Telecommunications Electronic Reviews (TER)

ter
telecommunications
electronic reviews

Volume 3, Issue 2, May 15, 1996

Contents

ter issues (/lita/publications/archive/ter)

REVIEW OF: Ralf Brown and Jim Kyle. Network Interrupts: A Programmer's Reference to Network APIs. Reading, MA: Addison-Wesley Publishing, 1994.

by Tony Toyofuku

As manager of our library's CD-ROM lab, a few years ago I was faced with a technical riddle: the standalone computers needed the program Microsoft CD Extensions (MSCDEX) loaded into memory in order to access the attached CD-ROM players. But with MSCDEX loaded, the menuing software we used required that

there be a disc in the CD-ROM player when the machine was first booted. If no disc was inserted into the player, the error message " Unable to read drive D: Abort? Retry? Fail?" would appear on the screen. For security reasons we kept the discs behind the reference desk, so this error was very common.

The simple solution was to load MSCDEX not from the "autoexec.bat" file, where it would normally be called, but rather from within the menuing software itself, every time a patron requested a CD-ROM title from the menu. Unfortunately this too led to problems, for if MSCDEX had been loaded previously, invoking it a second time into memory would generate an error message.

After searching for an answer to this conundrum, I found a solution; there is an interrupt function, 2Fh 1100h, that can tell whether or not MSCDEX is loaded into memory. Armed with this information, I was able to write a small C language utility program that checked if MSCDEX was loaded. If it was resident in memory, the CD-ROM search software would be invoked as usual. If MSCDEX was not loaded, the utility first loaded it into memory and then proceeded to call the CD-ROM search program.

I found the information on MSCDEX in the 1991 edition of PC Interrupts by Ralf Brown and Jim Kyle. That original work is an outgrowth of the Online Interrupts List (http://www.cs.cmu.edu/afs/cs/user/ralf/pub/WWW/files.html (http://www.cs.cmu.edu/afs/cs/user/ralf/pub/WWW/files.html)) compiled and maintained by Ralf Brown. Mr. Brown continues to maintain this list, but at 2017 pages for the most recent online release, it has grown so large that the 1991 single volume published by Addison-Wesley has been supplanted by two separate titles: the first, PC Interrupts, deals with all types of non-network interrupts, and Network Interrupts, the focus of this review, deals with network-related interrupts.

Network Interrupts is useful for anyone writing programs for the Intel 80x86 architecture and who needs access to interrupt calls. As Ralf Brown notes in a document that is available for FTP from his Web page:

An interrupt is a hardware signal that tells the CPU to temporarily stop what it is doing and go do something else. Without interrupts, the CPU would have to constantly check for external events; with interrupts, the CPU can work on something else and still respond to an event as soon as it occurs.

This is a straight reference book; there are no samples or tutorials on how to write assembler or C language programs that use these interrupts. While there are no instructions on how to use the information contained in this book, even beginning programmers with a textbook on how to write programs accessing the registers will find this book worthwhile.

The book "includes all calls for some three dozen major application programming interfaces" (APIs), both documented and undocumented ones. (p. 1) From Novell's Netware Shell to an operating system called "TopWare" by the Grand Computer Company, there is a wealth of data on network APIs. In addition to specifications on these upper layer protocols, Network Interrupts also includes data on the hardware layer APIs for PC/TCP's Packet Driver Specification, Microsoft's NDIS, and Novell's ODI.

Network Interrupts is divided into chapters according to API, then subdivided in numerical order by function. For each function there is a brief entry outlining the purpose of the interrupt, what registers are set when the interrupt is called, what values are returned, any known conflicts the interrupt has with other software, and "see also" references to related system calls. Lastly, there is a listing on whether the function call is documented or undocumented by the vendor. [1]

Network Interrupts is not just a copy of sections from the first edition that dealt with networking; it has been reorganized and greatly enlarged. Whole chapters are now devoted to APIs that only received passing mention in the first edition; where there was only one vague entry on FTP's Packet Driver Specification in the first edition, there is now an entire chapter. Likewise there was scant information on Novell's then new ODI drivers in the first edition, but Network Interrupts includes ten detailed pages on the subject. Chapters that did appear in the first edition have also been expanded: the section on 3Com's Bridge Application Program Interface (BAPI) has additional information on four interrupt functions, to increase the total to 13, and an expanded reference on BAPI return codes. (p. 87-90) Inexplicably, some information is not included in this newer volume--the interrupt function that I used for doing the installation check on MSCDEX 2Fh 1100h is not included in Network Interrupts, but it did appear in the 1991 edition of PC Interrupts.

