Volume 2, Number 9 November 1984

IBM Personal Computer Seminar Proceedings

The Publication for Independent Developers
of Products
for IBM Personal Computers

Published by International Business Machines Corporation
Entry Systems Division

IBM®

Changes are made periodically to the information herein; any such changes will be reported in subsequent Proceedings.

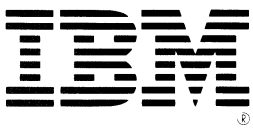
It is possible that this material may contain reference to, or information about IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming or services in your country.

IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM makes no warranty of any kind with respect to the accuracy or adequacy of the contents hereof.

This publication could contain technical inaccuracies or typographical errors. Also, illustrations contained herein may show prototype equipment. Your system configuration may differ slightly. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

All specifications are subject to change without notice.

Printed in the United States of America



Contents

Introduction and Welcome

These are the Proceedings of the IBM Personal Computer Seminar, designed for independent developers of products for IBM Personal Computers. The purpose of these Proceedings is to aid you in your development efforts by providing relevant information about new product announcements and enhancements to existing products. This issue is prepared in conjunction with this seminar. The Proceedings of future siminars for the IBM Personal Computers also will be published and will cover topics presented at those seminars.

Throughout these Proceedings, the term Personal Computer and the term family of IBM Personal Computers address the IBM Personal Computer, the IBM Personal Computer XT, the IBM PC*jr*, the IBM Portable Personal Computer, and the IBM Personal Computer AT.

Purpose

What is our purpose in putting out a publication such as this? It is quite simple.

The IBM Personal Computer family is a resounding success. We've had a lot of help in achieving this success, and much of it came from the independent developers.

As you proceed with your development, do you at times wish for some bit of information or direction which would make the job easier? Information which IBM can provide? This is the type of information we want to make available to you.

Since we want to be assured of giving you the information you need, we ask you to complete the questionnaire which appears at the end of these Proceedings. Your response to this questionnaire will be taken into account in preparing the content of future issues, as well as the content of seminars we will present at microcomputer industry trade shows.

Topics

The following list gives a general indication of the topics we plan to cover in future seminars and include in the IBM Personal Computer Seminar proceedings:

- Information exchange forum — letters to the editor format
- Development tools — languages, database offerings
- Compatibility issues
- New devices — capacities and speeds
- System capacities — disk and memory
- Enhancements in maintenance releases
- Tips and techniques
- New system software
- Hardware design parameters
- Tips on organizing and writing documents for clear and easy reading
- Changes to terms and conditions

IBM PC XENIX

Overview

The IBM PC XENIX (XENIX) is an operating system for the IBM Personal Computer AT utilizing the features of the 80286 processor. It was developed by Microsoft Corporation. XENIX[2] is a multi-tasking, multi-user system based on the UNIX[1] System III by AT&T Bell Laboratories. Up to three users can use the system at the same time: two users can access XENIX through ASCII terminals or PC's and the third user accesses XENIX using the system console.

XENIX includes enhancements by Microsoft Corporation and the University of California at Berkeley. These topics will be addressed later.

The XENIX system is comprised of three separate parts. The Operating System is the base package. It requires at least a IBM Personal Computer AT with 512KB of memory, 20MB fixed disk drive, a 1.2MB diskette drive, and a monitor. The Operating System is a pre-requisite for the Text Formatting and Software Development Systems.

The Text Formatting System provides high level formatting macros for document preparation in conjunction with nroff (a formatter for printers) and troff (a formatter for phototypesetters). Additional formatting facilities are also provided.

The Software Development System contains a large set of tools including the C programming language. Together, the C compiler, assembler, debugger, productivity tools and other programming utilities provide a complete environment for the programmer.

The Operating System is composed of many parts. One of the parts is the XENIX kernel. The kernel is the central part of the XENIX operating system. It controls all the resources of the computer and allocates them to the active processes. It manages the file system, automatically assigning disk space to files and directories as they are created, enlarged, or deleted, returning no longer used space to the "free list." The kernel takes advantage of the protected mode of the IBM Personal Computer AT 80286 processor to protect itself from the users and to protect the users from each other. Additionally, each user is granted access only to those files and directories that he is authorized to use.

Over 150 commands are available with the Operating System. They include two editors: vi, and ed. Ed is a line editor that is available in all UNIX systems. Vi is a full screen editor and other commands copy files and directories, remove files and directories, back up and restore the file system, etc.

The Bourne Shell is a language interpreter that serves as the interface between the user and the rest of the system. It serves as both an interpreter and a programming language. The shell permits the creation of new commands simply by writing shell procedures (also called scripts). The shell language contains flow-control constructs such as while, if-else, for, do, and case. It also allows definition of variables, parameter passing, and interrupt processing.

XENIX provides the capability to transfer files between IBM DOS diskettes and the XENIX file system. This capability is discussed later with other IBM/Microsoft enhancements.

The Text Formatting System is composed of a powerful collection of tools that complement writing productivity. In combination with any of the system's text editors, the text formatting system can be especially valuable in simplifying the generation of technical reports, formal papers, and other professional quality documents.

