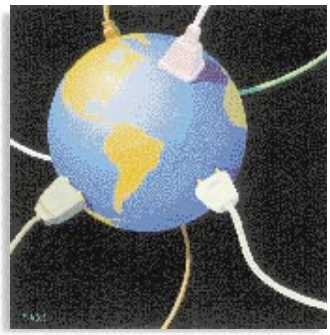
The Trend Toward 64 Bits



Each month, the TSC examines a key emerging technology or its use. This time, we evaluate the transition to 64-bit systems.

In 1964, the computer industry was "revolutionized" by the introduction of the 16-bit minicomputer. In 1978, the minicomputer had started the transition to 32 bits. By 1994, the transition to 64 bits was under way. There appears to be a manifest constant of about 15 years for each of these "epochs": Five years are expended in market establishment, five years for rapid product growth and a further five under the pressure of product insufficiency.

The first Unix system provided 16-bit memory addressing and 16-bit disk adddressing. By the mid-1970s, the 16-bit file programming interface had expanded. The "new" interface increased the maximum size of a file to 31 bits, allowing files to have 2GB of addressability (meaning that it gives an application the ability to move to any given address or position within a file up to 2GB in size). Because the developers of Unix believed it was important to simplify rather than complicate, access to the file system was provided through the same programming interfaces used to access individual files. This simplification enforced on file systems the same limitation that constrained ordinary files to a maximum size of 2GB, which is no longer sufficient to support enterprise-class applications.

Like the transition from 16 to 32 bits, the evolution from 32 to 64 bits is happening only as the business climate requires it. Niche markets have already achieved 64-bit development because of a need to solve narrowly focused problems. In a commodity business (a status the minicomputer industry has achieved), splashy features are not practical.

Although database vendors have allowed data sets to grow beyond 2GB by joining multiple files into a single, logical data set, this awkward solution is visible to the user. For several years, applications suppliers have been requesting file sizes beyond the 32-bit limitations present on current Unix systems. The first implementations appeared as large file systems that provide support for 2GB files on 4GB file systems. While this enables better use of the disks that are appearing on the market, it does not meet the needs of the applications.

Partial Transition

Distributed systems have already accommodated the expected growth to 64-bit files. Both the Network File System (NFS) and the Distributed Computing Environment (DCE) provide protocols that support 64-bit files. Without file system interface support, this has gone largely unnoticed, but the stage is set.

If the stage is set and the pressure is building, what is limiting the development of 64-bit systems? The high-volume, low-cost nature of commodity markets inevitably leads to standardization, and customer investment mandates release-to-release compatibility. The combination of standardization and compatibility can inhibit progress. While there is a customer need to extend to 64 bits, these same customers do not want to be forced to invest in systems that obsolete their present investments.

To make a transition from one paradigm to another, a migration strategy may include temporary programming interfaces. However, system vendors are reluctant to introduce programming interfaces that have limited lifetimes. The reason for this reluctance is not because such interfaces are difficult to develop but because they are difficult to obsolete. Without a transition period, a quantum leap from one paradigm to another can leave customer investments in the lurch.

A set of programming interfaces has been developed by independent software vendors (ISVs) and system vendors that permits access to 64-bit files on systems that "naturally" support 32-bit files. These interfaces, informally known as the Large File Summit interfaces, are intended to provide a transitional environment as applications are migrated from the existing 32-bit systems to the future 64-bit systems. Because the pressure to develop these interfaces came largely from the ISV community, their appearance on systems will be driven by the availability of the applications that need them. In general, end users will not request these interfaces, but they will require the applications that need them. These interfaces are not expected to appear in any programming standards because of their transitional nature, although they have been developed with the same degree of attention applied to Posix and the Single UNIX Specification.

Why 64 Bits?

The need for 64-bit computing can be generalized into two classes of applications: those that require large files and those that require large memory. Database servers are an example of those that need large file access. While database servers have been breaking databases into multiple files to exceed the 2GB limitation imposed by most systems, this is only a stopgap measure. As soon as systems with 64-bit files are available, database vendors will port to these platforms and expect to achieve tremendous improvements in performance. Database programs that cache records also benefit from greater memory.

On the other hand, simulation programs exemplify those that need large memory access. Recent advances in the simulation of mechanical systems have sig-

By Larry Dwyer

nificantly improved performance at the expense of memory. The more memory that is present, the more sharply the customer can tune the accuracy of the simulation while retaining its previously established elapsed time.

