Know Your Personal Computer

3. The Personal Computer System Software

S K Ghoshal

This article surveys system software organization in personal computers using the IBM personal computer software. Details of the hierarchical levels of the system software organization are explained.

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

An overview of the system software organization of a PC is given in *Figure 1*. BIOS (basic input output system) is the lowest level of the operating system. It hides the details of the specific motherboard and projects a higher level interface to the physical hardware. MSDOS (microsoft disk operating system), the most common operating system of the IBM PC uses it to run application programs. These programs can be written in any one of a number of high-level languages (HLL) such as Pascal, C, FORTRAN. Each HLL presents the programmer with an abstract view of the computer.

The complex operations supported by a HLL are implemented by a *runtime library (RTL)* specific to that language. The RTL makes calls to the MSDOS operating system so that various tasks may be performed by the IBM PC hardware and BIOS.

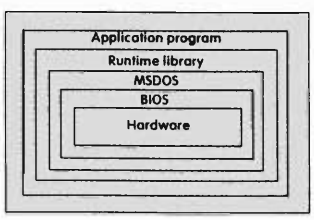

Figure 1 System software of computers are organized in layers like an onion.

Organization of System Software into Levels

Organizing system software into many hierarchical levels or layers as shown in *Figure 1* gives it the following advantages:

Portability: Programs written using a PC can be moved, compiled and run on any other computer.

Runtime Library

A Runtime Library (RTL) comes with every high level language compiler. It is a collection of large number of subroutines that perform well-defined but complex operations. These operations are needed by every programmer who uses the computer by programming it in the particular high-level language in question. She cannot write each and every detail of how that complex operation is to be done on that given computer. This is because she does not know all the details and does not have all the time needed to understand them before she writes and runs her first program. So the compiler implementor supplies a 'library' of canned programs which our programmer will need. RTL is written by the compiler implementor.

- *Scalability:* The hardware, software and the compilers used to execute an application program can be upgraded to improved versions without having to rewrite or change the program.
- *Modularity:* Any reconfiguration, upgradation or replacement of the software at any level does not affect its function, overall performance or organization.
- *Easy understanding and documentation:* The working of each program level can be understood and explained without getting into details of other levels.

The major disadvantages of organizing system software in layers are:

- Speed of operation. It slows down the application program execution.
- The full potential of the hardware is not used by an application program.

We will illustrate these points with a simple example. Consider the program in *Figure 2* which is written in the C programming language. It stores a number as an integer variable and then prints its value in the hexadecimal representation. Note that several advantages are gained by organizing the system software in a hierarchical fashion on IBM PCs as shown in *Figure 1*.

- This program can be run as an application in any general purpose computer with a C compiler.
- To port the program from one machine to another one must recompile it with a C compiler at the destination site and run it thereafter.
- The programmer need not understand the display hardware of the machine, the addresses occupied by it and so on.
- The output need not always be displayed. It may be redirected to a disk file or any other device, by the operating system.
- The programmer need not have any information on the word

```
main ( )
{
/* The following statement stores a hexadecimal number in i */
unsigned int i = 0 x 1234ABCD ;
printf ( " % x \ n ", i) ; / * The statement prints the value stored in i */
}
```

Figure 2 A small C program to store a value in a variable i and display it on a video display.

length of the CPU, the storage of the bytes of a word in physical memory and other details of the CPU-memory subsystem.

- The virtual models of the CPU-memory subsystem and display devices as seen by a C programmer are adequate to write portable programs.
- Anyone who knows C can understand what this program does. Enhancing the CPU-memory and the video display subsystems does not affect the semantics[1] of the program.
- Anyone writing the operating system and other system software knows how to use the basic commands to run the program correctly.

On the other hand, such an organization has the following disadvantages:

- Most of the execution time of the program is spent in the *printf* statement. Although only a write operation is needed for the output, the overhead of the *printf* statement is enormous. That is because the system call for an output operation passes through many levels of the system software. The actual write operation to the video memory occurs at the lowest level of the system software. Between any two levels, a unique and elaborate parameter passing protocol must be followed. All these operations consume CPU time and are called *overheads*.
- Even though the video graphics hardware displays millions of shades of colours, the output produced will only be in one colour, the one to which that environment has been set.
- The program writes only at the current cursor position in one

[1] The operation intended to be performed by a computer program is called its semantics. Unlike human languages, the semantics of computer programming languages and programs can never be ambiguous or ill-defined. The semantics of a computer program must be so straightforward that even a computer (which cannot do anything unless told how to do it and does not normally learn by experience or example) can carry it out. And to all computers in the world, it must mean the same thing.

What is Snowing?