The main components of the Text Formatting System are the nroff and troff formatters. The nroff formatter produces output for matrix line printer devices. The troff formatter, compatible with nroff, offers more sophisticated capabilities because it supports phototypesetters. For example, while nroff will use underlining, approximate spacing, and text size determined by the characteristics of the printer used, troff supports italics, variable spacing, and point size.

Additional formatting facilities may be used in conjunction with nroff and troff for building sophisticated mathematical equations and for designing complex tables. The eqn preprocessor can format complicated mathematical symbols and equations using commands which resemble English words. The tbl preprocessor gives control over material that must appear in tabular form, and it automatically calculates the information necessary to line up complicated columnar data even if elements have varying widths. It also centers tables, expands a table to the width of the current line, and encloses a table in boxes.

[1]UNIX is a trademark of AT&T Bell Laboratories
[2]XENIX is a trademark of Microsoft Corporation

Another series of text formatting utilities to analyze writing style and correctness are diction, style, and explain. The style utility analyzes writing style: it reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers. The diction utility finds all sentences in a document that contain phrases from a data base containing bad or wordy diction. The explain utility interactively suggests alternatives to phrases flagged by diction. The spell utility finds misspelled words in a file.

The XENIX Software Development System provides a large set of tools. It includes a C compiler, assembler, debugger, and productivity tools like "make" and "SCCS".

C is a general-purpose programming language. Most of the software in the XENIX system is written in the C programming language.

The C compiler can create programs for three different types of memory models: small, middle, and large. Each type of model is distinguished by its use of available memory space and how it uses program and data in that space.

Small model programs can be pure or impure in regard to space usage. Impure-text small programs occupy one 64K-byte physical segment. The program and data areas are combined in one segment. Pure-text model occupy two 64K-byte physical segments. One segment is used for data. The other segment is used for text (program) and is read-only. This means a pure-text program can be shared by several processes at a time (only one copy is loaded in memory). The total program size for a pure-text small model is 128K-bytes.

Middle-model and large-model programs generate pure-text programs.

Middle-model programs use one segment for data and as many segments as necessary for the program. There is no limit on the program (text) size. The only limitation is that the space required for variables (data) does not exceed 64K-bytes.

Large-model programs use as many segments as necessary for data and programs.

A significant feature of the IBM PC XENIX C compiler is DOS cross-compilation. This feature lets the programmer create the code to be run under IBM DOS (version 2.00 or later). The programmer creates and debugs C programs using the productivity tools available in XENIX, then recompiles the programs for DOS. The DOS file transfer utilities can then be used to copy the load module to a DOS diskette.

Sometimes a programmer determines that certain tasks cannot be done in C or will be better performed when written in assembler. The XENIX Software Development provides an assembler for 80286/80287, 8088/8087, 80186, and 8086. It supports an assembler similar to the IBM DOS assembler except for macro support.

Medium to large programming projects tend to have multiple source files. There may be too many dependencies between files to be remembered at all times. A modification in one file might affect other files. The command "make" is a program maintainer. It reads a file supplied by the programmer which specifies the file dependencies in the project. Usually a programmer specifies that a load module depends on some object modules, and each object module depends on a source file. If a source file has changed (its date is newer than the date for the object file) it causes the source file to be recompiled. The new object file is then linked with other object files to produce a new load module. This way the programmer does not have to remember what files have to be recompiled and relinked when a change occurs.

SCCS controls the maintenance of text files. The text can be any text or source code. The SCCS process is started by running the SCCS admin command on the files containing the text. SCCS provides other commands to retrieve and update the files, add new releases, and recreate a complete source of a release at any point in time. SCCS saves the original files and maintains a list of modifications (called deltas) applied to each of the files. To obtain a specific release SCCS starts with the original file and applies the correct deltas to the files involved.

The XENIX Software Development System contains additional commands like an archiver, a C beautifier, lex - a lexical analyzer, preprocessors, and a compiler-compiler.

One of the most widely voiced concerns about UNIX is the difficulty it can have with the file recovery after a power failure. In improving the UNIX file system, Microsoft has put in several features and utilities as well as general design modifications to allow the system to recover the file system automatically from a crash, performing the functions that UNIX's designers assumed would be handled by a systems programmer (The XENIX Operating System Abstract, Microsoft, 1982).

Multiple processes may access a file for reading and writing. When a process is reading information from the file we do not want that information to change until the reading has ended. When a process is writing to a file we do not want other processes accessing the information until the writing has been completed. It becomes necessary to lock different parts of a file at different times. Locking a file is a way to synchronize file use when several processes may require access to a single file. One or more bytes can be locked in any region of a file. If you are writing to a file you can lock all other users out of the area you are writing. Since you locked only a specific region of a file the rest of the file can be accessed or locked by other users. You only lock a region for as long as you need it.

The XENIX system provides a way to control the access of its resources. It is called semaphores. Semaphores are named like files and are created by a process that synchronizes the access of other processes to the resource. The control of a semaphore passes from process to process. The process that controls the semaphore has control of the system resource. A process maintain control of a semaphore until it relinquishes it. Another process that needs to access the resource waits for the resource until it gains control of the semaphore.

