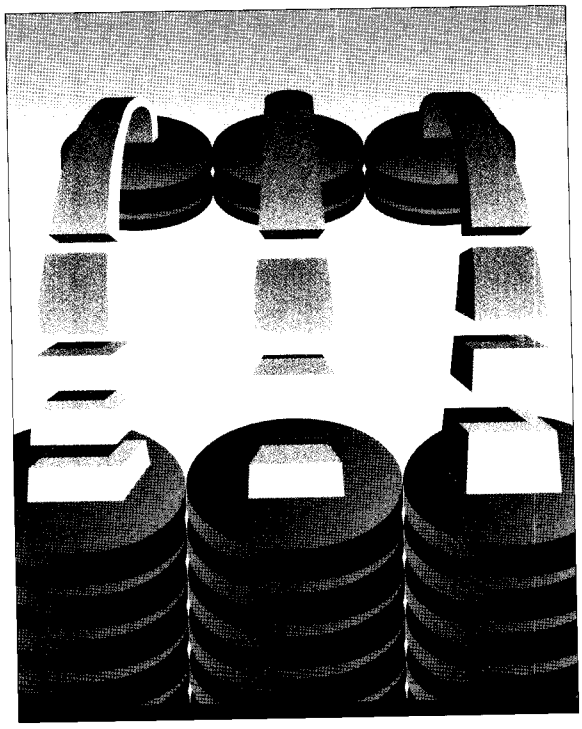
T A N D E M

SYSTEMS REVIEW

SPRING 1994



D-Series Releases of Guardian 90

Migration Planning

Application Code Conversion

Technical Information and Education

Product Update

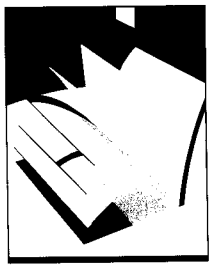
T A N D E M

SYSTEMS REVIEW

VOLUME 9, NUMBER 2

SPRING 1993

TANDEM



Editor's Note

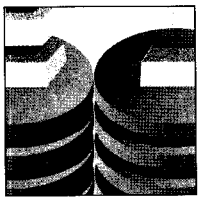
The D-series releases of the Tandem Guardian 90 operating system significantly increase the architectural capacity and practical limits of Tandem systems. In particular, the D-series releases allow more processes to run concurrently on a given processor than do previous releases. Users can thus take fuller advantage of the hardware capacity of Tandem systems and have the potential to increase the number of user applications on their existing Tandem equipment.

This issue of the *Tandem Systems Review* includes three articles on the D-Series Guardian 90 operating system that will help application designers and developers to implement the D-series software. These articles describe the D-series changes and their effects on system-level limits, the considerations and process when migrating from a C-series system to a D-series system, and the suggested methods for converting code in nonprivileged application programs from a C-series to a D-series system.

Please take a few moments to fill in the "Reader Survey" questionnaire at the end of this book. We would like to receive your comments so that we can better meet your technical information needs. *–SWT*

Tandem Systems Review is published quarterly by Tandem Computers Incorporated. All correspondence and subscriptions should be addressed to *Tandem Systems Review*, 10400 Ridgeview Court, Loc 208-65, Cupertino, CA 95014.

Subscriptions: $75.00 per year; single copies are $20.00. Detailed subscription information is provided on the subscription order form at the end of this book.



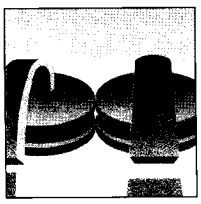
D-SERIES GUARDIAN 90

6 A Designer's Overview of the D-Series Guardian 90 Operating System

Wendy Bartlett

14 Migration Planning for D-Series Systems

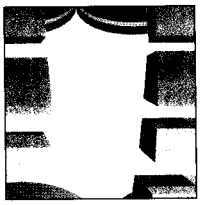
Sue Kuukka



26 Application Code Conversion for D-Series Systems

Ken Liu

DEPARTMENTS



2 Product Update

40 Technical Information and Education

46 Index of Articles

ProductUpdate

NonStop-UX Based Software

NonStop-UX Operating System, Release B10

January 1993

Release B10 of the Tandem NonStop-UX operating system provides a number of enhancements and new features. These include 5-millisecond granularity for high resolution timers; 1-millisecond real-time clock; bundled Multiple National Language Supplement (MNLS) software; fast reboot; support for 2-gigabyte disks; support for R2000-based processors; improved file-system operations; and full functionality of the sysadm facility.

Communications and Networking Products

LAN Nonintelligent Ethernet and Token-Ring Adapters

December 1992

Tandem now offers two groups of new LAN adapters. The new nonintelligent Ethernet adapters are high-performance, low-cost units available in 8- and 16-bit ISA and Micro Channel bus versions. The token-ring adapters are available in versions to support the ISA, EISA, and Micro Channel bus architectures.

LAN Manager 2.1

December 1992

LAN Manager 2.1, which replaces the previously offered LAN Manager 2.0 products, includes a number of installer and usability improvements and now also provides NetWare interoperability. LAN Manager 2.1 is shipped with OS/2 version 1.3. Entry-level and unlimited-user versions are available. Also available are 10-user and unlimited-user upgrades, upgrades from LAN Manager 2.0, and Mac Connectivity and Remote Access Server add-ons.

Tandem Access/Hub

December 1992

Tandem Access/Hub is an entry-level platform for the Tandem Access/One family of local area network hub products. The Access/Hub product allows users to connect up to a total of 12 PCs, PS/2 systems, Macintosh computers, and UNIX workstations over a 10Base-T (twisted pair) Ethernet network. The product's cascade function expands the Ethernet network up to 132 ports.

Tandem Access/Hub is offered in two versions, an entry-level unmanaged hub and a Simple Network Management Protocol (SNMP) managed hub. The unmanaged hub is ideal for small, departmental or remote-site Ethernet networks and can be easily upgraded to support SNMP with a field-installable daughter card. The SNMP version enables users to efficiently monitor, control, and configure the Access/Hub using an SNMP manager. Both products are available in 115-volt and 230-volt versions.

Workstation and Terminal Products

PSX NP3/25SL Notebook Computers

January 1993

The PSX NP3/25SL notebook computers incorporate the latest in notebook computer technology and provide a comprehensive modular upgrade path. These notebook computers are available in preconfigured versions with either a monochrome or a color display. The PSX NP3/25SL computers include, as standard features, a removable hard disk drive; 4 megabytes of RAM, with support for 20 megabytes; two credit-card-sized PCMCIA expansion slots; a battery life of four to six hours; 64 kilobytes of cache memory; MS-DOS and MS-Windows; a SmartPoint trackball mouse; and a leather carrying case.

The U.S. models also include a 14.4-kilobaud Data/Fax send-receive modem. International versions are available to support the German, French, and U.K. languages. Other languages are available on request with some lead time.

PSX EP4/66D and EP4/33s Workstations

January 1993

The EP4/66D and EP4/33s workstations offer increased processing speeds over previous EP-series models and provide all of the standard EP-series features. These include an enhanced video subsystem, greater memory expansion, cache memory support, and a 3.5-inch diskette drive. In addition, the internal hard drive storage capacity has been increased to support up to 2 gigabytes internally by using the 3.5-inch SCSI disk drives.

Panasonic 14-inch SVGA Color Monitor

January 1993

The Panasonic 14-inch SVGA monitor is a low-radiation, 230-volt monitor available for sale outside the U.S. and Canada. This monitor complies fully with the strict MPR-II standards for electrical and magnetic emissions. It supports 800 x 600 noninterlaced resolution and features an antiglare, tinted faceplate.

AST DOS

January 1993

Tandem is now including AST DOS in place of MS-DOS in all system shipments to Europe. AST DOS is now also available on selected U.S. systems.

Storage Products

3215-1 Second-Channel Feature for 5180 Tape Cartridge Subsystem

November 1992

The 3215-1 second-channel feature for the 5180 and 5180 ACL tape cartridge subsystems allows a single 5180 subsystem to be shared between two separate Tandem systems, thereby lowering the cost of ownership. With the 3215-1, a single 5180 subsystem can be connected to two supported Tandem systems or two separate processors in the same system. The 3215-1 feature is available for all Tandem systems that currently support the 5180. These include the Cyclone system, Cyclone/R system, CLX2000 system, and CLX800 system. All installed 5180 subsystems can be field-upgraded to incorporate the 3215-1 second-channel feature.

Security and POS Products (Atalla)

A6000 Network Security Processor

December 1992

The A6000 Network Security Processor is a hardware-based system that provides cryptographic security in IBM host environments. The A6000 automates cryptographic processing in a logically and physically secure environment for maximum protection. The hardware-based system design eliminates the security risks inherent in software-based approaches.

The A6000 performs such cryptographic functions as PIN Security (translation, verification, and encryption); message authentication; data encryption/decryption; key management (key generation, distribution, and translation); VISA unique key per transaction (derived key); and VISA CVV, MasterCard CVC (verification and generation). Connection to IBM systems is provided through a dedicated channel interface called the Atalla Channel Controller (ACC). Each ACC supports up to three A6000 Network Security Processors.

A Designer's Overview of the D-Series Guardian 90 Operating System

The D-series releases of the Guardian™ 90 operating system expand a number of system limits, including the number of processes that can run concurrently in a single CPU. Migration to D-series releases can be carried out with a minimum of effort.

The original Tandem™ operating system, designed in 1975, supported a system whose individual processors were much less powerful than any of today's PCs and workstations. It subsequently has evolved to support larger and faster processors, most notably with the addition of 32-bit addressing with the NonStop II system in 1981. However, by the time the Cyclone™ processor was under development, it became evident that the operating system would require extensive low-level redesign in order to rescale it to match the hardware directions of the 1990s. The result of that redesign is the D-series Guardian 90 operating system, whose first production-level release is D10.

The first part of this article describes the D-series changes and their effects on system-level limits. It assumes at least basic familiarity with the C-series architecture. The rest of the article is for those who either are interested in a designer's perspective on redesigning an existing operating system or want to better understand the effects of the new limits on C- and D-series applications. For a more detailed look at migration and application conversion considerations, read the companion articles, "Migration Planning for D-Series Systems" (Kuukka, 1993) and "Application Code Conversion for D-Series Systems" (Liu, 1993), in this issue of the *Tandem Systems Review*.

Background

The designers of any operating system have to make trade-offs between present efficiency and anticipated technology trends when determining the scale of the system. As a consequence, any successful operating system eventually needs rescaling in order to keep up with technology growth curves in processing speed, memory size, and I/O connectivity. The scope of the rework required depends on the structure of the operating system. One need look no farther than the history of IBM's OS/360 for an example of operating system evolution; it has a whole chain of larger-scale successors, including MVS, MVS/XA, and MVS/ESA.

The original Tandem operating system supported a very small-scale system by today's standards: its disk storage was measured in tens of megabytes and its memory in hundreds of kilobytes per processor. The machine could address only four data spaces of 128 kilobytes each, which imposed severe constraints on both system and application software. In the ensuing 15 years, Tandem processor power has grown by more than an order of magnitude, disk volume size has grown by over two orders of magnitude, and typical memory size has grown by nearly three orders of magnitude.

Many system limits were expanded during this period. The most important changes were the addition of 32-bit addressing, introduced in the A series with the NonStop II processor, and support for multiple code spaces, introduced in the B00 release. These features made it possible to greatly expand both system and application code and data space.

A different set of limits had to be raised to support the hardware growth curves of the 1990s. These limits, which have to do with the amount of concurrent activity that can be supported within a processor or system, permeate the lower levels of the system in the form of data structure and interface restrictions. In some cases, primarily involving the maximum number of concurrent processes per CPU, they also are visible to application software. Raising these limits required a substantial redesign of kernel data structures, internal interfaces, and interprocess message formats, along with the addition of application-level interfaces capable of supporting the new limits.

Nonprivileged programs can, in almost all cases, not only run on D-series releases without change but also take some advantage of new capabilities. However, rebinding, recompilation, or recoding may be required in order to take full advantage of having more concurrently-executing processes per processor. In a few cases, minor coding changes may be necessary in order to comply with the enhanced support of the D-series for ANSI language standards.

Rearchitecting vs. Redesigning the System

The operating system's basic architecture has not been changed in order to support higher limits. It retains its original client/server structure, based on processes communicating through messages. This architecture provides an effective platform for distributed computing. Similarly, the anatomy of the D-series software, in the way it partitions functions among specific processes and interfaces, differs little from that of the C series.

The redesign's focus was on resource limits that still were sized for the original 16-bit machine, as extensions made in the B- and C-series releases had formed only a partial transition to a 32-bit operating system. For example, many system data structures continued to reside in system data space, which meant that they all had to be allocated from 128 kilobytes of storage. Because of this fixed limit, to increase the number of elements of one kind, it was necessary to reduce the number of elements of another kind. This was not a desirable situation for accommodating growing applications.

Table 1.
Comparison of C-series and D-series limits.

System resource	C-series limit	D-series architectural limit	D-series practical limit
Processes per CPU (PCBs)	256	64,000	Up to 2500
Devices plus named processes per system (DCT entries)	4,096	> 64,000	Up to 34,000
Concurrent messages per CPU[1]	< 23,000	> 64,000	Up to 33,000
Subdevices per device	255	64,000	Subsystem-specific
Opens per subdevice	16	64,000	Subsystem-specific
Enscribe opens per volume	4,096	32,000	Well below 32,000

[1]Concurrent messages in C-series releases = link control blocks (LCBs) + extended link control blocks (XLBs).
Concurrent messages in D-series releases = message quick cells (MQCs).

Redesigning the Operating System

The requirements, external design goals, and internal design goals identified at the start of the project combined to set its scope and framework. That framework can be summarized as follows:

- Raise system limits where needed.
- Minimize migration effort.
- Build the operating system foundation for the rest of the 1990s.

The next section describes the project's requirements and the way it met them. This is followed by sections on external design goals and internal design guidelines that were considered to be important for the success of the D-series releases.

Requirements

This section discusses the high-level requirements for the D-series releases. The requirements addressed both the functionality to be provided and migration considerations.