One odd addition to this newer volume is the inclusion of the portable C language functions on "BSD 4.x Unix sockets" and "Windows sockets." The two chapters do not detract from the total value of the book, but they are largely superfluous; someone programming with BSD sockets would not consult a book on Intel-based interrupts for answers, nor would someone using the Winsock library likely use this book as a reference, as other more comprehensive material, such as the Winsock API specifications (http://www.intel.com/IAL/Winsock2 (http://www.intel.com/IAL/Winsock2)) are available.

With Microsoft pushing for microprocessor independence for Windows NT, the future of network programming is definitely away from having to make direct calls to the registers. But with many DOS/Windows 3.1 machines still on people's desktops, this reference is still essential. Whether you need to write a ten line utility in C that checks if MSCDEX is loaded or you are coding a full-blown networking program in assembly language, you will appreciate the information in this book.

Notes:

[1] For an interesting article on the use of undocumented functions see: Livingston, B. (1992). Undocumented Windows Calls: Deciphering the Charges Leveled at Microsoft. InfoWorld, 14(46), 98-100.

Tony Toyofuku (toyofuku@uci.edu (mailto:toyofuku@uci.edu)) is Electronic Services Librarian and Spanish and Portuguese Bibliographer at the University of California, Irvine.

table of contents | ter issues (/lita/publications/archive/ter)

Browser-mania, Caffeine Hype, and the Plain Truth

by Thomas C. Wilson

Has anyone else noticed the new computing model being touted today? If one believes many of the reviewers and authors for major industry publications, our next operating system will be Netscape, and all applications will be rewritten in Java. Hold on folks! Here comes some more hype...and sorry to say, many high-level administrative decisions are being made based on this dreck. Some of you out there may be

saying, "Hey, wait a minute, isn't this deja vu all over again?" And you're exactly right. Anyone remember all-in-one software packages, like Symphony, or the prediction that all programming would be with object-oriented languages?

Well, now the names have changed, but the same misguided models are being purported. Pundits, as well as technical people who should know better, are pitting Netscape against Microsoft, not so much in the browser wars, but as developers of the next generation operating system. "There's something happening here, what it is ain't exactly clear." [1]

Let's look a little under the surface. What exactly is an operating system? An OS is many things: an interface and set of tools for application developers, an agent between applications and the hardware, a set of utilities to manage the environment, a platform for handling common needs among applications (e.g., printer communication, network interfaces and protocols, port assignments, etc.), a traffic arbiter for multitasking, a user interface or shell, to name a few. The services provided by an OS are complex and necessary. If every developer had to include code in each application to perform port configuration or hardware management, we would have bloated, redundant software and incompatible configurations. In fact, to a large degree, the occurrence of such challenges, along with other limitations, has led to the demise of such OS's as MS-DOS. This rise of OS's that take care of many common services is evident, particularly starting with the introduction of the Mac OS in the early 1980's. Stating this argument is not to say that such advancements in OS's have always been an across-the-board success, just to note that OS's perform certain functions so that applications don't have to.

Now, what exactly is an application? An application is many things: a tool to perform--or assist the user in performing--certain tasks, a utility for creating, accessing, and manipulating information, a means of harnessing the power of the hardware and OS in order to accomplish a particular goal, a specialized environment that draws on the underlying resources of the hardware and OS, to name a few. Applications are software that depends on the OS for services that the OS offers via specified application programming interfaces (API).

Perhaps these distinctions and definitions appear obvious and overly abstracted. I mention them because there is in the mind of some the idea that Netscape will become the next OS of choice. Let's be really straight: Netscape is NOT an operating system. One could make a rational argument that it could become a user interface or shell environment, such as the Korn or C shells in Unix or Program Manager in Windows. An OS, however, it is not! In order for Netscape, the browser, to become an OS, Netscape, the company, would need to divert development resources away from browser enhancement and toward operating system design--not a trivial task. Yes, it could happen, but do we really want that to happen? Or is it likely given the industry today? I think not.

