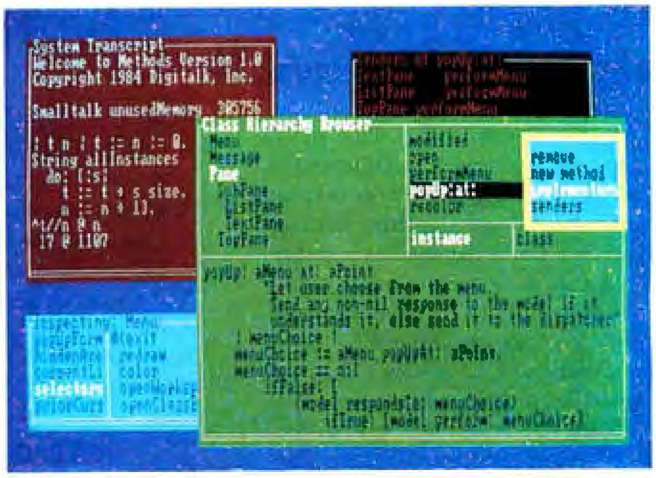
METHODS BY DIGITALK. A SMALLTALK PROGRAM DEVELOPMENT ENVIRONMENT FOR THE IBM PC. AN OBJECT-ORIENTED SPEED DEMON. AN EXCITING WAY TO PROGRAM.

Think about your problem. Browse the Methods Smalltalk source code. Select some building blocks like pop-up menus, windows, text-editors and dictionaries. Put them together. Try it. Oops, Methods tells you something is missing. No problem. Continue thinking, changing and trying. Now you're prototyping! Try out new ideas. Redefine your problem. And you don't have to throw this prototype away. Refine it until you like the finished product.

Methods by Digitalk. A new way to develop software for the PC. Use it for windowing, simulation and artificial intelligence applications. Use it by prototyping.

Methods is Smalltalk-80™ language compatible. It includes its own Smalltalk source code. You can extend it in Smalltalk and assembly language.

Methods operates on IBM PCs with 512K bytes RAM using MS-DOS or PC-DOS. Color and monochrome monitors are supported. No mouse is required. A Smalltalk language manual and an environment guide are included.



```
System Transcript
Welcome to Methods Version 1.0
Copyright 1984 Digitalk, Inc.

Smalltalk unusedMemory 285756

| t n | t := n := 0.
String allInstances
  do: [:s|
    t := t + s size.
    n := n + 1].
^t//n @ n
 17 @ 1107
```

```
popUp: aMenu at: aPoint
    "Let user choose from the menu.
     Send any non-nil response to the model if it
     understands it, else send it to the dispatcher"
  | menuChoice |
  menuChoice := aMenu popUpAt: aPoint.
  menuChoice == nil
     ifFalse: [
        (model respondsTo: menuChoice)
           ifTrue: [model perform: menuChoice]
```

Available from Digitalk for $250. Outside U.S. add $15.00 for shipping and handling. California residents add 6% sales tax. Visa and MasterCard accepted. Educational and Dealer Discounts Available.

DIGITALK, INC.
5200 West Century Boulevard
Los Angeles, California 90045
(213) 645-1082

Smalltalk-80™ is a trademark of Xerox Corporation. IBM is a registered trademark of International Business Machines Corporation. MS is a trademark of Microsoft Corporation.