System Resource Limits

The primary limits to be raised were:

- Number of concurrent processes, or process control blocks (PCBs), per CPU.
- Number of devices, including disk volumes, plus named processes or process pairs per system (destination control table [DCT] entries).
- Number of concurrent outstanding messages per CPU.
- Number of subdevices, for those devices that support them.

Table 1 shows the existing C-series and new D-series limits for these and several other resources.

The current D-series implementation limits are approximately an order of magnitude higher than the C-series limits for number of PCBs and DCT entries. The architectural limits for these resources are even larger, but currently are out of reach due to practical limits imposed by the availability of other resources such as virtual address space and physical memory.

The limit on concurrent outstanding messages per CPU (nearly 33,000, given enough physical memory), is somewhat higher than in the C-series releases, and in the D-series releases, message control blocks are allocated from significantly larger pools than the C-series link control block (LCB) pool. In C-series releases, most I/O processes contended to allocate LCBs from a single pool that seldom contained more than 1,000 elements. This contention artificially constrained message-system control block availability.

The C-series limitations on the number of subdevices per device continue to be appropriate for most existing device types. However, a few device types, such as the X.25 Access Method, needed to be able to support larger configurations and now have that capability in the D-series releases.

Code Compatibility

No source code changes or recompilation were to be necessary in order to run a nonprivileged program on a D-series system, provided that the program did not need to take advantage of higher limits.

In D-series terminology, an unconverted program refers to a C-series program that has not been changed or recompiled for running under a D-series release. Almost all unconverted, nonprivileged programs will continue to run correctly when moved to a D-series system and used within existing PCB limits. The primary cause of problems for TAL programs is reliance on unpublished procedures or undocumented side effects of published procedures. C-language and (rarely) COBOL85 programs may require minor source code changes for compliance with ANSI standards. More details can be found in D-series documentation (see the list at the end of this article) and the article "Application Code Conversion for D-Series Systems" (Liu, 1993), later in this issue of the *Tandem Systems Review*.

There was no requirement that the D-series releases support execution of C-series privileged code. Supporting C-series privileged interfaces and data structures in the D-series operating system, even if restricted to use within existing limits, would have greatly magnified the scope and complexity of the project. Any short-term gain from minimizing the need to convert nonkernel privileged software would have been paid out many times over in longer delivery schedules, higher defect rates due to the added complexity, and support costs for interfaces whose use would dwindle over time.

Operational Coexistence

Operational changes required to run a D-series system were to be kept to a minimum. In some cases it proved necessary to modify command syntax and display formats to reflect the higher limits. There also are process-related changes in the TACL™ (Tandem Advanced Command Language) at the macro and built-in level that apply even when TACL is used within existing limits, due to its role as the operational interface to Guardian process control.

Most D-series operational changes are unrelated to the new limits. Rather, they are a result of the ongoing migration from the older Communications Utility Program (CUP), Communications Management Interface (CMI), and network monitor (NETMON) programs to the Subsystem Control Facility (SCF) program as the command interface of choice for I/O processes.

Network Coexistence

D-series nodes had to be able to coexist in the same Expand™ and FOX™ (Fiber Optic Extension) network with C-series nodes. This requirement was fully met. Users are not required to simultaneously migrate all nodes, or even all FOX nodes, to the D series.

External Design Goals

The following two goals were regarded as important even though they were not fully attainable. They were expected to be met, insofar as it was at all practical to do so.

Transparent Access to New Limits

No changes were to be required in order to run a nonprivileged program on a D-series system and have it be able to take advantage of new limits. This goal has been met for most limits. However, as was known at the outset of the project, it was technically infeasible to fully meet the goal of providing transparent access to more processes per CPU.

Providing access to higher limits for programs running on a remote C-series node was not possible because the formats of messages sent from C-series nodes to other nodes did not, in general, support the new limits. Even if the formats had supported them, the C-series software responsible for interpreting the contents of network requests and replies would not have been prepared to handle the higher values.

Secondary System-Resource Limits

Raising limits on a number of system resources was desirable, but prevented for reasons related more to resources and timeframes than to technical infeasibility. Limits were raised in several areas, including supporting more concurrent Enscribe file opens and locks and more temporary files per disk volume.

Internal Design Guidelines

Internal design guidelines were developed early in the project. Their purpose was to provide a consistent framework for making design trade-offs within the constraints that the fixed requirements imposed.

Operating System Foundation for the 1990s

The D-series operating system must provide a solid base for future growth. To achieve this, it must be of high quality, extensible, and no more complex than necessary.

One form of foundation building was to reduce dependencies among privileged products by providing a better-architected interface between the operating system kernel and its clients. Another was the selective widening of fields for items whose limits were not being raised at this time. The main examples of this are node and CPU identifiers, which pervade most data structures, message formats, and interfaces. There are no plans to raise the current limits (255 nodes in an Expand network and 16 CPUs per system), but widening the fields greatly reduces the magnitude of the effort if a decision should ever be made to raise them.

Design Cohesiveness

It was important not to bend over so far backward in maximizing compatibility between existing documented procedures and their D-series equivalents that the old limits would look like defaults to someone designing a new application. The D-series interface should look like it was designed around the new limits rather than the old ones. This allows developers to write new applications in a consistent manner, which becomes the norm over time as older programs evolve and are replaced.

This approach resulted in some new procedures having different parameter default values than their old equivalents. When converting an application, it may be necessary to supply an extra parameter when calling a D-series procedure in order to invoke the behavior that was the default for its superseded equivalent.

Identification of Compatibility Problems

Inevitably, problems will arise due to attempts to access higher limits through the existing C-series interfaces. The developers needed to anticipate likely problems and make it as easy as possible to diagnose them when they occur.

Attempts to access a process whose identity falls outside of the existing limits are the most frequent classes of compatibility problems. Depending on the circumstances, the error usually is reflected in one of several ways: the target process may appear to be invisible, the caller may be returned a specific new error code indicating the problem or, in extreme cases, the caller may trap.

Support for New Limits in Tandem D-Series Software

Fully converting all Tandem D-series products would have provided users with a uniform view of the new system limits. However, a careful evaluation of the development resource investment required to achieve this goal compared with its benefits caused Tandem to choose a more targeted conversion strategy. By doing only selective conversions it became possible to devote adequate resources to other high-priority projects.

In order to make the best use of their own development resources, users are likely to take a similar approach when determining the extent of their own D-series conversions. To aid in this process, the Tandem D-series manuals and courses provide education in evaluating conversion alternatives and their associated benefits and costs.

A Closer Look at Migration to D-Series Releases

Unless a system needs substantially more than 256 processes per CPU, migration to the D series may not require any application conversion. Moving converted Tandem and third-party applications into high PINs may provide adequate PCB relief. Selective application conversion should be considered only if further relief is needed. Even then, most users are unlikely to need to perform more than minimal conversion.

Use of the new D-series interface is recommended when developing applications targeted only for D-series releases. This interface provides full access to the new system limits and is the foundation for functional enhancements in future releases.

The following sections take a more detailed look at those aspects of the design that have the greatest bearing on application migration and on coexistence within a network containing both C-series and D-series nodes.

The Existing Procedural Interface

The requirement that unconverted application code be able to run on a D-series system (see the section Code Compatibility, earlier in this article) meant that C-series published, nonprivileged interfaces must continue to exist and function as before, at least when used within the C-series system limits. Supporting this functionality proved to be a nontrivial effort; the procedure bodies had to be completely reimplemented due to changes in the D-series operating system kernel and data structures. The code within these procedures has to mediate between the application's C-series view of the world and the actual D-series underpinnings, which is at times an extremely complex operation. This system-level code also has to be in a sense bilingual in order to communicate properly with both remote D-series nodes and remote C-series nodes.

Extensions to the Procedural Interface

It was not obvious what to do about new limits that could not be expressed through the existing interface. Given the choice between adding parameters to an already complex interface and adding a new set of procedures designed around the new limits, the designers opted for the latter approach. Lack of space in the PIN field was the catalyst for taking that path, because of its impact on procedures that used process identifiers. Once that decision was made, the next step was to develop a set of standards and guidelines for interface design. The result was a set of new procedure definitions that are much more consistent than their predecessors.

D-Series Terminology

One important but often-overlooked design task is developing and agreeing on terminology. The project team found it hard to carry out a cogent discussion of design alternatives until there were commonly-understood terms that differentiated between old limits supported through the existing interface and new limits supported only through the extensions.

The most important new terms distinguish between forms of process identifiers that can be expressed within the existing interfaces and those that can be expressed only within the new D-series interfaces. Two terms of particular importance in the following sections are:

- *High PIN*: a process identification number (PIN) that has a value of 256 or higher.
- *Low PIN*: a PIN that has a value in the range of 0-254.

PIN 255 is never assigned to a running process in a D-series system.

For a more complete set of definitions, see the article "Migration Planning for D-Series Systems," later in this issue.

Because the formats of C-series process identifiers and process file names were externalized, a minor conversion effort is required for clients of the new procedures. C-series process identifier formats were documented to the bit level and not usually represented as any kind of data structure within application code. Procedural support for manipulation of process identifiers and process file names was minimal. As a result, C-series code that had to handle these items had hard-coded knowledge of their contents and manipulated them accordingly.

To make converted code more maintainable and extensible, D-series process identifiers are defined as black boxes of fixed length but undocumented format. The D-series programmatic interface extensions include procedures to decompose them. There also are new procedures that can manipulate all forms of external-form file names. A program that takes advantage of these new procedures need not understand or embody the details of file-name or process-identifier formats.

Use of New Limits on PINs

Most unconverted programs cannot be run safely at a high PIN, due to direct or indirect calls to a procedure such as MYPID. Minor code changes (TAL programs) or recompilation with the D-series compilers and their new Common Run-Time Environment (COBOL85, FORTRAN, C, PASCAL) are all that is usually required in order to run a program at a high PIN. However, as this section will demonstrate, the impact of running a program at a high PIN may well be greater on the programs that interact with it than on the program itself.

To minimize unpleasant surprises, the operating system will not run an object file at a high PIN unless the file has been marked as being runnable at a high PIN (that is, has its HIGHPIN flag set ON). There is no external way to force the system to run an unmarked object file at a high PIN. In the interest of safe computing, the program that is creating the new process also must affirm its ability to support a high-PIN offspring and its resultant new-format process identifier. It does so by calling the new D-series PROCESS_CREATE_ procedure rather than NEWPROCESS or NEWPROCESSNOWAIT.

A few other factors, such as availability of a high PIN, affect whether a new process actually runs at a high PIN. (See the D-series documentation listed at the end of this article for details.)

If a process creator asks that the new process run at a high PIN but the new process is given a low PIN instead, this is not treated as an error. The new process will function correctly; the only negative effect is that it will consume a limited resource (a low PIN).

Interprocess Communication Considerations

A thorough understanding of the interprocess communication restrictions associated with high PINs is a prerequisite to development of an effective migration plan if use of high PINs is required. The D-series articles in this issue of the *Tandem Systems Review* only provide an overview; the new D-series manuals fill in the details.

For the purpose of discussing D-series interprocess communication rules, the terms *process* and *interprocess* communication apply to all processes within the system, whether they are nonprivileged or privileged. The reason for stressing this point is as a reminder that I/O devices and disk volumes are implemented as processes and hence are subject to the same interprocess communication rules as other processes.

Communication Among Processes Running on D-Series Nodes. In most circumstances, an unconverted process cannot reference high PINs because they do not fit in the C-series interface's data structures such as process IDs. However, there is an important exception that works for a large class of requester programs. It is possible for an unconverted process running on a D-series node to use the existing OPEN procedure to open a high-PIN process as long as the latter is named. This usage works because it is not the caller's responsibility to specify the actual CPU and PIN of the target process. Instead, the system uses the name to look up the CPU and PIN kept in its own tables.

Unconverted server programs cannot accept requests from high-PIN processes. However, this situation is easily changed for most TAL servers by using Tandem's Binder utility to set the HIGHREQUESTERS object-file flag to ON. The article "Application Code Conversion for D-Series Systems" (Liu, 1993), later in this issue of the *Tandem Systems Review*, explains in more detail when this approach is effective and when a more thorough conversion is in order.

Interprocess Communication and High PINs: Mixed Networks. A requester process running on a C-series node cannot open a high-PIN process. This is true even when the high-PIN process is named, as the opener's system may need to specify the CPU and PIN of the target process and there is no way to fit a high PIN in the C-series network messages. For similar reasons, it also is impossible for a process running on a C-series node to create a high-PIN process on a D-series node.

Conversely, a process running at a high PIN cannot open or create a process on a C-series node. Doing so would require that the identification of the high-PIN process be passed to the target C-series node, which is not possible.

Because of these constraints, management processes that may need access to any process in a mixed network must be run at low PINs on D-series nodes. They must run at low PINs so they are visible to C-series nodes and must run on D-series nodes so they have access to any high-PIN processes that may exist.

Conclusion

Although its most obvious feature is the raising of specific system limits, the D-series operating system should rightly be regarded as an evolution in scale from its predecessors. As an integral part of its redesign, the lower levels of the system have been restructured to improve its reliability and extensibility. The result is a robust platform for user applications that increases application design flexibility by significantly reducing the need for cognizance of system resource limitations.

For Further Information

Detailed, product-specific information appears in the D-series product manuals and in individual product softdocs. The following documents provide an overview of the D-series Guardian releases.

D10.00 Installation Guide. 1993. Tandem Computers Incorporated. D-Series Site Update Tape (SUT). Subvolume Y9230D10, file INSGUIDE.

D-Series Product Enhancement Summary. 1993. Tandem Computers Incorporated. D-Series Site Update Tape (SUT). Subvolume Y9230D10, file DENHANCE.

D-Series System Migration Planning Guide. 1993. Tandem Computers Incorporated. Part no. 51439.

Guardian 90 Operating System Application Conversion Guide. 1993. Tandem Computers Incorporated. Part no. 69808.

Introduction to D-Series Systems. 1993. Tandem Computers Incorporated. Part no. 63950.