Clearly, we are observing serious browser wars at this time, and in that market, Netscape and Microsoft will go head-to-head. Who will come out on top--if that is even really relevant--is anyone's guess at this time. Some things to note: It will be easier for Microsoft to integrate its browser into its operating system than it will be for Netscape to build an operating system under its browser; Microsoft is pursuing a plan of including its browser as part of the OS package; Netscape has a significant portion of the browser market today, and many users pay nothing for it; and Netscape is a smaller company with a demonstrated history of regular--sometimes overly regular--updates to its product line. Many people are seeing the outcome of this battle as definitive for all of computing, but in computing we have nearly always--in computer terms--had multiple players in the marketplace. Keep in mind that Microsoft and Netscape are not the only browser vendors, and niche markets have a persistence all their own.

The browser wars, however, have clouded the vision of some. If one listens to many respected industry players and observers, one gets the impression that all software applications will be enveloped into browser technology within the next six months--or, as the story goes, those companies will cease to exist. Imagine for a moment what it would be like to get to spreadsheet functionality--since I can't call it spreadsheet software anymore--through a browser: first I load my browser (45 seconds), then I select the menu item on my bookmarks for the spreadsheet applet (5 seconds), then I wait for the applet to load (20 seconds), then I open the file I want to view (10 seconds), then I edit to my heart's content. Does this desktop scenario seem any more efficient than loading a spreadsheet package from the get-go? Actually, the prediction suggests that we will be getting our applets as we need them over the network--so add another 3 to 5 minutes at least for the load time.

Performance and applicability to task are, however, not the only issues at stake here. What about licensing costs? Some people are discussing options for charging cents per use for applets retrieved over the network. In some situations this would make sense (e.g., a searching applet associated with some remote data set); in others the users may be particularly price sensitive (e.g., it could be cheaper to license and install the software locally than to a access it via the network). Software companies are also accustomed to getting payments in larger chunks upfront, prior to use. Changing the economic environment is no guarantee of continued revenue streams; these levels of change are also not adopted over night.

Value to the user is also more than just price; it includes functionality as well. Helper apps cum plug-ins cum applets by their very nature are frequently "lite" versions of what users have become accustomed to accessing. In the case of viewing different file formats, such a technical arrangement may work fine. For the power user, however, such a structure will likely fail. The Belle- of-the-Ball applet development environment of today is Java, a simplified, standardized child of C++. As such, Java applets tend to be small in size, perhaps 100's, or maybe 1000's, of lines of code. While there are many interesting examples of animation, simulation, and illustration on the network today, such applets are far less complex than a commercial word processing package, often requiring hundreds of thousands or millions of lines of code.

Yes, browsers, applets, and Java have their place and show great promise for software distribution, multiplatform support, continuous updating, and client/server computing--all challenges facing the software industry. But is it realistic to assume that all software will be converted to applets written in Java? Or that one browser will replace all operating systems? Since when has the computing industry come to an agreement on a single development environment, a single platform, a single file specification, or a single anything?

Note:

[1] Buffalo Springfield (Stephen Stills). (1969). For What It's Worth.

Tom Wilson, Editor-in-Chief TER, TWilson@uh.edu (mailto:TWilson@uh.edu)

table of contents | ter issues (/lita/publications/archive/ter)

CALL FOR BOOK REVIEWERS -- TELECOMMUNICATIONS ELECTRONIC REVIEWS

Telecommunications Electronic Reviews needs qualified reviewers for books related to telecommunications and networking. There are many books available for review now.

The primary function of TER is to provide reviews of and pointers to telecommunications and networking resources, both print and electronic. The topics covered may include, but are not limited to, specific telecommunications and networking technologies; hardware and software; network operating systems; network applications; management tools and utilities; technical management issues; training and personnel issues; organizational, regional, and national networking; library perspectives; and research and development.

If you are interested in being considered as a reviewer, please e- mail a brief resume which speaks to your qualifications in this area, samples of your writing, and information about which topics you'd like to review to Book Review Editor, Pat Ensor, PLEnsor@uh.edu (mailto:plensor@uh.edu).

ter issues (/lita/publications/archive/ter)

About TER