Applications that are not expected to require either large memory or large files include spreadsheets, word processors, viewgraph generators and mail user agents. This means the pressure to migrate to 64 bits is neither universal nor compelling for the class of applications most frequently used by personal computer users. While some systems introduced to satisfy the large file/memory business will be exclusively 64-bit, some will simultaneously support both 32- and 64-bit programming interfaces to enable both the high-performance applications (e.g., databases) and the commodity applications (e.g., word processors).

The introduction of files larger than 32 bits will bring new challenges to file archiving. The "classic" model of backup assumed the system was populated by a large number of small files. It is possible to back up these files in groups at different times, thereby avoiding unnecessary downtime during the backup. As files grow, it is not possible to take the system down to back up the "files," because the entire file system may be populated with a single file.

What used to be a file is now a *record*. Because the system does not know about the structure of a file, it is no longer possible for general file system archive utilities to provide a reasonable backup environment. It will be up to the application to provide archiving capability. This represents an opportunity for collaboration between ISVs who develop large databases and ISVs who develop archive subsystems.

The Data Model

Software standards are designed to be architecture neutral. If this were not the case, standards would change each time a conflicting hardware architecture was introduced. Architecture neutrality of a specification lets a vendor implement the specification on an arbitrary hardware architecture. The specifications described in Posix, the Single UNIX Specification and ANSI C are, by definition, architecture neutral. Unfortunately, these specifications were developed during a time when systems were dominated by 32-bit architectures. This dominance led to the inadvertent inclusion of architecture-specific interfaces. While the various standards can be compatibly implemented on 32-bit systems, they are more difficult to implement on 64-bit systems.

The pressure to migrate to 64 bits is not universal.

The dominant 32-bit programming model assumes that the size of an integer equals the size of a long integer and also equals the pointer size. This is known as the ILP-32 (IntegerLongPointer-32-bits) model. If the natural progression to 64 bits is followed, one might expect the 64-bit data model to be ILP-64. Unfortunately, it is not possible to write a program that strictly conforms to ANSI C and accesses all of the natural integer data sizes (8 bits, 16 bits, 32 bits, 64 bits, pointers) unless one breaks the assumption that the size of a long integer equals the size of an integer. To accommodate strict conformance to ANSI C, a different data model has been chosen. This allows the application developer to create data types of 8-bit integers (char), 16-bit integers (short), 32-bit integers (int), 64-bit integers (long) and 64-bit pointers (*). This data model is designated LP-64 (LongPointer-64-bits).

With the realization that LP-64 is the correct answer, DEC, Hewlett-Packard, IBM, Novell, SCO and Sun Microsystems jointly endorsed the LP-64 data model and have generated recommendations to X/Open to amend the Single UNIX Specification to be "more" architecture neutral. Less than 20 changes were needed to improve the core specification. (These changes will appear in a future version of the specification.)

Although the LP-64 decision does not equate directly into a standard, the companies that endorsed the LP-64 decision are treating it as if it were a standard and will introduce 64-bit systems based on the LP-64 data model.

What It Means to ISVs and End Users

Applications that must be migrated to 64 bits will have to be inspected to remove dependencies on ILP-32 versus LP-64. All uses of "int" and "long" have to be aligned with the data definition needs of the application. This inspection and correction is not simple and could introduce unnecessary errors into the code. The only applications that are expected to be upgraded are those that cannot offer competitive solutions without resorting to 64-bit pointers (such as databases and simulators).

It is expected that many system vendors will provide systems that simultaneously support ILP-32 and LP-64. This allows applications that do not need to be migrated to 64 bits (such as word processors) to avoid the unnecessary changes required of an LP-64 application. On systems that don't support both ILP-32 and LP-64, applications will have to be modified.

Some applications that take advantage of the emergence of 64-bit systems are available now, and more will be ported to the 64-bit environment as business needs are encountered. Customers will continue to use 32-bit systems to run the applications that do not yet need 64-bit memory or 64-bit file systems. As hinted earlier, the market presently is in the second year of the five-year cycle needed to establish the market. By 1999, 64-bit systems should be into the rapid-growth phase for a product. Perhaps by 2011, we will be going through a similar exercise to achieve the 128-bit paradigm.

It should be remembered, though, that nature abhors a vacuum. As 64-bit systems become available, they will create a vacuum that will draw application developers to solve problems in ways that could not have been considered when systems were limited to 2GB. IT

Larry Dwyer *is a system architect for Hewlett-Packard Co. in Cupertino, CA. He can be reached at ld@hpldkbd.cup.hp.com.*