References

Kuukka, S. 1993. Migration Planning for D-Series Systems. *Tandem Systems Review*. Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

Liu, K. 1993. Application Code Conversion for D-Series Systems. *Tandem Systems Review*. Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

Acknowledgments

Literally hundreds of people contributed to making the D series a reality. It is impossible to name them all here, but I would like to recognize the people who set the framework, kept it cohesive as day-to-day decisions were made, and supported the team through the design phase: Nelson Bolyard, Nitin Donde, Garry Easop, Dave Kinkade, Eric Nellen, Alan Rowe, Robert Shaw, Frank Sheeman, Randy Shingai, Cindy Sidaris, Jim Smullen, David Wong, and especially Rich Larson.

Wendy Bartlett is the lead designer of the D-series Guardian 90 operating system. She has been with Tandem since 1978. Before joining Tandem, she worked for other companies as an applications and systems programmer. She has an M.S. degree in Computer Science from Stanford University.

Migration Planning for D-Series Systems

To support the increased capacity of Tandem™ systems, the D-series releases of the Guardian™ 90 operating system have raised many system limits and improved the way the Guardian 90 system manages processes. D-series systems allow more processes to run concurrently in a CPU than C-series systems do. To take advantage of this increased capacity, users need to plan their migration to D-series systems. For most users, migration is a simple and straightforward process.

The key to a successful migration is to develop and implement a migration plan. Tandem recommends that users first analyze their requirements for migration and then migrate to the D-series system in phases. A phased approach allows users to focus on one aspect of migration at a time, providing the benefits of the D-series enhancements while minimizing the effort at each phase. Migration tools are available to help users migrate to the D10 release.

Most users can raise system limits sufficiently for their needs without going through all the migration phases described in this article. For example, only a few users will have to convert application code.

Users should follow the phased approach when they migrate the first one or two systems in their network. After they gain experience in migration, most users can abandon this approach and migrate other systems in a single step if they choose to do so.

This article describes the process of migrating a C-series system or network of systems to the D-series operating system. It briefly explains each phase of migration, outlining the benefits and costs of migration at that phase. A hypothetical user profile demonstrates the limits relief a system can obtain at each phase.

The article is aimed at system managers and system programmers responsible for planning and implementing the migration to D-series systems. It assumes that users have read "A Designer's Overview of the D-Series Guardian 90 Operating System" (Bartlett, 1993), the preceding article in this issue of the *Tandem Systems Review*. In addition, users should be familiar with the terms defined in the accompanying sidebar before reading the rest of the article.

Defining Migration

D-series migration is the process of moving one or more systems from the C30 release of Guardian 90 to a D-series release. Tandem anticipates that users who need relief from the C-series limits, or who need new products being introduced on the D10 release, will migrate from the C30 release to the D10 release. Most others will migrate from the C30 release to the D20 release. This article focuses on migrating to the D10 release of the Guardian 90 operating system.

During migration planning, users define their hardware requirements, operational requirements, and potential conversion requirements for the migration. The work needed to migrate will vary according to the user's requirements. Many users only need to make operational changes; a few will have to perform some type of conversion.

To ensure that their hardware is compatible with the D-series system, users may have to upgrade discontinued hardware products. The Software Product Maintenance Levels document, available on the D10 release Site Update Tape (SUT) or through the InfoWay™ online information service, identifies the hardware products no longer supported in the D-series environment.

Users may have to change certain operational procedures and utilities, including the SYSGEN configuration file and portions of their TACL™ (Tandem Advanced Command Language) macros and command files. They will also have to switch from the Communications Management Interface (CMI) to the Subsystem Control Facility (SCF) to configure and control some data communications subsystems.

Most users do not have to convert any application programs to achieve their migration goals. However, a few users must convert portions of their application software to migrate successfully to a D-series system.

Conversion includes any change that allows a program to take additional advantage of the D-series enhancements. Even when conversion is needed, most users can achieve their migration goals simply by recompiling their application programs. Only a few will have to change their application source code. The article by Liu in this issue of the *Tandem Systems Review* (Liu, 1993) describes application code conversion for D-series systems.

Definitions

The following definitions describe essential components of the D-series system.

A *process control block (PCB)* contains information about the state, resources, and environment of a process in a CPU. A C-series system has a limit of 256 PCBs per CPU.

A *process identification number (PIN)* identifies a process in a particular CPU. To keep track of processes, the operating system assigns a specific PIN to each process running in a CPU.

A *high PIN* has a value of 256 or higher. High PINs are introduced with the D-series system.

A *low PIN* has a value that ranges from 0 through 254. The D-series system never assigns PIN 255 to a running process.

A *named process* had a name assigned to it when it was created. Other processes communicate with it by using its name.

An *unnamed process* had no name assigned to it when it was created. Other processes communicate with it by means of an identifier that contains its CPU number and PIN.

A *long file name* is a dollar sign ($) followed by seven alphanumeric characters for a disk file or a device, and a $ followed by six alphanumeric characters for a process.

Conversion

Conversion includes any change that allows a program to take additional advantage of the D-series enhancements. Thus, a program is *converted* when users have applied any of the following changes to it. It is *unconverted* if it uses the existing C-series interfaces and has not been changed in any of the following ways.

First, users can set options in the Tandem Binder linkage-editor utility to take advantage of the new object-file attributes provided by the D-series system. One attribute permits a Tandem Application Language (TAL) program to be executed at a high PIN. A second allows an unconverted process to be accessed by a high-PIN requester. A third directs the operating system to run a process as a named process, automatically supplying a name if the user does not specify one.

Changing an object-file attribute does not guarantee that a program will run correctly. If the program calls a procedure that has changed in the D-series system, users may have to recompile it or make code changes.

Second, if an application is written in a high-level language such as COBOL, users can recompile it with the D-series run-time libraries, part of the Common Run-Time Environment (CRE). (Some facilities that previously existed in the separate run-time libraries of the high-level languages are now provided by the general services of the CRE.) The recompiled application receives the benefits of using the new D-series procedures, and users do not have to change the application source code.

Third, to allow the application to continue functioning correctly in a D-series system, users may have to make application code changes, such as removing the dependencies on unpublished procedures or undocumented side effects of published procedures. For more information about undocumented features, see the Hilights document available on the D10 release SUT or through InfoWay.

Finally, users can change application code to use the new D-series procedure calls. These procedures allow the application to take direct advantage of the higher process control block (PCB) limits on D-series systems.

Table 1.
Sample user profile.

System configuration
4-CPU Cyclone system
36 mirrored disks
2 tape drives
20 I/O controllers, capable of handling a maximum of 1200 devices (terminals and printers)
Applications and users
80 local users connected through TACL interfaces
4 applications: order entry, inventory control, production control, and sales accounting
450 users connected only by the applications, not TACL interfaces

A Phased Approach to Migration

Tandem recommends that users migrate from the C30 release to a D-series release in several steps, or phases. Users do not have to take advantage of all the D-series enhancements, and most users can complete their migration without going through every phase. A phased approach allows users to migrate to a target phase that appropriately supports their applications and increases the limit on concurrently running processes to an extent suitable to their needs.

The phased approach reduces the operational risk of migration and allows users to gain experience gradually in a controlled environment. It is a conservative approach; most users will go through each distinct phase only for the first system or two that they migrate to the D-series environment. Once they understand the procedure thoroughly, they can migrate systems by going directly to the target phase.

Altogether, there are five phases of migration, defined as follows:

- Phase 0 establishes a stable C30 environment.
- Phase 1 installs the D10 release without making high PINs available during system generation.
- Phase 2 reconfigures the D10 release with high PINs available. Using the SYSGEN utility, users set selected Tandem I/O processes (IOPs) to run at high PINs.
- Phase 3 sets selected Tandem processes (not configured with the SYSGEN utility), as well as recompiled non-TAL applications, to run at high PINs.
- Phase 4 installs user applications in which the source code was partially or fully converted to use the D-series programmatic interface and to run at high PINs.

Table 1 shows a sample user profile, including the system configuration, applications, and users. The article uses this example to demonstrate the extent to which each phase of migration relieves the PCB limits of C-series systems.

Table 2 shows the distribution of PCBs in each CPU of the sample user system, the total number of processes running in the system, and the number of free (available) low PINs. At this stage the system is running a C-series release of the Guardian 90 operating system.

In the example, there are not enough PCBs left in each CPU to enable the system to be completely fault tolerant if one CPU fails. Assume, for example, that Process $A runs in CPU0 and its backup process, Process $A1, runs in CPU1. If CPU0 fails and Process $A1 takes over as the primary process, there are not enough PCBs left in the other CPUs for Process $A1 to create another backup. Because only one copy of Process $A1 is running, the system is no longer fault tolerant.

Another example assumes that, in a Tandem Pathway transaction processing environment, only one copy of the server is running. If a processor fails, the PATHMON process attempts to start copies of the server in a designated CPU. If not enough PCBs are available in that CPU, the user immediately loses the server's functions. In both these examples, the systems are prime candidates for migrating to a D-series system.

Table 2.
PCB distribution in each CPU of the sample user system.

Process types and totals	CPU0	CPU1	CPU2	CPU3	Total
Operating system processes (IOPs)	78	78	78	78	312
System processes and Tandem utilities	48	58	68	48	222
User applications	105	105	105	105	420
Total PINs	231	241	251	231	954
Free low PINs	25	15	5	25	70

Phase 0: Establishing a Stable C30 Environment

During Phase 0, users should develop their migration plan. They should anticipate their D10 requirements for all the systems in the network by choosing a target phase and configuration that meet the requirements of each system. At this phase, users should identify the order in which their systems should migrate.

It is recommended that users first migrate a test or development system to the D10 release. Next, they should migrate operator console nodes or network management nodes. When deciding on the order in which to migrate production nodes, users should consider the function of each node, the effort involved in its migration, and its associated limits issues, if there are any. For example, after migrating a production node that had limits problems, a user might next migrate a C30 node because it communicates extensively with the D-series node, even though it does not have intrinsic limits problems.

Users with partitioned files created by Tandem's Enscribe database record manager or partitioned tables created by the NonStop™ SQL relational database management system can migrate to the D-series system one node at a time. They do not have to migrate all nodes simultaneously. Nodes within a FOX™ (Fiber Optic Extension) ring can also migrate one system at a time. Users can follow the *D-Series System Migration Planning Guide* (1993) to aid their planning.

Benefits of Phase 0

The conservative approach to upgrading a system minimizes the number of changes to be made at the same time. Phase 0 allows users to upgrade the functions that are optional in the C30 release but mandatory in the D10 release. For example, for communications processes that support SCF on C-series systems, the user should convert CMI commands to SCF commands during this phase.

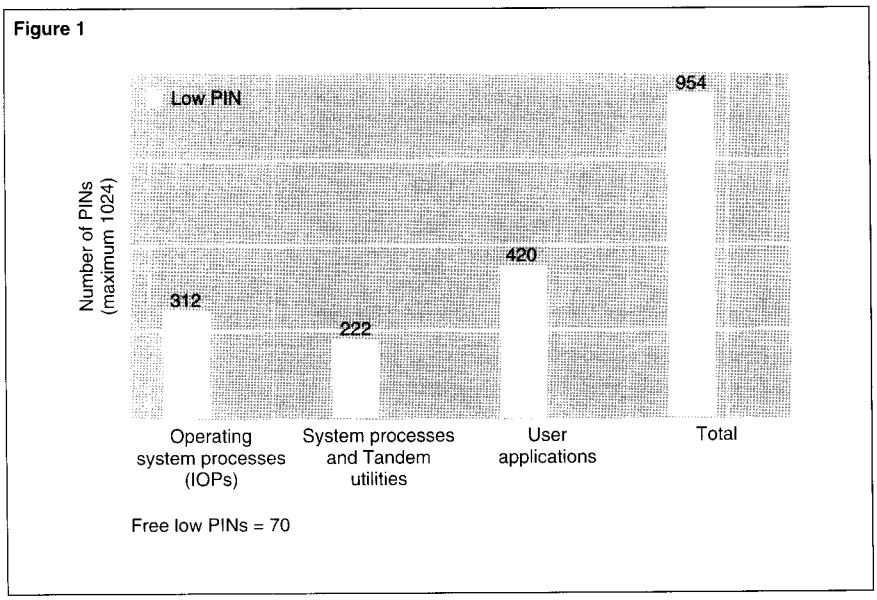


Figure 1.
A sample 4-processor Cyclone system in Phase 0. All processes run at low PINs.

Minimizing change reduces the risk and impact on operations and end users and spreads training over a longer period of time. Carrying out Phase 0 establishes a stable fallback position, and, most important, leads users to develop a migration plan that will simplify the migration of their system or network.

Phase 0 Tasks

Phase 0 tasks include eliminating discontinued hardware and software, as defined in the current Software Product Maintenance Levels document (available on the D10 release SUT or through InfoWay).

Users should also identify the memory and disk requirements for migrating to the D10 release. The D10 release uses at least 10 to 15 percent more memory than previous releases; the minimum memory requirement is 16 megabytes per CPU.

It is recommended that users take baseline peak-load measurements to help estimate the impact of migrating to the D10 release. The new PROFILE program can help users estimate the CPU consumption and memory required when moving to the D10 release. Before using the PROFILE program, one should read the new D10 Perform document, available on the D10 release SUT or through InfoWay.

Users can make most required D10 operational changes while still running the C30 release. They should determine which tasks these are and implement them. Moreover, they should ensure that the latter phases of the migration plan include those operational changes that cannot be made until the D10 release is installed. For example, users cannot migrate certain communications products to SCF before moving to the D10 release.

Users should understand to what extent Tandem products have been converted to take advantage of the D10 enhancements. (For details, read the Product Enhancement Summary on the D10 release SUT.) In addition, users should understand which D-series enhancements are planned for any Tandem Alliance and third-party software that they use.

After reviewing application software, users should estimate the need, effort, and cost to recompile high-level language applications and convert nonprivileged application code. For development and test systems, users should develop a plan, if needed, to maintain and test C30-based applications in a D-series environment.

Finally, users should plan to convert privileged code, if there is any. Unconverted privileged programs will not run successfully in the D-series environment and may cause processor halts.

During Phase 0, the user profile outlined in Tables 1 and 2 remains unchanged. Figure 1 shows the user profile graphically to demonstrate low- and high-PIN usage.

Migration Planning Tools

To facilitate the D-series migration, Tandem has developed three migration tools. The FINDER program enables the user to scan source files or command files for predefined search patterns in portions of code that will require changing before the program or macro can run on the D-series system.

FINDER does not change the source file; it flags the changes in an output file. Users can employ FINDER to help evaluate the scope of work required for migration.

The CMITOOL program takes CMI command files and converts the CMI commands to equivalent SCF commands. The PROFILE program, discussed earlier, estimates the D10 CPU consumption and memory increase using C30 measurements.

All three tools can run on the C30 release. FINDER and CMITOOL are available on both C30 and D10 release SUTs, and PROFILE is available on D10 release SUTs only.

Phase 1: Installing the D10 Release With Low PINs Only

During Phase 1, users install the D10 release on their system, but they do not configure any processes to run at high PINs. They accomplish this by ensuring that the SYSGEN CONFTEXT file explicitly configures 256 or fewer PCBs per processor. Phase 1 is an interim phase; users running a system in need of PCB limits relief are not likely to stay at this phase. (Users might migrate all nodes in a network to D-series systems to ease operations, even when most network nodes do not need PCB limits relief. Systems that do not need limits relief could remain at Phase 1.)

Benefits of Phase 1

Phase 1 allows users to verify that their unconverted applications run correctly on a D-series system. Moreover, it leaves all processes in the system accessible to any other system in their network.

Phase 1 allows users to make the operational transition to the D-series system. Operators gain experience with the D-series release and users have access to most new limits, including those for subdevices in X.25 networks, current opens for Enscribe files, and message-system control blocks. Users who are currently experiencing limits problems with Link Control Blocks (LCBs) obtain immediate relief at this phase.

Phase 1 establishes a stable D-series environment. Software developers can recompile and test converted application code, which may have been developed prior to the D-series release being installed. In addition, Phase 1 makes new D-series products and features available.

Phase 1 Tasks

During Phase 1, users should become familiar with the operational changes in the D10 release, particularly changes in error messages and displays. Users should migrate any discontinued functions that could not be migrated to replacement products while they were still running the C30 release.

In addition, users should determine whether all unconverted, nonprivileged applications will run correctly on the D-series system when they run at low PINs. Users also should convert any privileged code that is to run on the D-series system, if they did not do so during Phase 0. Phase 1 gives users the opportunity to recompile high-level language applications, convert TAL applications, and develop mixed-language programs using the Common Run-Time Environment (CRE).

Figure 2

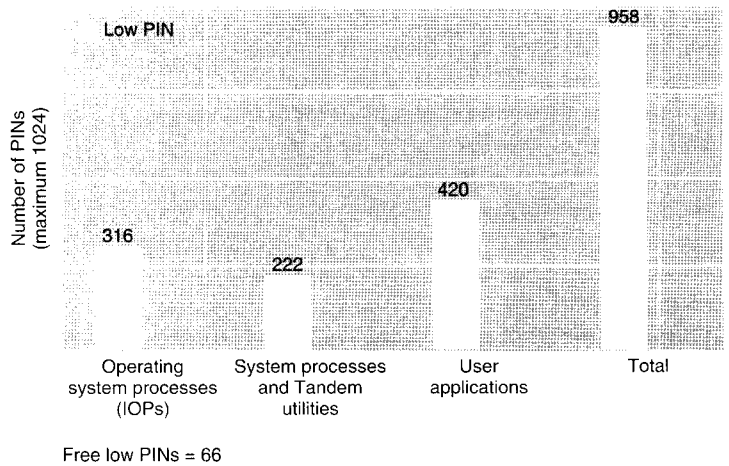


Figure 2.
A sample 4-processor Cyclone system in Phase 1. All processes run at low PINs.

If users must continue to maintain C-series applications in addition to developing D-series applications, they should plan to do their development work on a D-series system. Many D-series tools use the new D-series procedures and, as a result, cannot run on C-series systems. However, the C-series compilers and associated tools can run on a D-series release. Therefore, users should keep a separate set of C-series tools on the D-series node for C-series application development and maintenance.

At this interim phase of migration, users have access to most of the new D-series limits, but they do not gain any relief from PCB limits. The number of free low PINs does not increase. Figure 2 shows that four additional low PINs are used by the system in the sample user profile because D-series systems do not assign PIN 255 to a running process.

Phase 2: Running Tandem IOPs at High PINs

During Phase 2, users reconfigure the D10 release making high PINs available. Using the SYSGEN utility, users configure selected Tandem I/O processes (IOPs) to run at high PINs. At this phase, the system begins to obtain PCB limits relief.

Regardless of the network configuration, users can, with no effort, achieve PCB limits relief by means of the disk processes. Usually, the Tandem Disk Process (DP2) is configured with multiple disk processes for each disk volume. One of the disk processes in the volume group is the addressee for incoming messages; it is called the *head PIN*. The other processes in the group are called *tail PINs*.

When users employ the HIGHPIN modifier to run a disk IOP at a low PIN, only the head PIN runs at a low PIN. The tail PINs run at high PINs if high PINs are available. The disk IOP can be accessed from a C-series system because all communication with a disk process is addressed to its head PIN.

Users can also employ high PINs for most Tandem IOPs that do not need to be directly accessible from a C-series node. Users cannot run Tandem's Expand™ network IOPs at high PINs; Expand IOPs are flagged accordingly.

When users configure more than 256 PINs in a CPU that contains IOPs, the SYSGEN utility attempts to configure the IOPs to run at high PINs. To make a specific device or volume accessible from a remote C-series node, users must mark it explicitly by using the HIGHPIN OFF parameter in their SYSGEN configuration files. In a stand-alone or strictly D-series network, both converted and unconverted applications can access high-PIN IOPs.

To prevent processes initiated by TACL that default to high PINs from running at high PINs during this phase, the user should configure TACL to start processes at low PINs. This step ensures that only IOPs run at high PINs during this phase. At the same time, it enables the user to configure sufficient PINs at this phase to allow the system to move from Phase 2 to Phase 3 or 4 without requiring a new SYSGEN procedure to increase the number of available high PINs.

Benefits of Phase 2

The major benefit of Phase 2 is relief from the PCB limits of C-series systems. Depending on the users' requirements and application, this phase may provide adequate relief. It may be the target phase for the majority of users. By limiting the high-PIN processes to Tandem processes started by the SYSGEN utility, users reduce the chance of unexpected problems that could occur if other processes default to run at high PINs. They also reduce the chance that any compatibility issues might arise between unconverted applications and converted applications.

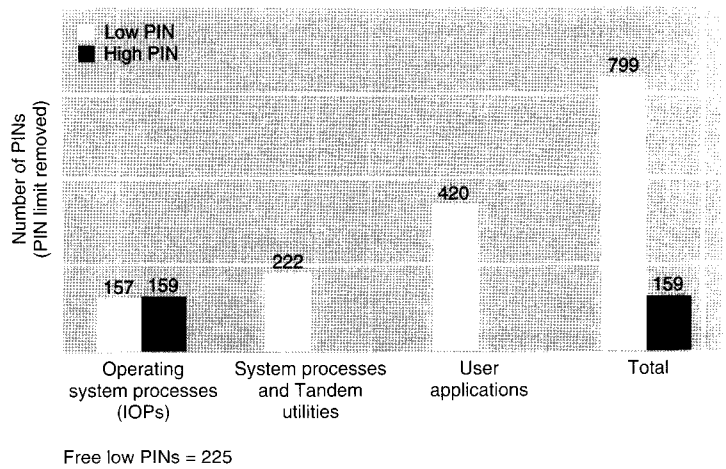
The User Profile at Phase 2

Figure 3 shows the user profile at Phase 2. During this phase, the sample user moved only disk tail PINs and line printer IOPs into high PINs, freeing up low PINs. With little effort, the user has made it possible to maintain the workload if a CPU fails. There are now enough available PCBs to generate new backup processes if a CPU fails.

Phase 3: Running Tandem Utilities and Recompiled Non-TAL User Applications at High PINs

During this phase, users should determine which additional Tandem processes can run at high PINs and which third-party software has been converted to run at high PINs. Before moving a Tandem or third-party process into a high PIN, users may have to convert part of any user program that the process opens or runs. In addition, users should identify the non-TAL applications that can be recompiled and run at high PINs without requiring any code conversion.

Figure 3



Benefits of Phase 3

Phase 3 increases the relief from PCB limits. By moving more processes and applications into high PINs, Phase 3 makes more low PINs available for unconverted processes.

Users may find that this phase provides adequate limits relief for their system and allows them to add applications to fully utilize the system. In a few cases, users have the potential to improve application performance. For example, users could improve the performance of a Pathway application by adding more servers or making servers static that previously were configured as dynamic because of PCB limitations.

Figure 3.
A sample 4-processor Cyclone system in Phase 2. Some IOPs run at high PINs.

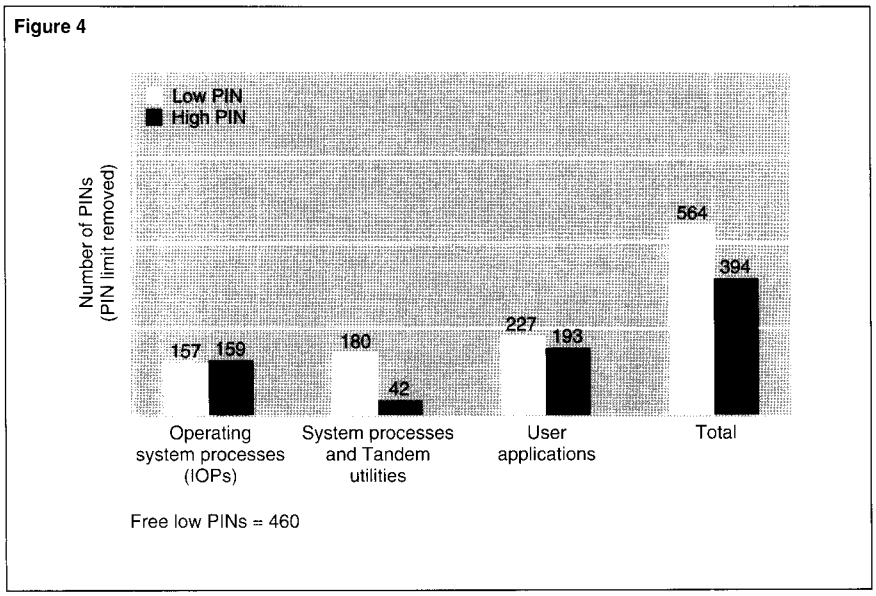


Figure 4.

A sample 4-processor Cyclone system in Phase 3. Some system processes, utilities, and application processes (Pathway servers) run at high PINs.

Table 3.
Requester-server communication: process communication rules for requesters and servers running at high and low PINs.

	Server process			
Requester process	D-series low PIN unconverted	D-series low PIN converted	D-series high PIN converted named	D-series high PIN converted unnamed
D-series low PIN unconverted	Yes	Yes	Yes	No
D-series low PIN converted	Yes	Yes	Yes	Yes
D-series high PIN converted named	No	Yes	Yes	Yes
D-series high PIN converted unnamed	No	Yes	Yes	Yes

Process Communication Rules

Users need to train their operations personnel to manage and troubleshoot issues arising from high-PIN process communication. They should fully understand the process communication rules for requester and server processes running at high and low PINs, as shown in Table 3.

An unconverted, D-series low-PIN requester can communicate with a converted, named high-PIN server because the high PIN process is being accessed by name, not by CPU number and PIN. An unconverted program cannot access a high-PIN server by CPU number and PIN because its data structures cannot hold the larger PIN. Table 3 shows that an unconverted low-PIN requester cannot access an unnamed, converted high-PIN server.

Requesters or servers on a C-series node can only communicate with low-PIN D-series requesters or servers (converted or unconverted). To determine whether the processes in an application can communicate without problems, refer to the *Guardian 90 Operating System Application Conversion Guide* (1993).

Figure 4 shows the sample user profile at Phase 3. During this phase, many system processes and utilities have moved into high PINs. The user also recompiled half of the Pathway servers to allow them to run at high PINs. The system now has more free low PINs, which, in most cases, is adequate for running the applications that the user does not wish to convert. Moreover, the user did not change any source code to achieve this relief from PCB limits.

Phase 4: Running Converted Application Code at High PINs

During Phase 4, users can execute converted application code at high PINs. This phase is for the few users who need to obtain even more relief from the PCB limits or who want to convert completely to the D-series interfaces. Phase 4 allows users to make more low PINs available for any remaining unconverted processes and also to take full advantage of high PINs.

Phase 4 requires a greater understanding of the new D-series Guardian 90 procedure calls and code conversion requirements. Users need to understand their objectives for converting an application. They must evaluate how many additional processes they need in each CPU, how much benefit they would get from converting a specific program to run at a high PIN, and how many other programs need to be converted in order to be able to run that program at a high PIN. For more information about conversion, see "Application Code Conversion for D-Series Systems," the companion article in this issue of the *Tandem Systems Review* (Liu, 1993).

Figure 5 shows the sample user profile at Phase 4. During Phase 4, the user converted source code in many additional application processes to allow them to run at high PINs. This leaves the system with even more free low PINs.

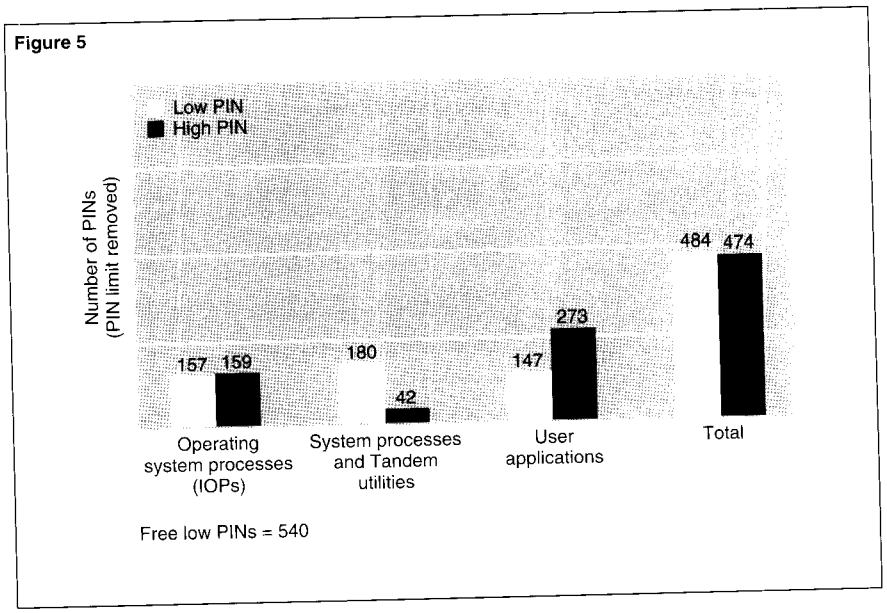


Figure 5.
A sample 4-processor Cyclone system in Phase 4. Additional application processes run at high PINs.

Networks

When migrating to the D-series system, users must consider the implications of process communication between network nodes. Any process that needs to be accessed by a C-series system must run at a low PIN on the D-series system.

In general, TACL processes should run at low PINs to allow access to C-series systems and prevent problems arising from unconverted programs that cannot support high-PIN creators. Tandem also recommends that the $SYSTEM disk process remain at a low PIN as long as there are C-series systems in the network.

Figure 6 shows the network view from a D-series process ($PR01) running at a high PIN. The $PR01 process can communicate with both high- and low-PIN processes running on another D-series system, but it cannot be accessed by a C-series system.

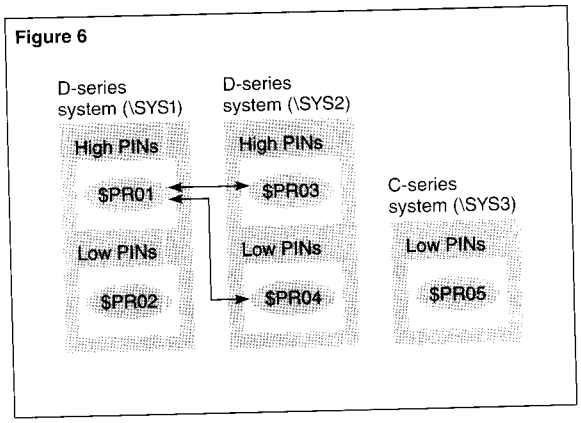


Figure 6.
Network view from a D-series high-PIN process ($PR01).

Figure 7.
Network view from a D-series low-PIN process ($PR04).

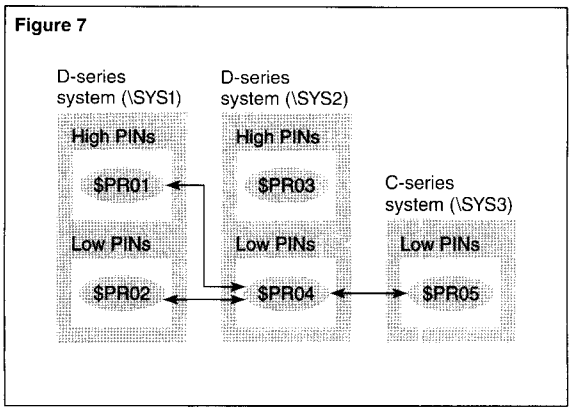


Figure 8.
Network view from a C-series process ($PR05).

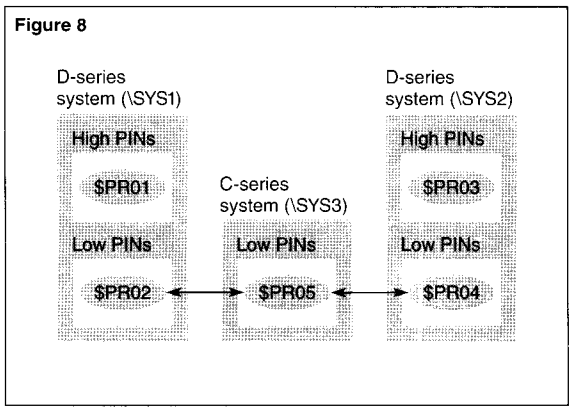


Figure 7 shows the network view from a D-series process ($PR04) running at a low PIN. The $PR04 process can communicate with all processes in the sample network, including those running at high and low PINs and those running on the C-series system.

Figure 8 shows the network view from a C-series process ($PR05). The $PR05 process can communicate with D-series processes running at low PINs, but it cannot access those running at high PINs.

To understand how processes and devices communicate in the network, users should create a communications map for each distributed application they plan to convert. The map can identify considerations such as process communication in mixed networks and between high-PIN and low-PIN processes. Users can find an example of a communications map in the *D-Series System Migration Planning Guide* (1993).

As users migrate network nodes to D-series systems, they also should eliminate the dependence on long file names for processes, disk files, and devices as a means of providing network security. The D-series procedural interface allows access to long names for remote files, as long as the process, disk file, or device resides on a D-series node.

Once users have migrated all network nodes to D-series systems, the process communication restrictions discussed above no longer apply to converted applications. However, these restrictions still apply to unconverted applications; they cannot communicate with unnamed high-PIN processes or handle long names of entities on remote nodes.

Performance

The average transaction pathlength (CPU cycle consumption) on the D10 release will increase approximately 10 to 15 percent. Communications-intensive applications may have somewhat longer pathlengths.

An increase in pathlength may or may not affect the throughput of the system. The effect on existing systems will vary depending on the nature of the application and the system configuration. A detailed C-series-to-D-series performance comparison and capacity planning document is available on the D10 release SUT.

Migration Planning Documentation, Education, and Services

To assist users with their migration to the D-series system, Tandem offers several new manuals and documents that discuss in detail the migration planning process and application conversion requirements.

- *Introduction to D-Series Systems.*
- *D-Series System Migration Planning Guide.*
- *Guardian 90 Operating System Application Conversion Guide.*
- *D-Series Product Enhancement Summary* (available on the D10 release SUT).

In addition, the Tandem Education Group offers a class called "Migrating to the D10 Release of the Guardian Operating System." With this information, most users can successfully plan and implement their migration to the D-series system.

For users who prefer to take advantage of individual guidance at their own sites, Tandem offers a fee-based professional service, the D-Series Migration Planning Service. This service helps users develop a plan for a successful transition to the D-series system. It provides software migration specialists who work with the users' staff to develop a phased, high-quality plan for a smooth software migration.

Conclusion

Migrating to the D-series system will be a straightforward process for most users. The key to a smooth and successful migration is to develop a good migration plan during Phase 0 and then implement it. Migrating in phases allows users to take advantage of the increased process capacity and other enhancements provided by the D-series system, while minimizing the effort and risk at each phase.

References

Bartlett, W. 1993. A Designer's Overview of the D-Series Guardian 90 Operating System. *Tandem Systems Review.* Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

D-Series System Migration Planning Guide. 1993. Tandem Computers Incorporated. Part no. 51439.

Guardian 90 Operating System Application Conversion Guide. 1993. Tandem Computers Incorporated. Part no. 69808.

Introduction to D-Series Systems. 1993. Tandem Computers Incorporated. Part no. 63950.

Liu, K. 1993. Application Code Conversion for D-Series Systems. *Tandem Systems Review.* Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

Acknowledgments

I'd like to thank Wendy Bartlett, Ernie Chilberg, and the reviewers for their contributions to this article.

Sue Kuukka has worked in Customer Support since 1981. She is the Technical Support Program Manager responsible for the D10 release.

Application Code Conversion for D-Series Systems

The D-series releases of the Tandem™ Guardian™ 90 operating system significantly increase the architectural capacity and practical limits of Tandem systems. In particular, the D-series releases allow more processes to run concurrently on a given processor than do previous releases. Users can thus take fuller advantage of the hardware capacity of Tandem systems and have the potential to increase the number of user applications on their existing Tandem equipment.

Tandem developers modified the Guardian 90 operating system to provide the increased capacity of its D-series releases. At the same time, the D-series releases provide an interface compatible with the one users have now.

Generally, users can run existing C-series-based application programs on D-series systems without having to change their application code. In most cases, users can realize the benefits of the increased architectural capacity of D-series releases simply by migrating their applications to a D-series system.

However, users may wish to change some application programs to take advantage of certain D-series enhancements. For example, users may want to run applications at high process identification numbers (PINs) after they have configured the operating system and Tandem products to run at high PINs. In these cases, Tandem recommends that users convert some application programs to take advantage of the many high PINs available in D-series systems.

Typically, these application programs make direct system procedure calls, as in a Transaction Application Language (TAL) program. For most application programs written in high-level languages that do not make direct system procedure calls, only recompilation is required to take advantage of the D-series enhancements.

The effort to convert an application can be minor or significant, depending on the complexity of the application. Often, users can achieve the desired result by converting only parts of their application programs. Before converting, users should analyze the application and identify the programs that can be converted with the greatest ease. By selecting programs wisely, users can minimize the conversion effort.

This article describes how to convert code in nonprivileged application programs from C-series to D-series Guardian 90 operating systems. It discusses the strategy recommended by Tandem for converting applications, the criteria for determining when converting applications is desirable, and the methods for performing code conversion.

The article is intended for system programmers, application programmers, and system managers who have performed D-series migration planning and want to convert their applications to take full advantage of the D-series enhancements. Readers are expected to have read the accompanying articles in this issue of the *Tandem Systems Review*, "A Designer's Overview of the D-Series Guardian 90 Operating System" (Bartlett, 1993) and "Migration Planning for D-Series Systems" (Kuukka, 1993). Readers should be familiar with the functions and terminology of the Guardian 90 operating system, in particular with system procedures for developing requesters and servers.

Enhancements to D-Series Systems

D-series systems include major extensions, additions, and modifications that enhance the facilities of C-series systems. These enhancements affect the process management interface, the file system interface, system messages, and error handling. The following descriptions of these enhancements provide the background information needed to understand application conversion. For more information about the D-series enhancements, see the *Guardian 90 Operating System Application Conversion Guide* (1993).

Extensions to the Process Management Interface

The process management interface allows users to access process management facilities provided by the Guardian 90 system. To keep track of processes, Guardian 90 assigns a unique process identification number (PIN) to each process running in a Tandem processor. The upper limit of the PIN value determines the maximum number of processes that can run concurrently in a processor.

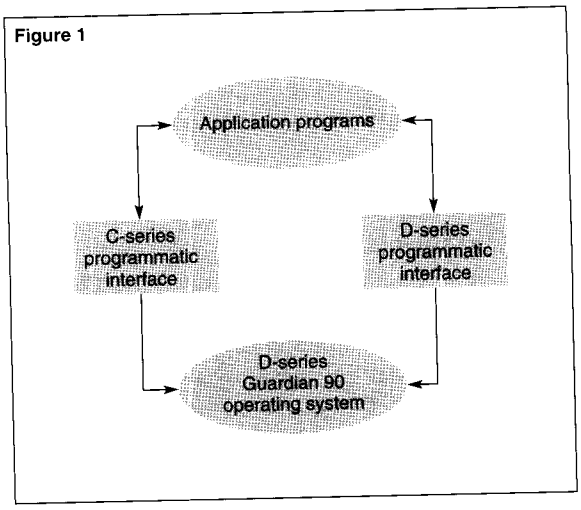


Figure 1.
D-series systems have two programmatic interfaces, one for C-series procedures and one for D-series procedures.

D-series systems allow PIN values far larger than C-series systems do. If an application is to run successfully at a PIN higher than is allowable on a C-series system, users may have to convert some portion of that application.

Two Programmatic Interfaces. To implement the D-series enhancements, Tandem developers changed the Guardian 90 programmatic interface. To remain compatible with applications developed on C-series systems, a D-series system has two interfaces for system procedures, as shown in Figure 1. One interface is the same as the interface in the C-series system; the other is unique to the D-series system. Because both interfaces are present, the D-series enhancements are an option rather than a requirement. Applications that are migrated to a D-series system without being converted can use the C-series interface.

The two programmatic interfaces function in somewhat different ways. C-series interfaces continue to behave as they do on a C-series system when they are used within the C-series limits. For example, a C-series-based application generally can run at a low PIN on a D-series system without any modification. C-series interfaces may exhibit new error behavior when they are used together with new limits on a D-series system. For example, a call to the MYPID procedure from a high-PIN process will trap. This condition occurs because the PIN value of the process does not fit into an 8-bit PIN field defined by the C-series system, and the MYPID procedure has no other means to specify an error condition.

D-series interfaces may default to different behavior than that of their C-series equivalents. For example, when a process opens the $RECEIVE file through the C-series OPEN procedure, it receives C-series system messages. When a process opens the $RECEIVE file through the D-series FILE_OPEN_ procedure, it defaults to receiving D-series system messages.

New System Procedures. The D-series operating system has many new user-accessible system procedures. Tandem provides declarations for many of the constants and structures used as parameters in the new system procedures. These declarations make system procedure calls easier and reduce application coding efforts. For example, Tandem provides the ZSYSTAL file (which contains these declarations) for TAL applications.

Process Handle. A *process handle* is a 10-word structure, introduced in D-series systems, that uniquely identifies a named or unnamed process in a Tandem network. For a named process pair, a process handle identifies a specific member of the pair; each member has its own process handle.

Process handles provide the same function in D-series systems that process identifiers do in C-series systems. The system uses process handles for process control, such as in the PROCESS_STOP_ procedure, or to gather information, such as in the PROCESS_GETINFO_ procedure. Programs use process handles to track and communicate with one another. Process handles were created because the larger size of D-series PIN values required a change in the process identifiers.

The process handle format is subject to change in future operating system releases. Users should not design applications to extract information (such as CPU number or PIN) directly from a process handle except by using a system procedure such as the PROCESSHANDLE_DECOMPOSE_ procedure.

New Object File Attributes. The D-series operating system provides three new object-file attributes. Users can set these attributes by using compiler directives in the source program, by using options when compiling the program, or by using the Tandem Binder linkage-editor utility. The attributes are as follows:

- The HIGHPIN attribute specifies to the Guardian 90 system that the program can be run at a high PIN.
- The HIGHREQUESTERS attribute allows an unconverted process to be accessed by a high-PIN requester.
- The RUNNAMED attribute specifies to the system to run the program as a named process. If the user does not specify a process name, the system generates one.