A modern video display adapter has its own memory (video RAM) and processor (called display co-processor). If you write directly into the video memory, you may interfere with the display co-processor attempting to read it for producing the necessary red, green, blue or composite video signals. Even though the video RAM has two independent read/write ports, the CPU gets a higher priority when both display processor and CPU try to simultaneously access it. As a result, the display processor does not get the data it needs from the video memory. It thus goes haywire and sends the beam aimlessly, illuminating bright spots which appear all over the screen. This appears like snow on the screen. BIOS knows many details of the hardware it controls and makes it appear *clothed* and more sophisticated than it actually is. Do not try to control the hardware. Let BIOS handle it. Even MSDOS gets to the hardware via BIOS. Look at *Figure 1* again. MSDOS in turn has its own share of idiosyncracies. So do not talk to MSDOS directly. Ask your RTL to do that. Just write your application in C as I have in *Figure 2*. It will work without your having to know anything about the hardware, BIOS, MSDOS and the RTL of the C compiler.

given window, giving the programmer no control over its absolute location on the two-dimensional video screen.

- The program uses only the default font type and size. This occurs despite the capacity of the display hardware and driver system softwares for supporting various font types and sizes.

The moment one tries to alleviate these difficulties by writing directly into the video memory or directly programming the display controller hardware, the program loses its portability. Thereafter it remains tied to that specific configuration of the video display hardware.

One of the advantages of organising system software in a hierarchical fashion is that the programmer need not understand the display hardware of the machine, the address occupied by it and so on.

In the context of *Figure 1*, let us see how the program given in *Figure 2* is executed.

- The application program of *Figure 2* calls the function *printf*. The program to carry out *printf* is in the runtime library file that comes along with the C compiler. The parameters supplied to *printf* are the 32-bit integer which is a number(in our case *0x 1234ABCD*) and the print format (in our case % x).
- The runtime library converts the data to be printed into an

Beware of Turbocharging

Many compilers produce a code that directly modifies some internal data structures of low-level operating systems and writes into device control registers directly. In other words, the runtime libraries of such compilers, directly call BIOS (the BIOS call can be nonstandard and undocumented) or initialize hardware, bypassing intermediate levels of system software. This is called *supercharging or turbocharging*. The implementors of such compilers have inside knowledge of the IBM PC architecture and BIOS from sources that are not always open. The object code produced by such compilers runs much faster than the ones produced by portable compilers but has no guarantees. Any upgrade or replacement to the system at any of the levels shown in *Figure 1* may render an old and useful turbocharged code unable to work. Invest your money and ideas on portable compilers. CPUs rapidly get faster over the years. The temporary gain in speed from turbocharging today may be obviated in a couple of years by upgrading. You also risk becoming obsolete in a couple of years if you turbocharge your code.

ASCII string. It then calls the operating system to print the string. In the IBM PC and MSDOS environment, it makes a function call 09 and a system call to the service routine of INT 21H which is part of the code text of the MSDOS operating system.

- MSDOS determines how to perform the output operation. At this stage, it can redirect the output to an open file. In case the output has to be written on the video display device, it calls BIOS by making a system call to the service routine of INT 10H with a function code to display a character at the current cursor position. Typically, it makes as many BIOS calls as there are characters in the string to be displayed.
- BIOS knows the type of the display adapter, the memory addresses of the video RAM and the I/O port addresses of the video controller. Its internal data structure gives the mode in which the video adapter is initialized; so it knows when to scroll the display, perform a carriage return and line feed the cursor. Scrolling is done either by physically moving data in the video RAM by using the CPU, or by setting an appropriate page register in the video display adapter. BIOS can therefore handle the ASCII codes of carriage return and line feed char-

Instructions to obtain services of BIOS and MSDOS

MSDOS is accessed through the software *interrupt* instruction. There are eight different DOS interrupts. One of these, INT 21H, implements 87 different functions which are referred to as the DOS *function calls*. BIOS is also entered using an interrupt. INT 10H is used to enter video-IO routines and use the display of the IBM PC.

Interrupts

Events other than branch and subroutine call instructions which alter the normal sequential flow in a program are called *interrupts*. Four major types are: requests for attention by I/O devices, invoking operating system service from a user program (service call) and errors such as arithmetic overflow/underflow using undefined instruction. There is no uniform terminology to describe these events. IBM and Intel call all of them *interrupts*, Motorola and the MIPS computer corporation call them *exceptions*, and DEC calls interruptions caused by system calls and error conditions *traps*. Whenever exceptions/interrupts occur the current program is suspended and the control jumps to service the interrupt. The current state of the CPU is saved so that the program can later resume from the point where it was suspended.

acters if they appear in the string to be displayed.

- The text of BIOS is in the EPROM. It writes directly into the video memory producing the desired display.

Similar mechanisms exist in other computer architectures and operating system organizations. BIOS is simpler to understand than some of them.

In subsequent articles we will learn more about the system software of the IBM PC. Feel free to write to me if you have any questions or comments about this article.

Address for correspondence
S K Ghoshal,
Supercomputer Education and Research Centre,
Indian Institute of Science,
Bangalore 560 012, India.

Optical illusions ...