Shared data segments (also called shared memory segments) are used to improve the speed and facilitate communication among processes. It works like a temporary file, except it is kept in memory. A process can create a shared memory segment. Other processes can request, wait if necessary, and gain access to a shared segment. Once a process has finished accessing a segment it signals other processes. Shared data allows processes to pool information without creating temporary files. The process that created the shared data segment may allow simultaneous access to the segment by multiple processes and can free the segment when the space is no longer needed.

There are some new system administration commands that are unique to XENIX. These are super-user commands that facilitate the maintenance of the users' logins. The system administrator or super-user is the only user permitted to use these commands. The mkuser command is used to add a user login to the system. It handles all directory creation, shell assignment, and password updating. When the super-user creates a user login, he assigns one of the shells (Bourne shell, C shell, or Visual Shell) as the default shell. The rmuser command does the opposite. It makes sure the user's mail box is empty and that all files belonging to that user have been deleted. Rmuser then deletes the user's name from the password file and removes his home directory.

The chsh command allows the super-user to change a user's default shell.

The XENIX Visual Shell is a full screen shell interface that optionally replaces the Bourne shell as the command interpreter for inexperienced users. It is recommended for casual users of the system and those with specialized needs. The Visual Shell replaces sometimes cryptic operating system command with a visual menu of the most commonly executed utilities and provides help information for each one. The user can select a file for use with a command simply by pointing to it via cursor movement keys. Another major feature of the Visual Shell, especially valuable to software developers, is the ability to easily customize the shell. By developing appropriate shell scripts or programs and changing the words on the menu, the application developers can replace complex data processing tasks with simple interfaces requiring minimal computer knowledge.

XENIX provides the capability to transfer files between IBM Personal Computer DOS diskettes and the XENIX file system. There are several utilities to access files and directories on DOS diskettes. The commands have a "dos" prefix. They include support to copy files, list files and directories, erase files, and create and delete directories.

One of the commands is doscp. It copies files between a DOS diskette and XENIX file system. Doscp takes two arguments, file1 and file2, and copies the content of file1 to file2. Either filename can be a DOS file by using the form

```
device:file
```

where "device" is the path name of the special device containing the DOS file.

A simplified way to specify the DOS device is provided in XENIX via a set of capital letters that represent the DOS devices. For example, if a DOS diskette is 48 TPI and in drive A, to copy the DOS file "file1" to the XENIX file system and keep the same filename, the syntax is:

```
doscp A:file1 file1
```

To copy file1 from the XENIX file system to a 48 TPI DOS diskette in drive B, and rename it to file2, the syntax is:

```
doscp file1 B:file2
```

Along with the Microsoft enhancements, IBM PC XENIX includes many popular University of California at Berkeley extensions. Among the most significant are the C shell, vi, and termcap/curses.

The C shell (csh) is a command language interpreter. Like the standard IBM PC XENIX Bourne shell (sh), csh is an interface between the user and the system translating command lines typed at a terminal into corresponding system actions. It is packaged as part of the IBM PC XENIX Operating System.

The C shell offers features above those provided by the Bourne shell. The C shell, for instance, can maintain a history list into which it places the text of previously entered commands. Using a special notation you can reuse commands or words from previously entered commands to form new commands. This mechanism is useful for repeating previous commands or correcting typing mistakes.

The alias facility simplifies commands, supplies default arguments, and performs transformations on commands and their arguments. This mechanism enables the user to substitute a personally preferred name for a system command, for a system command and its arguments, or even to represent multiple commands or pipelines.

Also provided by the C shell are control structures similar to those of the C language, as well as features to help trace its actions, making the C shell especially friendly for users who want to write shell procedures.

The vi editor is a full screen text editor based on a set of mnemonic commands. Most commands are single keystrokes that perform editing functions. The vi editor combines line and screen oriented features to increase editing productivity. It is packaged as part of the IBM PC XENIX Operating System.

The vi editor's capabilities range from inserting characters to deleting lines, from searching for a character to replacing a string wherever it is found in a file, and from splitting a line to rejoining them. Cursor control permits movement in any direction, to any line and to a word, sentence, line, and/or paragraph boundary.

Vi permits multiple files to be edited simultaneously, and permits users to temporarily exit the editor to execute any system command. This can be especially helpful when you are writing or debugging programs or shell scripts.

Vi also supports a series of options to tailor its execution to your individual needs. These options include features to display hidden and end of line characters in files, set tabs, display line numbers, and prevent messages from other users reaching your screen. Additionally, vi allows you to predefine these options in a file so that they are automatically set each time vi is invoked.

Another system feature developed at Berkeley is termcap. Termcap is actually a data base describing terminals. It is used by programs such as vi to interface with a user's terminal. Termcap in IBM PC XENIX supports a large number of terminals, including terminals manufactured by vendors other than IBM. The user can also add support for new terminals as they become available. Termcap is packaged as part of the IBM PC XENIX Operating System. Some of the terminal features that are described in a termcap data base entry include the number of lines, the number of columns on each line, how the screen is cleared, backspace capability, cursor motions, highlighting, underlining, and insert/delete character keys.