One uses these object-file attributes when converting an application. The attributes can provide additional information to the system and can eliminate the need to convert certain processes.

Extensions to the File System Interface

A D-series system also provides two file-system interfaces for access to disk files, devices, and processes. One provides the same interface as in a C-series system, and the other is unique to the D-series file system.

A D-series file name is a variable-length string with its length specified as a separate integer variable. The D-series file-system interface provides three major enhancements.

First, by extending process file names, the interface supports processes (specifically, unnamed processes) running at high PINs. Second, it accepts and returns file-name string parameters. Therefore, users do not have to convert file names from external to internal format to use D-series system procedures. Third, the interface automatically expands a partially qualified file name to a fully qualified name, eliminating some coding effort.

Extensions to System Messages

The D-series operating system provides new user-level system messages that a converted application can read from the $RECEIVE file. Some D-series messages supersede one or more C-series messages, and other D-series messages support new procedures or features. For example, the D-series -101 message, which indicates that a process was deleted, supersedes the following C-series messages: -5, which indicates that a process has stopped; -6, which indicates that a process had an abnormal end (abend); and -2, which indicates that a named process was deleted because a CPU failed.

A Strategy for Application Conversion

The primary objective of converting an application is to take advantage of the large number of high PINs offered by the D-series operating system, so that more application processes can run concurrently. Because the operating system uses the PIN extensively for process management and interprocess communication, users may have to make some effort to convert applications that have complicated process-management functions and use a large amount of interprocess communication.

Tandem recommends that users examine each candidate application and determine the extent to which individual processes communicate with one another. Users should focus first on converting programs that have less complicated interprocess communication, thereby minimizing the conversion effort.

Most user applications contain only nonprivileged code and do not have to be modified unless they use the D-series system enhancements. If a program contains privileged code, users must convert it. This article only discusses conversion of nonprivileged code.

Applications Consisting of a Single Program

Though not common on Tandem systems, some user applications may consist of a single program. Typically, the end user initiates a single-program application through the TACL™ (Tandem Advanced Command Language) interface. Multiple copies of the application might be running at the same time for multiple users; however, each copy of the application has only one process.

An application containing a single program requires the least amount of conversion effort, because it has little or no interprocess communication. In terms of effort, a single-program application is a good candidate for conversion.

When many users run a single-program application at the same time, the system generates multiple processes from the same application object code. By converting the application, users can free a potentially large number of low PINs that can support complex, large-scale, unconverted applications. By carefully identifying single-program applications, users can achieve maximum gains (in terms of freeing low PINs) for minimal conversion efforts.

Applications With Multiple Programs

A typical application consists of many processes that communicate with one another to achieve a specific objective. Converting this application may involve changing parts of each program. In keeping with Tandem's conversion strategy, users should try to maximize the gain (free as many low PINs as possible) and minimize the conversion effort.

Because the two versions of the programmatic interface in the D-series system follow well-specified rules, it is possible, and even desirable, in some cases, to convert a program *partially*. In a partial conversion, the application uses some of the D-series enhancements that support high PINs.

For each process that participates in an application, users should evaluate its interprocess relationships. They can do this by determining which processes can create the process being evaluated, which processes can open it, which processes it can open, whether it keeps track of its own identity, whether it keeps track of its creator's identity, and whether any other process keeps track of its identity.

Once users have analyzed and evaluated the relationships among the participating processes, they can decide which features should be provided for each process. With respect to high PINs, an application process can:

- Run at a high PIN.
- Accept and recognize a high-PIN creator.
- Open a high-PIN process.
- Support high-PIN openers or requesters.
- Create a high-PIN process.
- Monitor the status of a high-PIN process.

As the amount and complexity of interprocess communication and management increase, so does the effort to convert a process. Figure 2 shows the effort, from low to high, required to convert processes, based on the functions performed by those processes.

Conversion may involve no more than recompiling the application, if the application is written in a high-level language that does not make direct system procedure calls (using library routines). The remainder of this article describes code conversion for an application that does make direct system procedure calls.

In converting a multiple-program application, users should determine whether they must convert every program involved in the application. For a particular program that is to be converted, they should determine whether they must convert every superseded C-series procedure call to its corresponding D-series procedure.

Figure 3 shows a hypothetical application containing three basic process types. The first is a requester process ($REQ) that opens and sends requests to a server. The second is a server process ($SRV) that is opened by a requester process and receives and services the requests. A server process may also maintain an opener table to track its openers. The third process type is a monitor process pair ($MON) responsible for creating, monitoring, and managing both the requester and the server. One can use this application to conceptualize an application conversion strategy.

For the sake of simplicity, this article discusses separately each situation in which a process may need to be converted. If a process needs to use multiple D-series features, users should convert it one feature at a time, applying the relevant information in different sections of the article. This iterative approach maximizes the benefits of conversion (achieving PIN relief) while minimizing the conversion effort. Many user processes can take advantage of certain D-series features after users perform only partial code conversion or, in some cases, no code conversion at all.

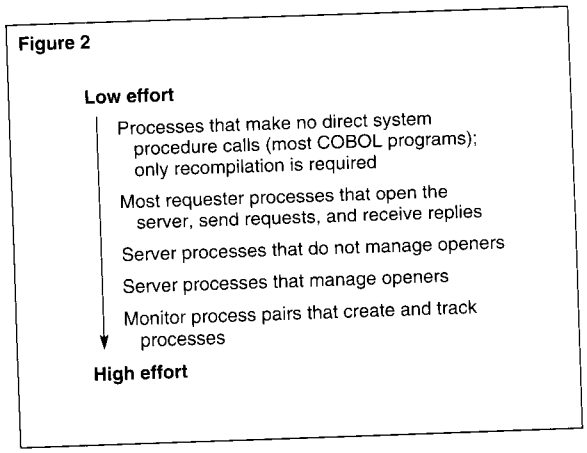


Figure 2.
The effort needed to convert processes based on their functions.

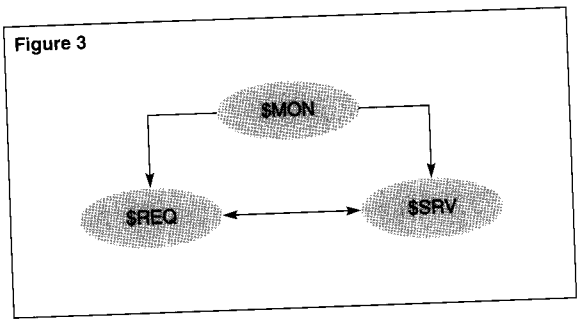


Figure 3.
A hypothetical application including a requester, server, and monitor process.

The examples in this article use the TAL language. However, the concept of application code conversion does not depend on a particular language. Topics discussed in the TAL context also apply to other language environments that interact with system procedures.

Figure 4.
A sample program segment calling the C-series MYPID procedure to obtain its own process identifier.

Figure 4

```
INT cpu^pin;          ! CPU and PIN values
  . . .
cpu^pin := MYPID;     ! Return the CPU and PIN
                        values
```

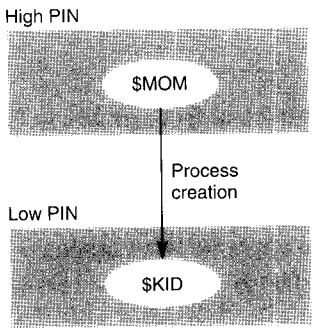
Figure 5

```
INT cpu^number,                                    ! CPU number
    pin^number,                                    ! PIN number
    phandle[0:ZSYS^VAL^PHANDLE^WLEN-1];            ! Process handle
error := PROCESSHANDLE_GETMINE_(phandle);          ! Get my phandle
                                                   ! Return caller's CPU and PIN values
error := PROCESSHANDLE_DECOMPOSE_(phandle,cpu^number,pin^number);
  . . .
```

Figure 5.
A sample converted program segment calling D-series procedures to obtain its own process identifier.

Figure 6.
A child process must be able to communicate with a high-PIN creator.

Figure 6



Converting a Process to Run at a High PIN

If a process is to run successfully at a high PIN, users may have to perform some conversion. To convert a program to run at a high PIN, users must set the HIGHPIN attribute for that object file. Users also have to convert the MYPID procedure for that program if the procedure is used.

The HIGHPIN attribute informs the system that the user intends to run the process at a high PIN. There are three methods of setting the HIGHPIN attribute. First, one can set the attribute as a compiler directive in the source program. For example:

```
?HIGHPIN
```

Second, one can set it as a compiler run-time option. For example:

```
> TAL /IN TALSRC, OUT $S.#TALLST/TALOBJ;
HIGHPIN
```

Finally, one can set it as an option using the Binder utility command. For example:

```
@CHANGE HIGHPIN ON IN TALOBJ
```

When binding multiple object files into a single target object file, the Binder utility sets the attribute only if all the constituent object files specify the HIGHPIN attribute. Users can employ the Binder utility to check the status of the HIGHPIN attribute for an object file by using the following command:

```
@SHOW SET HIGHPIN FROM TALOBJ
```

If a process does not retrieve its own process identifier, users do not need to convert further. However, if a process calls the MYPID procedure to obtain its CPU and PIN values, users need to convert the MYPID procedure for that process in order to run it at a high PIN. Figure 4 shows a sample program segment calling the MYPID procedure.

If a high-PIN process calls MYPID, a trap condition occurs because the MYPID procedure cannot return a 16-bit value, required for a high PIN, into an 8-bit PIN field defined by the C-series system. Users must convert the process to use the PROCESSHANDLE_GETMINE_ and PROCESSHANDLE_DECOMPOSE_ procedures introduced in D-series systems. Figure 5 shows a sample converted program segment that calls these procedures.

Users only need to make minimal code changes to allow a process to run at a high PIN. However, in a multiple-process application, users should thoroughly evaluate the impact that running a process at a high PIN may have on other participating processes.

Converting a Process to Accommodate a High-PIN Creator

Figure 6 shows that a process ($MOM) may need to create another process ($KID) in a user application. Users must analyze the child process to determine whether it can tolerate a high-PIN creator. Typically, a process obtains the creator's process identifier by using either the MOM or the LOOKUPPROCESSNAME procedure, defined in C-series Guardian 90 systems. These procedures, like all C-series procedures, provide an 8-bit PIN field, which cannot accommodate the 16-bit value of a high-PIN creator.

Therefore, if the creator is running at a high PIN, users must change the preceding C-series procedures to the PROCESS_GETINFO_ or the PROCESS_GETPAIRINFO_ procedure introduced in D-series systems. Figure 7 shows a sample converted program segment that uses the PROCESS_GETINFO_ procedure to obtain the process handle of the creator of a given process.

Often, after creating a child process, the creator opens the child process and sends startup messages to it. In this case, also, users may need to convert the child process to allow a high-PIN creator to open it. High-PIN openers are discussed later in this article.

Figure 7

```
. . .
INT mom^phandle[0:ZSYS^VAL^PHANDLE^WLEN-1];    ! Mom process handle
. . .
error := PROCESS_GETINFO_(,,,,mom^phandle);    ! Get creator's phandle
```

Figure 7.
A sample converted program segment calling the PROCESS_GETINFO_ procedure to obtain its creator's process handle.

Figure 8

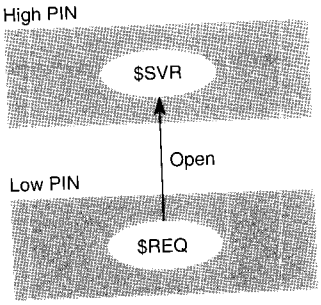


Figure 8.
An unconverted requester can open a named server running at a high PIN.

Figure 9

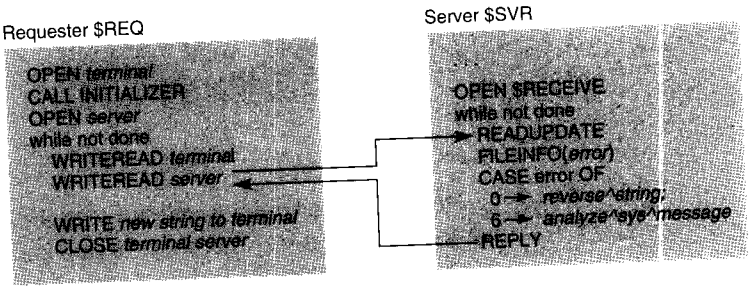


Figure 9.
Communication between a simple requester and server.

Converting a Process to Open a High-PIN Process

A requester is a process that opens and sends requests to a server process. (See Figure 8.) To communicate with a high-PIN server, the requester can run at a high PIN or a low PIN. The simple example shown in Figure 9 demonstrates the concept of requester and server and shows a typical communication sequence between a requester and a server.

As Figure 9 indicates, a requester can still use the C-series OPEN procedure to open a named high-PIN server process if the following conditions are met:

- The requester (or opening process) neither creates nor monitors the status of the high-PIN server.
- The server is a named process.
- If the server is on a remote D-series system, it must have a short process name (a dollar sign and fewer than five alphanumeric characters).

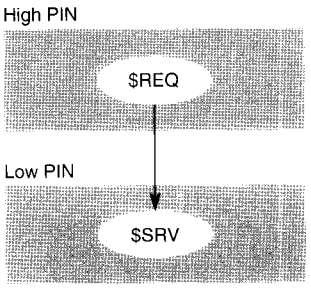
Figure 10.
A sample converted program segment calling the FILE_OPEN_ procedure to open a high-PIN server.

Figure 10

```
. . .
INT length, server^file^number;
STRING server^desc [0:3];
. . .
server^desc ':=' "$SRV";
length := 4;
. . .
error := FILE_OPEN_(server^desc:length,
                    server^file^number);
. . .
CALL WRITEREAD (server^file^number,...);
. . .
CALL FILE_CLOSE_(server^file^number);
. . .
```

Figure 11.
An unconverted server can accommodate a high-PIN requester by setting the HIGHREQUESTERS attribute.

Figure 11



However, the OPEN procedure cannot open an unnamed high-PIN process. In this case, users must convert the program code to use the FILE_OPEN_ procedure introduced in D-series systems. Figure 10 shows a sample converted program segment that calls the FILE_OPEN_ procedure.

Converting a Server to Accommodate a High-PIN Opener

Suppose a requester has been converted to run at a high PIN. Users may have to convert the server to accommodate the high-PIN opener if the server tracks, stores, and manipulates the opener's process identifier.

However, in some situations, a server can accommodate a high-PIN requester without requiring code conversion. The following discussion can help users determine whether a server needs to be converted to accommodate a high-PIN opener.

No Code Conversion Required

If the server does not use the opener's process ID, one can use the compiler or the Binder utility to set the HIGHREQUESTERS attribute for the server object file. Users do not have to change the server program code. By setting the HIGHREQUESTERS attribute, the server program informs the Guardian 90 system that, even though the server is not converted, it is safe to allow a high-PIN opener to access it, as shown in Figure 11.

In some cases, however, servers do need to maintain an opener table to keep track of the openers. Each time the server receives a message from a requester, it retrieves the requester's process ID and compares it to the process ID stored in the opener table to verify the legitimacy of the requester.

Figure 12 shows a sample code fragment that defines an opener table. This example uses the C-series-based, 4-word process ID format to keep track of the openers.

Figure 13 shows a sample code fragment that verifies the opener. In Figure 13, the *process^id* term contains the requester's process ID returned by the RECEIVEINFO procedure. If users have converted the requester to run at a high PIN, the opener-table definition shown in Figure 12 and the *verify^opener* procedure shown in Figure 13 can still work correctly without modification.

A C-series process ID can handle a high-PIN requester by using the synthetic PIN (PIN 255, reserved for system use only). One can let a process ID with a PIN 255 represent a process running at a high PIN because the information in the rest of the process ID is adequate to identify the actual process for the purpose of opener management. A synthetic PIN can represent a high-PIN requester when the HIGHREQUESTERS object-file attribute is set for an unconverted server program. Figure 14 shows the format of the process ID.

As shown in Figure 14, the process name of a high-PIN process can uniquely identify it in the system. For an unnamed high-PIN process, the combination of the verifier and CPU number also can uniquely identify the process.

In certain cases, by using the synthetic PIN, the server can accommodate high-PIN requesters without code conversion. However, users should be cautious with this approach; it can be dangerous if it is abused.

To simplify application conversion, users can set the RUNNAMED attribute for the requester. The D-series operating system guarantees that it will be a named process. The server can then depend on the requester's process name as a unique process identifier and, thus, does not require code conversion.

Converting the Server

Users must convert server program code if the server directly manipulates the CPU number and PIN of a high-PIN requester, or if the server sends the requester's CPU number and PIN as its unique process identifier in a message to another process. Users may have to take a few steps to convert the server, depending on its function.

Defining an Opener Table. The server must define an opener table that uses a process handle to identify the opener processes. If the server tracks its openers, it might define an opener table that uses a process ID to identify openers. Users must convert the opener table to use a process handle instead.

Figure 12

```
LITERAL max^opener = 3;
STRUCT .opener^table;
  BEGIN
    INT current^count;              ! The current count of openers
    STRUCT ocb [1: max^opener];     ! Array of opener IDs
      BEGIN
        INT crtpid[0:3];            ! Use C-series ID format
        END;
  END;
```

Figure 12.
Defining the opener table.

Figure 13

```
INT PROCEDURE verify^opener(process^id);
INT .process^id;
BEGIN
INT i := 1;

WHILE i <= max^openers DO
  IF opener^table.ocb[i].crtpid = process^id for 4
    THEN RETURN 1     ! We know the opener
  ELSE
    i := i + 1;       ! Keep looking

RETURN 0;

END;
```

Figure 13.
Verifying the opener.

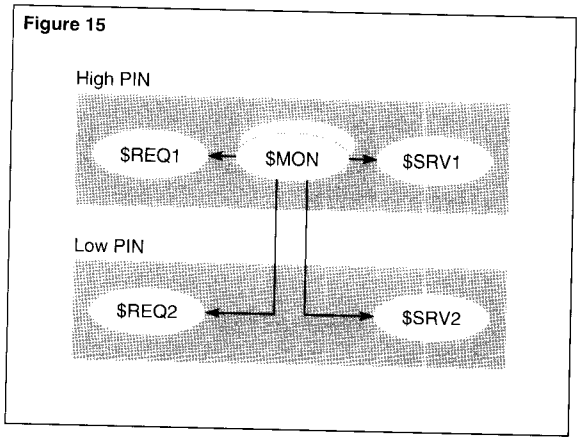
Figure 14

```
process-id [0:2]        = Process name or verifier
           [3].<0:3>    = Unused
              .<4:7>    = CPU number
              .<8:15>   = Actual PIN for a low-PIN
                          process
                          Synthetic PIN for a high-PIN
                          process
```

Figure 14.
The format of the process ID.

Opening the $RECEIVE File. The server should use the D-series FILE_OPEN_ procedure to open the $RECEIVE file rather than the C-series OPEN procedure. By using the FILE_OPEN_ procedure, the server informs the D-series operating system that it supports high PIN requesters and is ready to receive D-series system messages.

Figure 15.
A simple application configuration that uses a monitor process pair to create and manage processes.



Reading D-Series System Messages. The server must be able to read D-series system messages, which in some cases differ from C-series system messages; also, D-series messages are extensible. Users should create a READ[X] or READUPDATE[X] message buffer, as specified in the declarations in the ZSYSTAL file.

Using the FILE_GETRECEIVEINFO_ Procedure. The server should use the D-series FILE_GETRECEIVEINFO_ procedure to obtain opener information from system messages. If the server calls the C-series RECEIVEINFO or LASTRECEIVE procedure, users should convert it to call the FILE_GETRECEIVEINFO_ procedure. Also, if the server reads the C-series system messages -30 and -31 from the $RECEIVE file, users must convert the server to read the D-series system message -103, which indicates that a process has opened, and -104, which indicates that a process has closed.

Processing System Messages. If an opener has a CPU failure, or if its remote system fails or becomes partitioned from the server because of a network failure, the server does not receive a process-close message. Therefore, a server that maintains an opener table must read and process the status-change messages to check whether a certain system or network change may affect the processes the server is tracking.

To monitor status changes in local processors, the server can continue to read the C-series system message -2, which indicates that a CPU is down, and -3, which indicates that a CPU is up. Also, the server can continue to use the MONITORNET procedure to receive status-change messages from remote processors. The following D-series system messages supersede the C-series system message -8, which indicates a change in the status of a network node.

- The -100 message, which indicates that a remote CPU is down.
- The -101 message, which indicates that communication with the node has been lost.
- The -113 message, which indicates that a remote CPU is up.

Users should convert the server to read and process these messages, which provide more information about status changes. The D-series OPENER_LOST_ procedure facilitates a server's opener-table management. The server can use the OPENER_LOST_ procedure to search the opener table for any process affected by a status change indicated in the system messages.

Converting the Process Management Function

Sometimes, if an application contains multiple processes that perform extensive interprocess communication, it uses a monitor process to create and manage processes. Figure 15 shows a simple application configuration containing a monitor process pair.

Generally, converting a monitor process pair takes more effort than converting any other type of process. This is because D-series procedure calls have replaced the corresponding C-series calls for most procedures that create and manage processes. (The new procedure calls accommodate high-PIN processes.) Typically, a monitor process performs the following functions:

- Create high-PIN processes.
- Create low-PIN processes.
- Create backup processes.
- Manage processes.

In some cases, users may consider a full conversion of the monitor process, since most of the C-series system procedures that support the listed functions have been superseded. One can partly convert a monitor process, but partial conversion can introduce some complexity and confusion when C-series procedure calls and D-series procedure calls coexist in the same program. C-series procedures and D-series procedures use different formats for process IDs, file names, system messages, buffer sizes, and other values. During the code conversion, it may be difficult for the application programmer to switch frequently from one format to another, though many D-series system procedure calls exist to provide format conversions between these two interfaces.

Creating High-PIN Processes

To convert the process creation function, users should employ the D-series PROCESS_CREATE_ procedure. By default, the procedure attempts to create a high-PIN process if users have set the HIGHPIN attribute of the object file. Figure 16 shows a sample code fragment that uses the PROCESS_CREATE_ procedure.

Figure 16

```
. . .
error := PROCESS_CREATE_ (program^file:program^file^length,,,,
                          priority,
                          cpu_number,
                          process_handle,
                          error^detail);
```

Figure 16.
Creating a high-PIN process.

Creating Low-PIN Processes

In certain situations, a monitor process must create low-PIN processes. The C-series NEWPROCESS procedure always creates a process at a low PIN. If the monitor process creates a low-PIN process through the PROCESS_CREATE_ procedure, one can use the *create^options* parameter to direct the procedure to create a low-PIN process.

Creating Backup Processes

The monitor process uses the PROCESS_CREATE_ procedure to create its backup process. In comparison with the NEWPROCESS procedure, the PROCESS_CREATE_ procedure simplifies the steps for creating a backup process and requires less programming.

Managing Processes

To activate, suspend, stop, abend, and debug a process, the D-series system provides the PROCESS_ACTIVATE_, PROCESS_SUSPEND_, PROCESS_STOP_, and PROCESS_DEBUG_ procedures. These procedures provide functions similar to those available in the C-series system.

The D-series PROCESS_GETINFO[LIST]_ procedure, which replaces many C-series procedures, provides information about processes. To set certain attributes for a process, users can convert the program to call the PROCESS_SETINFO_ or PROCESS_SETSTRINGINFO_ procedure.

Users can employ the D-series CHILD_LOST_ procedure to facilitate system-message processing for a monitor process. The procedure accepts the child process handle and a system message, then checks the child process handle against the system message and indicates whether the status has changed. The message can be a C-series (-2, -5, -6, or -8) system message or a D-series (-2, -100, -101, or -113) system message indicating process deletion or a change in a process or network status.

Anticipating Side Effects

When converting a program, users should examine the entire application for the possible side effects of running in a D-series system. In particular, users should check data structures, the procedure interface, the user interface, interprocess communication messages, and event messages.

Users should inspect the data structures used by the application and the databases it creates or maintains. They should make sure all items and fields are defined to accept all possible values in a D-series system with enhanced limits.

Users should make sure that the parameters passed through the procedure interface are defined to allow large enough values in a system with enhanced limits. They should also ensure that the user interface will accept and display these larger values.

Application processes may send messages to one another. Users may have to redefine the message format to accommodate the larger values in a D-series system. Some applications may have to support multiple message formats so that messages from unconverted processes or processes running on C-series systems can still communicate with one another.

Conclusion

Most nonprivileged user applications can run on D-series systems without being changed as long as the applications do not use the new D-series features. However, users may want to convert application code to take advantage of some of the enhancements provided in D-series systems.

When planning to convert an application, users should thoroughly evaluate the relationships among the processes in the application. Understanding these relationships is the key to a successful application conversion.

In particular, users should focus on the uses of process IDs, interprocess communication, and process management. Once these issues have been resolved, subsequent conversion activities are relatively straightforward. Often, users can minimize code changes by determining which parts of the application require conversion and leaving the remaining parts unconverted.

References

Bartlett, W. 1993. A Designer's Overview of the D-Series Guardian 90 Operating System. *Tandem Systems Review*. Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

Guardian 90 Operating System Application Conversion Guide. 1993. Tandem Computers Incorporated. Part no. 69808.

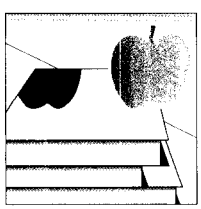
Kuukka, S. 1993. Migration Planning for D-Series Systems. *Tandem Systems Review*. Vol. 9, No. 2. Tandem Computers Incorporated. Part no. 89805.

Acknowledgments

Special thanks to Wendy Bartlett and Garry Easop for sharing their technical knowledge and expertise and thanks to all the reviewers of this article for providing technical and editorial suggestions.

Ken Liu joined Tandem in 1987 and is currently working in the Technical Support and Services Group. In the past year, Ken developed and taught the D-Series Application Conversion class.

Technical Information & Education



The Technical Information and Education department is an annotated list of new Tandem education courses and consulting and information services, as well as other technical information of interest to Tandem users.

Tandem Education

Tandem Education now offers a new training option, Interactive Distance Learning (IDL). IDL is a satellite-based instruction method that links students and instructors over an interactive network. By using a keypad, students can ask questions or make comments to the instructor who is teaching in another location.

The following paragraphs provide highlights of the latest education courses, including IDL presentations, offered by Tandem. To sign up for a class or to order an independent study program (ISP), users should call 1-800-621-9198. Full descriptions of all the available courses and ISPs appear in the *Tandem Education Course Catalog* and on InfoWay.

Migrating to the D10 Release of the Guardian Operating System (IDL)

This three-day IDL presentation provides an introduction to Guardian 90 system concepts unique to, or changed by, the release of D10-series software. The course enables users to determine what modifications, if any, may be required to migrate to the D10 series.

Tandem Systems Executive Overview

This high-level introduction to the Tandem fundamentals is an ideal starting point for managers unfamiliar with Tandem systems. The one-day course discusses the capabilities of Tandem systems in the areas of fault tolerance, data integrity, networking, distributed database, security, linear expansion, and performance.

Tandem Systems Executive Overview (IDL)

This is the interactive distance learning version of the lecture-based course described above.

Tandem Systems Technical Introduction

For technical personnel, this course (formerly called Concepts and Facilities) provides the best introduction to working with Tandem systems. The four-day course begins with a high-level introduction to the Tandem fundamentals. This part contains the same material as the one-day Tandem Systems Executive Overview course. The course then moves on to a more technical level, explaining how the Tandem fundamentals have been implemented in hardware and software products. Exercises are included that allow students to familiarize themselves with the basic system concepts and some of the standard utilities within the Guardian 90 operating system environment.

Tandem Systems Technical Introduction (IDL)

This course is the same as the lecture-based version described above, but taught in three days as an interactive distance learning presentation.

Tandem Professional Services

Tandem Professional Services is a program in which trained Tandem experts deliver technical consulting services at the user site. Tandem now offers a number of standardized services and will announce additional ones in this department of the *Tandem Systems Review* as they become available. Following are brief descriptions of new professional services offered by Tandem. For more information, users should contact their local Tandem representative.

Operations Review Service

This service provides an assessment of the user's computer operations environment in a number of areas. These include problem management, escalation procedures, change management, security management, capacity management, production management, and configuration management. The goal is improvement in any or all aspects of application availability, security, reliability, and efficiency.

In performing the service the Tandem specialist interviews representatives from the user's operations staff, observes current operations, and collects procedural information. Tandem then presents its conclusions and recommendations at a meeting with the user and in a final report.

RDF Implementation Planning Service

When Tandem's Remote Duplicate Database Facility (RDF) product is integrated into a disaster recovery strategy, the business resumption plan demands that the RDF software be configured and operated properly from the start. The RDF Implementation Planning Service fully prepares the user for implementation of the RDF product. The service provides the user with a complete analysis of the systems and operations to be used for the RDF implementation. The user receives sizing and configuration information, an RDF implementation plan, and training on the implementation and operational phases of the RDF product.

D-Series Migration Planning Service

This service assists users in developing a plan for migrating from the C-series releases of the Guardian 90 operating system to the D-series. The goal is to develop the most efficient migration strategy that has the least impact on the user's normal system operation. As part of the service, Tandem provides the user with D-series configuration and sizing information; a detailed migration project plan, listing specific tasks and a recommended order of system migration for multinode networks; an outline training plan for the user's staff; and the *Guardian 90 Operating System Application Conversion Guide.*

Technical Information Series Books

Tandem Technical Information Series (TIS) books are issued on a yearly subscription basis and individually. Each monograph covers in depth a technical topic related to Tandem systems. For more information about the TIS books, users should call Melissa Wibom at 1-800-538-3107.

Distributed Application Design Guidelines
March 1993

This book gives guidelines for designing distributed applications. Designers of distributed applications have literally hundreds of variables to cope with and evaluate in the design process. The results of a distributed application design can be a highly efficient application if one chooses the right distribution alternatives. This book describes the impact of key application, database, and network configuration options and emphasizes the importance of operations and management in a distributed application environment.

A Performance Guide to NonStop SQL Applications
April 1993

This book describes methods for improving the performance of NonStop database applications. It has six self-contained sections that discuss application design issues and operating issues. The topics covered by corresponding sections are: improving concurrency; setting file slack space for loading and reloading; maintaining table size; index prototyping; processing subsets of data from large sets and cursor repositioning; and batch updating techniques. The book is intended for Tandem users experienced in the areas of database administration and program design.

Trade Publications

Transaction Processing: Concepts and Techniques
Jim Gray and Andreas Reuter
Morgan Kaufmann Publishers
San Mateo, California, 1993
ISBN 1-55860-190-2

The purpose of this massive, 1,000-plus page book, as stated by the authors in their preface, is to give the reader an understanding of how large, distributed, heterogeneous computer systems can be made to work reliably. It presents a distributed system application development approach that can be used to solve real-life problems. Although the book is short on theory, it is long on practical, down-to-earth information and concentrates heavily on basic transaction issues. In other words, this is a pragmatic book aimed at the transaction processing specialist who is, or will be, responsible for making things work.

The book covers the subject of transaction processing comprehensively and in great detail. It consists of 16 chapters, grouped into 7 subject areas, which can be read more or less independently and in almost any order. Included are chapters on such topics as faults and fault tolerance, transaction-oriented computing, concurrency control, transaction management and recovery, and implementation of a transaction file system. The last chapter in the book presents overviews of a number of commercially available transaction processing systems. The book also includes an extensive glossary of transaction processing terms and a fairly good index.

In their discussions of transaction processing systems and applications, Gray and Reuter repeatedly stress the importance of implementing the ACID properties (atomicity, consistency, isolation, and durability) of transactions. The authors also discuss the application of transaction processing principles in client/server computing. The book does not cover SQL implementation or application design.

The book is written in a straightforward and readable style with numerous figures and code fragments to help illustrate the concepts and techniques discussed. Almost all of the chapters have exercises of varying levels of difficulty with answers to most of them provided at the end of each chapter. In addition, most chapters conclude with a section that provides a historic perspective on the subject covered. Although the book is intended primarily for advanced students and professional programmers, those who just want to get a good idea of what transaction processing is all about can profit from a selective reading of the material.

InfoWay Service

InfoWay™ is an online electronic service that provides a flow of information among Tandem™, its customers, and its partners. The Summer 1992 issue of the *Tandem Systems Review* contains an overview of the basic features of InfoWay. These include electronic mail; the software version analyzer (SVA); and access to interim product modifications (IPMs), training class schedules, Tandem and ITUG publications, marketing literature, and the Tandem Alliance Directory.

Users can sign up for access to InfoWay either through their account team or the Tandem NonStop Support Center (TNSC), or by filling out a form provided on their site update tapes (SUTs). To learn more about InfoWay, users should contact their local Tandem representative. The following describes access methods for connecting to the InfoWay service and the new features available in InfoWay Release 5.0.

Access to InfoWay

Of the four methods that Tandem provides to access the InfoWay service, two require the installation of Tandem's SysWay™ software. This software is a gateway that temporarily and transparently connects two Tandem systems that are not connected by Expand™ software. Tandem delivers SysWay on all SUTs of Release C20 and later.

With SysWay, InfoWay services and databases appear as if they are on the user's Tandem system. Information from the databases can be downloaded and printed locally. Users can access InfoWay electronic mail software and they have the ability to exchange mail and files with their account teams. SysWay runs as a nonprivileged process. Its connections are fully secure, temporary, and are dissolved upon user request.

Access From a Tandem NonStop System Using Asynchronous Dial-Up Lines

This access method requires SysWay software and either the Tandem TERMPROCESS or Asynchronous Terminal Process (ATP6100). Each asynchronous link must have its own port with an attached modem that supports the Hayes AT command set. Usable modems include those complying with the Microcom Network Protocol (MNP) at 300, 1200, 2400, 4800, and 9600 bps, as well as those complying with the CCITT standards V.32, V.22 bis, and V.22.

Access From a Tandem NonStop System Using the X.25 Access Method

For this access method, terminals must be connected to a Tandem host running under a Guardian™ operating system. Both the X.25 Access Method and SysWay software are required. In addition, users must have a leased line connecting the Tandem system to the InfoWay service point or an X.25 public data network (PDN).

Access From a Stand-Alone PC or Workstation Using Asynchronous Dial-Up

No Tandem host system is required for this method, and any terminal emulation software can be used to access InfoWay. However, by employing one of Tandem's terminal emulation packages, the PC6530 package or the Tandem Terminal Emulators (TTE) package, users can have the added benefit of being able to transfer files in either direction between their PCs and InfoWay. Modem requirements are the same as those for the Tandem NonStop™ system using asynchronous dial-up lines.

Access From a PC or Workstation Using X.25 PAD Dial-Up Capability

This method provides direct connection with a wide range of workstations and personal computers. Terminals must operate in conversational mode with 7 data bits, even parity, a baud rate up to 9600, full duplex, and no local echo.

Connecting Transfer Systems

Tandem's TransWay™ software, which is standard on all SUTs for Release C20 and later, is a gateway that connects one Transfer™ system to another. It allows electronic mail to flow between the systems. The connections can be on the same Expand network or (with the use of SysWay software) on different Expand networks.

Without the TransWay software, a user must access the InfoWay PS Mail™ software to exchange electronic mail with the user's account team. This is also true if the user is operating from a stand-alone personal computer. With TransWay software installed, a user can exchange electronic mail through the user's own Transfer mail system.

TransWay software periodically sends electronic mail messages in batches while simultaneously retrieving return mail. As a result, communications channels are accessed less frequently and costs are reduced. The user's system manager can set the transmission frequency.

InfoWay Release 5.0

Release 5.0 provides users a new software delivery service, the addition of block-mode capability, and new search features. These are described in the paragraphs that follow.

The SWEDS Software Delivery Service

To provide users an online method of ordering and procuring IPMs, InfoWay Release 5.0 offers the SWEDS software delivery service. Through the SWEDS service, users can ask for electronic delivery of IPMs or send requests to Tandem's Software Distribution and Release Control Group (SDRC) or the Regional Distribution Center (RDC) for delivery of IPMs on a physical medium such as magnetic tapes, cartridges, or CD-ROM disks.

Electronic Delivery. Electronic delivery requires a Tandem system at the user site and the additional software specified later. User options include immediate and delayed delivery.

Immediate delivery commences while the user waits online. Delayed delivery occurs at a user-specified time with no further user intervention. The latter requires TransWay software that is configured to dial out to Tandem at user-specified times. The advantage of delayed delivery is that it gives users the ability to request deliveries when transmission rates are less expensive.

For both delivery methods, SWEDS software informs the user of the size of the requested file and the time it will take to download it. Files are automatically compressed before the download occurs.

Delivery by Physical Media. For delivery of IPMs by physical media, the user places a request through the SWEDS order submenu. The SDRC or RDC processes the request and ships the requested IPMs to the user.

SWEDS Requirements. Electronic delivery through the SWEDS service requires installation of SysWay and TransWay software as well as SWEDS software. Users must also have signed up for access to InfoWay. Previous releases of TransWay required Transfer software, but with TransWay version C31, the Transfer software is not required if TransWay is used only for the SWEDS service.

Addition of Block Mode

InfoWay Release 5.0 provides block mode access as well as conversational mode. The new default is block mode, but conversational mode can be specified during logon to InfoWay.

In Release 5.0, users can configure the access method when they set up their user profiles from the Functions Menu of block-mode InfoWay. These profiles are active across sessions and the user can change them while in an InfoWay session. Users can also accomplish block-mode file transfers to a PC as well as to a Guardian file.

InfoWay services that are available in block-mode include Mail, SVA, Mail DIRectory, and Mail SIGs. The change-password feature is also offered in block mode.

Expanded Search Features

In InfoWay Release 5.0, a new date field called *Update Date* has been added as a search field for most databases. This field provides users the ability to search for documents on the basis of the date the documents were last updated in InfoWay. This is helpful to users who wish to view the most current changes to product documentation or updates in Tandem training class schedules.

For fields that support wild cards, users can enter leading wild cards for database searches. A leading wild card is a wild card, such as *, that is used as the first character in the search criteria. For example, a search for **25* retrieves documents that contain *X25, XA25*, and all other documents that contain *25*. As before, online help describes whether a field supports wild cards.

Users can also enter a Boolean operator, such as <, >, <=, or >=. For example, the search criterion *>920102* retrieves all documents that contain a date later than January 2, 1992.

Release 5.0 also offers an enhancement in connection with the maximum number of documents to be retrieved in a database search. This specification is still user-configurable, but is now retained between dial-in sessions of InfoWay.

TandemSystemsReview*Index*

The *Tandem Journal* became the *Tandem Systems Review* in February 1985. Four issues of the *Tandem Journal* were published:

Volume 1, No. 1 Fall 1983
Volume 2, No. 1 Winter 1984
Volume 2, No. 2 Spring 1984
Volume 2, No. 3 Summer 1984

As of this issue, 21 issues of the *Tandem Systems Review* have been published:

Volume 1, No. 1 Feb. 1985
Volume 1, No. 2 June 1985
Volume 2, No. 1 Feb. 1986
Volume 2, No. 2 June 1986
Volume 2, No. 3 Dec. 1986
Volume 3, No. 1 March 1987
Volume 3, No. 2 Aug. 1987
Volume 4, No. 1 Feb. 1988
Volume 4, No. 2 July 1988
Volume 4, No. 3 Oct. 1988
Volume 5, No. 1 April 1989
Volume 5, No. 2 Sept. 1989
Volume 6, No. 1 March 1990
Volume 6, No. 2 Oct. 1990
Volume 7, No. 1 April 1991
Volume 7, No. 2 Oct. 1991
Volume 8, No. 1 Spring 1992
Volume 8, No. 2 Summer 1992
Volume 8, No. 3 Fall 1992
Volume 9, No. 1 Winter 1993
Volume 9, No. 2 Spring 1993

The articles published in all 25 issues are arranged by subject below. (*Tandem Journal* is abbreviated as TJ and *Tandem Systems Review* as TSR.) A second index, arranged by product, is also provided.

Index by Subject

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
PATHFINDER–An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES *(cont.)*					
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
State-of-the-Art C Compiler	E. Kit	TSR	2,2	June 1986	83937
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CLIENT/SERVER					
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
DATA COMMUNICATIONS					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DATA MANAGEMENT					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Batch Processing in Online Enterprise Computing	T. Keefauver	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32968
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Online Information Processing	J. Viescas	TSR	9,1	Winter 1993	89804
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
Optimizing Batch Performance	T. Keefauver	TSR	5,2	Sept. 1989	28152
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
OPERATING SYSTEMS					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Approach	A. Borr	TSR	1,2	June 1985	83935
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83935
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
PERFORMANCE AND CAPACITY PLANNING					
A Performance Retrospective	P. Oleinick, P. Shah	TSR	2,3	Dec. 1986	83938
Buffering for Better Application Performance	R. Mattran	TSR	2,1	Feb. 1986	83936
Capacity Planning Concepts	R. Evans	TSR	2,3	Dec. 1986	83938
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Credit-authorization Benchmark for High Performance and Linear Growth	T. Chmiel, T. Houy	TSR	2,1	Feb. 1986	83936
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
Estimating Host Response Time in a Tandem System	H. Horwitz	TSR	4,3	Oct. 1988	15748
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938
Getting Optimum Performance from Tandem Tape Systems	A. Khatri	TSR	2,3	Dec. 1986	83938
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
Optimizing Sequential Processing on the Tandem System	R. Welsh	TJ	2,3	Summer 1984	83933
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PERFORMANCE AND CAPACITY PLANNING *(cont.)*					
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Performance Considerations for Application Processes	R. Glasstone	TSR	2,3	Dec. 1986	83938
Performance Measurements of an ATM Network Application	N. Cabell, D. Mackie	TSR	2,3	Dec. 1986	83938
Predicting Response Time in On-line Transaction Processing Systems	A. Khatri	TSR	2,2	June 1986	83937
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
The Performance Characteristics of Tandem NonStop Systems	J. Day	TJ	1,1	Fall 1983	83930
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11978
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
PERIPHERALS					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
Data-Encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Data-Window Phase-Margin Analysis	A. