Know Your Personal Computer

9 High-Level Operating Systems

S K Ghoshal

This article briefly surveys the predominant operating systems that run on the IBM PC architecture.

Siddhartha Kumar Ghoshal works with whatever goes on inside parallel computers. That includes hardware, system software, algorithms and applications. From his early childhood he has designed and built electronic gadgets. One of the most recent ones is a sixteen processor parallel computer with IBM PC motherboards.

Introduction

As we had stated in Part 3 of this series, there are low-level and high-level operating systems that form the major part of system software on top of which application programs run on the IBM PC architecture. BIOS, which we described in detail in Part 8, is the low-level operating system. In this article, we shall survey the high-level operating systems. We shall for brevity call them operating systems in this article; this is the popular notion too.

Three types of operating systems are predominant on PCs:

- MS-DOS, called DOS by many people
- Windows
- Unix

In each type there are many versions and variants. There are many sources for Unix. DOS and Windows are mainly from Microsoft. These names are trade-marked. Microsoft owns the names 'MSDOS' and 'Windows'. As of now, the name 'Unix' is owned by Novell Inc. So other developers of Unix-like operating systems (some of whom may give it for free) cannot call it Unix. That (fortunately) is the only restriction.

By now, all the three operating systems have advanced features. All the applications that you need to run your PC are available on any of the three operating systems. Some people are familiar

The previous articles of this series were:

1. Introduction to computers, January 1996.
2. The personal computer hardware, February 1996.
3. The personal computer system software, April 1996.
4. The CPU base architecture, July 1996.
5. The CPU base instruction set and assembly language programming, November 1996.
6. Memory organisation, February 1997.
7. Input-output ports, May 1997.
8. Basic input -output system (BIOS), July 1997.

with only one of these operating systems, while there are others who are familiar with more than one system. There are people who change from one to another gradually. If you are already familiar with one system, the aim of this article is not to change your preference or influence you in any way in your future decision in that regard. That depends on what you do and intend to do with your PC, rather than what you read in *Resonance* about PCs. Thus after briefly comparing and contrasting the three operating systems, we shall discuss what is common across all operating systems that run on PCs.

DOS

It is the original operating system of the PC. Initially it had very limited capability. It required very little memory and power of the CPU and board-level architecture. It has come a long way since then, meandering through many versions and bug fixes. It has grown along with the PC architecture. The evolution of the PC architecture and that of MSDOS have traditionally influenced each other. As of now, MSDOS is still the preferred operating system of millions of PC users. If you do not mind typing in a lot of commands through the keyboard, manage your own files and directories and work on low-cost PC platforms, MSDOS could be your choice. MSDOS does not make exorbitant demands from the PC architecture to be installed on it or to work comfortably with it. Considerable literature and documentation exists about MSDOS, which helps you to develop applications and products on this platform rather conveniently. Information about its internal organization is known by its end-users. Many applications have been developed on DOS by independent developers and their collective experience cannot be coerced or exploited by any vendor or any group of people. However, MSDOS has become quite dirty inside its bonnet. It has to provide so many features for which it was not designed. It handles many types of devices on the PC architecture which adds to its baggage. And of course it has to keep compatibility

with all the previous MSDOS versions, even though some of their aspects no longer make any sense. Thus MSDOS is dying a slow death. It cannot adapt itself rapidly to the changing needs of the applications or take advantage of the enhancements of the architecture, because it is already too bulky and badly organized. As of now, the task of designing and implementing a better MSDOS by rewriting the old code and re-organizing the traditional data-structures is much more formidable than developing an MSDOS-like, but new operating system from scratch, after a good design. Thus it already might have become financially and practically unattractive for any vendor or dedicated individual to use MSDOS.

Windows

It started out as a badly designed Graphical User Interface of MSDOS, running on top of MSDOS, as yet another layer of system software. PCs then had slow processors and low-resolution graphics. Thus Windows was unattractive. But Bill Gates and his crew at Microsoft did not give up. They refined Windows to become a full-fledged operating system taking all the advantages of the architectural features of the 80X86 and the PC motherboard. (See Parts 2 and 5.) The dependence of Windows upon MSDOS has been greatly reduced over a period of time. (This dependence caused many errors when you tried to run applications on the earlier generations of Windows.) Now, Windows is indeed a full-fledged operating system working on its own. At present there are two versions of Windows. *Windows NT* is designed from the operating system purists' point of view. It does not allow misbehaving applications to run. (See Part 5). As many applications must misbehave if they have to run at all (see Part 6) on PCs, many developers and users do not like *Windows* NT. For them, there is the flexible *Windows 95* which allows some known forms of misbehavior. That means Windows 95 will have to undergo many revisions to satisfy the emerging needs of applications

Your Own Operating System

You can always design and develop your own operating system, for the three main systems may not meet your needs or they may be badly implemented on the PC architecture and you may need to rectify that. With sustained effort, all by yourself, you can design and implement a better operating system. You will get all the support from the PC architecture to implement it. That is as long as the demands of your operating system on the architecture are reasonable. However, it is highly improbable that many would change over to your operating system just because it is a better one.

Returning the Compliment

'Build a command-language interface that even an idiot can use, and only an idiot will ever want to use it' was the reigning principle before graphical user interfaces running on PCs changed that mind-set. Experienced users and people who know a lot about what is inside PCs need a working environment that is easy to use. As they need to get a lot of work done in a session, an easy way of interacting with the computer helps them considerably. Even Peter Norton (of Norton commander and Norton Utilities fame) uses Windows for his routine work these days.

and has all the features of a practical operating system. (See Part 8 for the debate in this regard in a general context). If you have not used a computer before, Windows could be a good environment for you. Most work gets done by clicking the mouse buttons after pointing the cursor at an icon that describes the application that you want to run. You can run many applications simultaneously, without having to remember strange commands.

Every product of an engineering design goes through three phases. First a grossly inadequate implementation fails to entice users. Then an immensely large and complex system offers mechanisms to do things you always wanted to do, but baffles you with its complexity. Some of the mechanisms available in the second generation do not work correctly all the time. Only in the third generation, a system that is adequate, powerful, reliable and easy to use, emerges. Windows has recently reached this stage, from which it will set the specifications for future operating systems.

Unix

Unix was born and nurtured in research laboratories and universities where it ran on minicomputers and was looked after well by a group of people whose common characteristics were:

They knew their basics rather well. That is why Unix is so well designed and this design made it live so long and get ported to so many architectures. Unix is the most common operating system today across architectures. There is no well known computer architecture that does not host some variant of Unix.

They interchanged their programs and ideas. That is why information can be shared and exchanged between computers and users so easily on Unix machines. So programs can be re-used for many different purposes in Unix.

They were honest. They had the capability to do harm to their (more) gullible colleagues, which they did not intend to do. That is why much privacy and information security cannot be enforced in Unix. You expect others will not look at or modify what they know you will not like to share with them. There is no way you can guard them against illegal access by an expert.

They were responsible and knew what they were doing. Unix trusts the user. Thus it is possible to destroy huge amounts of information by the owner herself without any question or warning by Unix. If you did that by mistake, Unix does not give you a second chance to do it right. If you are a novice or a know-all, do not mess with Unix.

They themselves were the users of Unix. That is why Unix is so configurable. An advanced user of Unix can and does add to the capabilities of Unix in many ways which is good. However, most advanced users need to do this one way or the other, which is bad. If you do not have the ability or willingness to change Unix, you should not complain about what it cannot do for you. The entire source code of Unix is available for some variants of Unix (e.g. Linux for PCs) and you can always re-implement it to suit your needs and tastes. Thus there are many variants of Unix and hence not too many standards are followed across them.

They loved being cryptic. That is why Unix commands are so short and seem meaningless at first encounter. So C, the programming language in which Unix applications are written and implemented, is so cryptic in its syntax.

If you have some of these characteristics, Unix is good for you. Graphical User Interface and Windowing environment running on top of Unix on PCs is provided by the X-windows library. Its code is given away free and can be customized too by the end-user. All the software tools that are needed for application development on a Unix platform using PCs are

Unix commands are so short and seem meaningless at first encounter. So C, the programming language in which Unix applications are written and implemented, is so cryptic in its syntax.

Bismarck said "Man cannot alter tides. He can only float along with them and steer"

available for free on the Internet. What you develop using these tools, you should give away for free like most of us do. When Unix came to PCs first, it viewed the PC architecture like it did many other architectures before PCs' advent. Unix abstracted away the architectural features of 80X86 by acting through software drivers and ignored the power of the board-level architecture. This made the initial versions of Unix that ran on PCs slow, consuming a lot of memory. And in a few places within its repertoire, Unix was not working correctly.

There are dedicated groups of developers now, both in the industry and in groups that give away software for free, who develop Unix specifically targeted for the 80X86 and the PC architecture. (Refer to Part 8 of this series). Unix as of now, is rapidly adapting itself to PCs where memory and processor power are still a bit lower than what they are in workstations. PCs, however, are rapidly becoming as powerful as workstations in the low price ranges. But these two revolutions are in no way coordinated, either within their own work-groups or among Unix and PC developers. If you are a competent PC applications developer who loves Unix, just keep taking advantage of both the trends. Recall what Bismarck had said: "Man cannot alter tides. He can only float along with them and steer".

Operating Systems on PCs

In order to host applications on a PC architecture, an operating system must have the following capabilities:

It must support programs executing on top of it. Multiple programs, each having its own virtual address space, should be able to run simultaneously on top of the operating system. MSDOS and early versions of Windows were not fully capable of this, but now Windows has achieved it. Unix always had it.

Each program should work without interference from other pro-

grams and the operating system. The addresses in the virtual address space should be translated into disjoint swathes of physical address space (see Part 1). The processes may acquire or release blocks of memory. There are many other resources (needed by application programs) that the operating system maintains and controls. The operating system must maintain accounts of all these transactions. Like an expert banker, it knows when (and to whom) it is safe to lend a resource and when it is not. It aims to make sure that all the application programs eventually get what they need in a matter of time. Another (and this aspect is often kept secret by its developer) policy it follows is to make sure that the operating system itself never runs out of resources it needs to do its work. All the predominant PC operating systems are good in this aspect.

It must allow programs to talk between themselves if they need to. It may open channels of communication (which also is a resource) and manage them. If one party talks too much without the other responding, the operating system, like an expert telephone operator, renders the proper advice and does the needful about the status of the connection.

It must manage information on behalf of the application programs. It supports a file system. The application programs can create, delete, modify and interchange files. Different ownership and access rights are enforced by the operating system. Collection of records (let us say on employees in a firm), structured databases, mail messages and other types of data (e.g., video or audio) are often managed by the operating system. When multiple programs try to access a common database, the operating system ensures proper coordination between them so that the integrity of the shared information is maintained.

In future we may elaborate on these issues and write more on Unix variants that run on PCs. If you have any reaction to this article, write to me.

Suggested Reading

- *Que's MSDOS 5 User's Guide*. Prentice Hall of India, 1993.
- Peter Norton. *Peter Norton's Complete Guide to Windows-95*. Prentice Hall of India, 1996.
- Jack Tackett and others. *Using Linux*. Prentice Hall of India, 1996.
- Linux Source code from http://www.linux.org/

Address for Correspondence
S K Ghoshal
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore 560 012, India
email:
ghoshal@serc.iisc.ernet.in
Fax: (080) 334 1683