As a complement to termcap, curses is a screen updating and cursor movement set of library functions that are packaged as part of the IBM PC XENIX Software Development System. Curses relies on the information in the termcap data base to tailor terminal commands allowing you to write terminal independent programs. With these functions, you have a simple and efficient way to use the capabilities of the terminal attached to your program's standard input and output files.

Next, we will briefly compare XENIX with IBM PC DOS for the IBM Personal Computer AT.

The IBM PC DOS is a single tasking, single user system that operates in real mode. XENIX, on the other hand, is a multi-tasking, multi-user operating system that runs in protected mode. The protected mode feature isolates the users from the kernel and the users from each other: that is, each process has its own space and consequently it cannot adversely affect the system.

IBM PC DOS supports memory sizes up to 640KB, while IBM PC XENIX fully supports the 3MB of RAM available on the IBM Personal Computer AT. It architecturally supports up to 16MB.

The complete XENIX system (Operating System, Software Development System, and Text Formatting System) has over 200 commands and 150 library functions while DOS includes about 50 commands.

Another point of contrast between XENIX and DOS is the number of products that are integral to the XENIX system but are available as separate products under DOS.

For example, the command cu allows a XENIX system to communicate with another XENIX system. In DOS it becomes necessary to acquire a separate product: pcm, or async 2.0, or a 3101 emulator. The Operating System in XENIX provides commands for asynchronous networking: uucp and micnet. Similar facilities in DOS are provided by cluster and PC-Network products.

The full screen editor vi is part of the XENIX Operating System. Similar editors in DOS, like the personal editor or the professional editor, are separate products in DOS.

The Text Formatting Package in XENIX includes nroff and troff. These are facilities to format text files which are somewhat similar to those available as DOS products: Script/PC, Easywriter, PeachText.

The Software Development package provides a large number of utilities. It includes a debugger, adb. In DOS the same function is performed by the Professional Debugger. There are utilities in XENIX, like Source Code Control System and Make (a program maintainer), which do not have counterparts in DOS, even as separate products.

UNIX/XENIX Comparison

The following chart is intended as a guide in comparing UNIX System III, and IBM PC XENIX function. Although it is not complete, it is hoped that the chart is helpful in understanding the product differences. Note that "III" is AT&T's UNIX System III and "XENIX" is Microsoft Xenix for the IBM Personal Computer AT.

	III	XENIX		III	XENIX
300	X		COL	X	X
4014	X		COMB	X	X
450	X		COMM	X	X
ACCT	X		CONFIG	X	
ACCTCMS	X		COPY		X
ACCTCOM	X	X	CP	X	X
ACCTCON	X		CPIO	X	X
ACCTMERG	X		CRASH	X	
ACCTON		X	CREF	X	X
ACCTPRC	X		CRON	X	X
ACCTSH	X		CRYPT	X	
ADB	X	X	CSH		X
ADMIN	X		CSPLIT	X	X
AR	X	X	CT	X	
ARCV	X		CTAGS		X
AS	X	X	CU	X	X
ASKTIME		X	CUT	X	X
ASSIGN		X	CW	X	X
AT	X	X	DATE	X	X
ATQ		X	DC	X	X
ATRM		X	DD	X	X
AWK	X	X	DEASSIGN		X
BACKUP		X	DELTA	X	X
BANNER	X	X	DEROFF	X	X
BASENAME	X	X	DEVNM	X	X
BC	X	X	DF	X	X
BCOPY	X		DICTION		X
BDIFF	X	X	DIFF	X	X
BFS	X	X	DIFF3	X	X
BS	X		DIFFMK	X	X
CAL	X	X	DIRCMP	X	X

	III	XENIX		III	XENIX
CALENDAR	X	X	DIRNAME	X	X
CAT	X	X	DISABLE		X
CB	X	X	DOSCAT		X
CC	X	X	DOSCP		X
CD	X	X	DOSDIR		X
CDC	X	X	DOSLS		X
CHECKCW	X	X	DOSMKDIR		X
CHECKEQ	X	X	DOSRN		X
CHGRP	X	X	DOSRMDIR		X
CHMOD	X	X	DPR	X	
CHOWN	X	X	DTYPE		X
CHROOT	X	X	DU	X	X
CLRI	X		DUMP	X	
CMP	X	X	DUMPDIR		X
ECHO	X	X	KILL	X	X
ED	X	X	LABELIT	X	
EFL	X		LC		X
EGREP	X	X	LD	X	X
ENABLE	X	X	LEARN	X	
ENV	X	X	LEX	X	X
EQN	X	X	LINE	X	X
ERRDEAD	X		LINK	X	
ERRDEMON	X		LINT	X	X
ERRPT	X		LN	X	X
ERRSTOP	X		LOGIN	X	X
EX		X	LOGNAME	X	X
EXPR	X	X	LOOK		X
F77	X		LORDER	X	X
FACTOR	X	X	LPR	X	X
FALSE	X	X	LS	X	X
FGET.DEMON	X		M4	X	X
FGREP	X	X	MAIL	X	X
FILE	X	X	MAKE	X	X
FIND	X	X	MAN	X	X
FINGER		X	MESS	X	X
FSCK	X	X	MKDIR	X	X
FSDB	X		MKFS	X	X
FWTMP	X		MKNOD	X	X
GDEV	X		MKSTR		X
GED	X		MKUSER		X
GET	X	X	MM	X	X
GETOPT	X	X	MMCHEK	X	
GETS		X	MMCHECK		X
GRAPH	X		MMT	X	X
GRAPHICS	X		MORE		X
GREEK	X		MOUNT	X	X
GREP	X	X	MV	X	X
GRPCHECK		X	MVDIR	X	
GUTIL	X		MVT	X	
HALTSYS		X	NCHECK	X	X
HD		X	NEQN	X	X
HDR		X	NETUTIL		X
HEAD		X	NEWGRP	X	X
HELP	X	X	NEWS	X	X
HP	X		NICE	X	X
HYPHEN	X	X	NL	X	X
ID	X	X	NM	X	X
INSTALL	X	X	NOHUP	X	X
JOIN	X	X	NROFF	X	X

	III	XENIX		III	XENIX
OD	X	X	STYLE		X
PACK	X	X	SU	X	X
PASSWD	X	X	SUM	X	X
PASTE	X	X	SYNC	X	X
PCAT	X	X	SYSADMIN		X
PR	X	X	SYSDEF	X	
PREP		X	TABS	X	
PROF	X	X	TAIL	X	X
PROFILER	X		TAR	X	X
PRS	X	X	TBL	X	X
PS	X	X	TC	X	
PSTAT		X	TEE	X	X
PTX	X	X	TEST	X	X
PWADMIN		X	TIME	X	X
PWCHECK		X	TIMEX	X	
PWD	X	X	TOC	X	
QUOT		X	TOUCH	X	X
RANDOM		X	TP	X	
RANLIB		X	TPLOT	X	
RATFOR	X	X	TR	X	X
RCP		X	TROFF	X	X
RED		X	TRUE	X	X
REFORM	X		TSET		X
REGCMP	X	X	TSORT	X	X
REMOTE		X	TTY	X	X
RESTOR	X	X	TYPO	X	
RESTORE	X	X	UMASK	X	X
RM	X	X	UMOUNT	X	X
RMAIL	X		UNAME	X	X
RMDEL	X	X	UNGET	X	X
RMDIR	X	X	UNIQ	X	X
RMUSER		X	UNITS	X	X
RSH	X	X	UNLINK	X	
RUNACCT	X		UNPACK	X	X

	III	XENIX		III	XENIX
SACT	X	X	UUCLEAN	X	X
SAG	X		UUCP	X	X
SCC	X		UULOG	X	X
SCCSDIFF	X	X	UUNAME	X	
SDB	X		UUPICK	X	
SDDATE		X	UUSTAT	X	X
SDIFF	X	X	UUSUB	X	X
SED	X	X	UUTO	X	X
SETMNT	X	X	UUX	X	X
SETTIME		X	VAL	X	X
SH	X	X	VC	X	
SHUTDOWN	X	X	VI		X
SIZE	X	X	VLX	X	
SLEEP	X	X	VDLCOPY	X	
SNO	X		VPMC	X	
SOELIM		X	VPMSNAP	X	
SORT	X	X	VPMSTART	X	
SPELL	X	X	VPR	X	
SPLINE	X	X	VSH		X
SPLIT	X	X	WAIT	X	X
STACKUSE		X	WALL	X	X
ST	X		WC	X	X
STAT	X		WHAT	X	X
STRINGS		X	WHO	X	X
STRIP	X	X	WHODO	X	X
STTY	X	X	WRITE	X	X
XARGS	X	X			
XREF	X	X			
XSTR		X			
YACC	X	X			
YES		X			

IBM PC XENIX "Open" Architecture

IBM Entry Systems Division established a policy of "open" architectures when it announced the IBM personal computer. The intent of that policy was to encourage both hardware and software developers to develop products for the family of IBM Personal Computers.

The "open" architecture philosophy was a design goal of IBM PC XENIX. It was hoped that by allowing, and encouraging, independent developers to enhance the product through both application and kernel products that the success of the product would be ensured.

PC XENIX provides a number of well defined interfaces which allow independent developers to enhance the basic product offering. Along with the ability to enhance PC XENIX it is also possible to tailor the PC XENIX kernel to the specific needs of a particular market sector. The following sections describe the various aspects of the PC XENIX "open" architecture in more detail.

Reconfigurable Kernel

The PC XENIX kernel can be reconfigured in various ways to more specifically address the needs of the user. The following list of configurable items is not exhaustive, but serves as an indication of the level of configurability designed into the PC XENIX product.

- PC XENIX Kernel Size
- Size Of Kernel Buffers
- Number Of Concurrent Processes
- Number Of Open Files Per Process

PC XENIX was designed to allow users to quickly and easily modify the parameters which define each of these kernel features.

Configurable Device Drivers

The most flexible feature of PC XENIX is its ability to conform to any user-specific hardware. This is accomplished through a kernel module called a device driver. PC XENIX is shipped with the following device drivers.

- Keyboard Device Driver
- Display Device Driver
- Asynchronous Communications Device Driver
- Floppy Disk Device Driver
- Hard Disk Device Driver

However, if a user requires the ability to interface to a different hardware device a user supplied device driver must be incorporated into the PC XENIX kernel. The guidelines for the development of the device driver, as well as the installation of the device driver, are provided in the PC XENIX documentation.

Device drivers are the link between the PC XENIX kernel and the external world. The capability to alter or enhance this connection provides the user with an unlimited flexibility. For example, a user that requires the ability to have more than three users logged on at any particular time may want to write a device driver for the IBM PC-Network Adapter which would provide the capability to increase the number of active users on the IBM Personal Computer AT.

The following list describes some of the areas in which device drivers may be written to provide an enhanced level of function over and above that of the shipped product.

- IBM Binary Synchronous Communications Adapter
- IBM Synchronous Data Link Control (SDLC) Adapter
- Magnetic Tape Adapter
- Non-Standard Disk Adapter
- Data Collection/Analysis Subsystem
- Any Non IBM Supported Hardware Device

Configurable Pseudo Device Drivers

Pseudo device drivers comprise a subset of the general class of device drivers. What is meant by the term pseudo device is that a device is not actually present. However, the user may wish to utilize the well defined device level interface in PC XENIX to achieve some level of enhanced functionality in the kernel.

The classic example of a pseudo device is that of a RAM disk. PC XENIX allows the user to quickly and efficiently develop a pseudo device driver which simulates the functions of an actual disk, but is maintained in main memory. There are a number of other pseudo device applications which all follow the same basic philosophy as that of the RAM disk pseudo device.

Clearly, a major advantage in PC XENIX is its well defined device interface which allows users to quickly and easily enhance the features provided in the system.

Configurable Interrupt Handlers

The PC XENIX product is a general purpose operating system which is based on a time sharing scheduling algorithm. However, there is a class of applications which requires a specific response time to external or internal events. In general, PC XENIX cannot adequately address this class of applications. There is, however, a way for the user to employ PC XENIX in this environment.

The ability in PC XENIX to modify and enhance the kernel level functions by incorporating new device drivers allows a user to develop a real-time response system. This type of system is based on a special type of device driver, typically, referred to as an interrupt handler or interrupt service routine. In general, this type of device driver is driven by one of the following events.

- Internal Clock Interrupt
- External Device Interrupt
- Internal Software Interrupt

Once the interrupt service routine (device driver) receives the interrupt it is capable of the following types of functions.

- Posting The Interrupt
- Queuing Real-Time Functions
- Invoking Real-Time Functions
- Utilizing Any And All Kernel Functions

Utilizing this technique of configuring real-time interrupt handlers into the PC XENIX kernel allows the developer to provide real-time and quasi real-time applications. This technique can be used to ensure that the real-time activities respond within the desired period of time.

PC XENIX and System Bios

PC XENIX was developed to take full advantage of the IBM Personal Computer AT architecture. In order to accomplish this, it was necessary for PC XENIX to operate in the protected mode of the INTEL 80286 which is the processor of the IBM Personal Computer AT. Utilizing this mode of operation prohibits certain compatibility mode (INTEL 8086) code from executing properly.

In order to avoid confusion and complication in this area it was decided to develop the system level BIOS in compatibility mode which would allow all DOS applications to operate properly. However, when executing PC XENIX the system level BIOS is only partially used. To better understand the issues concerning PC XENIX and BIOS the following sections were included.

Power On System Test (POST)

When the IBM Personal Computer AT is powered on, the code responsible for Power On System Test (POST) is performed. This code is resident in the BIOS ROM and is responsible for ensuring the proper functioning of the system. If particular hardware components are not attached or are not functioning properly then this code will alert the user to this fact and attempt to provide an error message which isolates the problem.

Since this code is executed prior to any other code it is not necessary to be concerned with the mode in which this code executes. Likewise, it is not necessary to duplicate the functions of this code in the PC XENIX kernel. Therefore, some code savings are provided by allowing the system level BIOS to perform these power on system tests prior to loading and executing the PC XENIX kernel.

Bootstrap Loader

The only other feature of the BIOS ROM that is used when PC XENIX is installed is the bootstrap loader. This piece of code is executed immediately after a successful POST. This code loads a few sectors of code from the active partition on the boot device and then allows that code to execute. The loaded code is typically an operating system specific loader which will ultimately load the operating system.

The features provided by the ROM BIOS bootstrap loader allow the operating systems to be uniformly and properly loaded based on the user's definition of the active partition and the boot device.

Real Mode Implementation

The major obstacle which prevents real mode (standard DOS) programs from executing in the protected mode of the INTEL 80286 is the level of protection which is imposed by the processor during execution. The INTEL 80286 provides the following four levels of protection while executing in the protected mode.

- Restricted Use Of Certain Instructions

- Separation Of System And User Code
- Separation Of Users' Code
- Data Type Checking

The other enhancements provided by the INTEL 80286 have less of an effect on the ability of a real mode program to execute in a protected mode environment than the actual protection enforcement mechanism.

Multi-tasking Incompatibilities (System Level BIOS)

Finally, the system level BIOS was implemented to conform to a single-task and single-thread environment. The BIOS was developed to optimally exploit this environment including the use of the following coding techniques.

- Timing Loops
- Device Polling
- Non-Reentrant Code

PC XENIX and Adapter BIOS

The architecture of the more advanced IBM Personal Computer I/O adapters has migrated towards a separate adapter processor with its own adapter memory. This processor is programmed to receive high-level commands from the system processor and perform complex operations without the aid of the system processor.

The architecture of this type of I/O adapter can dramatically increase the I/O through-put by off-loading the system processor. Since the architecture of these adapters support the ability of the adapters to have onboard memory, both ROM and RAM, it is not surprising to find that these adapters also provide a ROM-resident routine which is mapped into the system processor's memory address space.

The function of this ROM-resident routine, or set of routines, is to provide the following basic services required by an application in order to utilize the adapter for I/O operations.

- Adapter Power On System Test
- Adapter Diagnostic Test
- Adapter Initialization
- System Level Interrupt Vector Initialization
- Basic Input/Output Services (BIOS)

In short, there is a direct correlation between the functions provided by the system level BIOS and the adapter level BIOS. It should be noted that many of the same advantages and disadvantages of the system level BIOS are present as well.

Real Mode Implementation

As mentioned in the section concerning the system level BIOS the major obstacle which prevents real mode (standard DOS) programs from executing in the protected mode of the INTEL 80286 is the level of protection which is imposed by the processor during execution. The other enhancements provided by the INTEL 80286 have less of an effect on the ability of a real mode program to execute in a protected mode environment than the actual protection enforcement mechanism.

Since the ROM-resident routine, or routines, which are mapped into the address space of the system processor are programmed in real mode, it is virtually impossible for those routines to be used while the processor is executing in the protected mode.

Clearly, this prohibits this code from being utilized by either the PC XENIX kernel or a PC XENIX application.

Interrupt Vector Allocation

The introduction to this section presented a number of functions which are performed by the code which are resident on the enhanced I/O adapters. One of those functions was to initialize one or more of the interrupt vectors of the system processor in order to be used as a direct entry point into the routine, or routines, that are ROM-resident on the adapter and constitute the adapter level BIOS.

Although this technique is a very efficient one while executing in the real mode of the INTEL 80286 it is typically not allowed while executing in the protected mode of the processor. The major reason for this is that in order to preserve the integrity of the operating environment the interrupt vectors of the processor must be carefully regulated. Therefore, this type of uncontrolled access to the interrupt vectors is not tolerated in protected mode.

Multi-tasking Incompatibilities (Adapter Level BIOS)

Finally, the adapter level BIOS was implemented to conform to a single-task and single-thread DOS environment. The BIOS was developed to optimally exploit this environment including the use of the following coding techniques.

- Timing Loops
- Device Polling
- Non-Reentrant Code

Although these features may not provide as great a problem as they do in the system level BIOS it is not clear that the appropriate level of device semaphoring and queuing is provided to support a full multi-task environment. However, this a moot point since the code will not operate properly in the protected mode of the INTEL 80286.

Affected Adapters

A number of the newer enhanced IBM Personal Computer I/O adapters are affected by the lack of protected mode BIOS at the adapter level. The following is a short list of the adapters which currently are affected.

- IBM Cluster Adapter
- IBM Personal Computer Network Adapter
- IBM Enhanced Graphics Adapter

This is not meant to constitute a complete and exhaustive list of the adapters which are affected by the lack of protected mode BIOS. It indicates the general architecture of adapters that might be affected.

INTEL 80286 Real Versus Protected Mode

The previous sections discussed the effects of executing a program in either the real or the protected mode on the INTEL 80286. However, the terms were not well defined. This section will attempt to provide more detail with respect to each of these two operating modes.

A key point to remember is that the INTEL 80286 can operate in one of two distinct modes. The real mode of operation allows the 80286 to emulate the 8088/86 processors with some minor instruction differences, while the protected mode of operation allows the 80286 to provide a variety of enhanced features. The following is a list of the type of features that the 80286 processor provides while executing in the protected mode.

- Memory Management
- Memory Protection
- Increased Memory Addressing
- Enhanced Memory Addressing Modes
- Enhanced Processor Instruction Set

Each of the major features in this list will be discussed in detail below in order to provide a better understanding of the advantages and disadvantages of program execution in each of the two modes of operation.

Memory Management

The INTEL 80286 while operating in real mode has the same limitations on memory addressing as the 8088/86. Simply stated, the address space is limited to 1 M-byte. However, due to the design of the IBM Personal Computer the actual maximum application memory address size is limited to 640 K-bytes in order to allow space for memory mapped I/O devices.

Even though the IBM Personal Computer AT may have up to 3 M-bytes of memory, if the processor is operating in real mode, only the first 640 K-bytes are actually available for program execution. On the other hand, if the 80286 is operating in the protected mode, as in the case of PC XENIX, the program addressability is much more than 640 K-bytes.

While operating in protected mode the processor can address up to 16 M-bytes of real memory and considerably more logical memory. Utilizing this mode PC XENIX is capable of managing as much memory as the user is capable of installing in the IBM Personal Computer AT. This feature of PC XENIX allows the scope and complexity of applications to grow and expand to meet the needs of even more sophisticated technological improvements.

Memory Protection

There is no explicit memory protection while executing in the real mode of the INTEL 80286. Therefore, it is not possible to restrict the following types of activities.

- Restricted Use Of Instructions
- Protection Of System Code
- Protection Of Users' Code
- Data Type Enforcement

However, while executing in the protected mode of the INTEL 80286 it is possible to provide the following four levels of protection which allow protection at the system as well as the user level.

- Restricted Use Of Certain Instructions
- Separation Of System And User Code
- Separation Of Users' Code
- Data Type Checking

The advantages and disadvantages of each must be weighed by the developer with respect to each application. However, it is clear that the level of software integrity is dramatically increased while executing in the protected mode.

Register Usage

It should be noted that do to the pipelined architecture of the INTEL 80286 some of the instructions which are executed in real mode operate differently than one would expect on a standard INTEL 8088/86. A complete list of these instructions and the differences can be found in the IBM Personal Computer Seminar Proceedings describing the IBM Personal Computer AT. Likewise, the INTEL 80286 reference manual provides this information in more detail.

Application Compatibility

The major concern of most users and developers is that of application compatibility. Simply stated, "What software do I now have that will execute on this machine or operating system"? This section is intended to provide answers to this question and others that arise from it.

In order to more accurately address the issues surrounding this topic it is necessary to discuss both execution level compatibility and source level compatibility. Each of these will be discussed relative to the following three major application environments.

- UNIX System III Applications
- PC/IX Applications
- PC-DOS Applications

UNIX System III Applications

PC XENIX is an AT&T System III based UNIX. This means that the majority of the kernel, utilities, and applications were migrated directly from the distributed AT&T source code. The total number of lines of source code that migrated from the initial AT&T source code to PC XENIX was well over 400,000 lines.

The ability to migrate UNIX System III based source code to PC XENIX is clearly not a major task. However, there are a number of problems which do arise if the source code attempts to utilize any of the following features of the C language.

- Integer/Pointer Coersion
- Pointer References To Control Hardware
- Pointer References To Control Software
- Machine Dependent Coding Techniques

If the programming techniques used in the application are relatively clean and conform to the basic standard I/O library interface then the task of migrating the application is greatly reduced.

The ability to migrate applications at the object or executable level is virtually non-existent. The major reasons are listed below.

- Different Processor Instruction Sets
- Slightly Different "a.out" Formats
- System Call Initiation Differences

Therefore, the only level of compatibility which exists between applications running in a UNIX System III environment and PC XENIX is at the source level. Once again, this is only true if the appropriate code standards have been followed. Otherwise, the migration task can be very difficult.

PC/IX Applications

The level of application compatibility between PC/IX applications and PC XENIX is the same as that for standard UNIX system III applications. Although this might seem to be somewhat limited due to the fact that PC/IX operates on both the IBM Personal Computer XT and AT, the problem arises from the fact that PC/IX is operating in real mode and PC XENIX is operating in protected mode. Therefore, it is as though there were two separate machines and so the level of application compatibility is reduced to the same level as previously discussed.

PC-DOS Applications

The ability to migrate PC-DOS applications source code to PC XENIX is clearly not an easy task. There are a number of problems which exist. The major problem areas are caused by the different operating environments in each of the two products. For example, PC-DOS applications have the following characteristics.

- Non-reentrant Code
- Fixed Addressing After Initial Load
- Non-swappable
- Total Machine Resource Philosophy

On the other hand, UNIX applications conform to a much different set of guidelines and have different characteristics, such as.

- Reentrant Code
- Swappable Code
- Dynamic Relocation
- Process/Kernel Resource Philosophy

Many C language applications can be migrated from the PC-DOS environment to the PC XENIX environment if the following features are avoided.

- Integer/Pointer Coersion
- Pointer References To Control Hardware
- Pointer References To Control Software
- Machine Dependent Coding Techniques

Likewise, if the programming techniques used in the application are relatively clean and conform to the basic standard I/O library interface then the task of migrating the application is greatly reduced.

The ability to migrate applications at the object or executable level is virtually impossible. The major reasons are listed below.

- Different Processor Instruction Sets
- Slightly Different "a.out" Formats
- System Call Initiation Differences

Summary

PC XENIX is a full function, multi-task, multi-user operating system which exploits the capabilities of the advanced INTEL 80286 architecture. This provides users with the capability of minicomputer performance and environment at microcomputer prices.

Notes

IBM Corporation
Editor, IBM Personal Computer Seminar Proceedings
4629
Post Office Box 1328
Boca Raton FL 33432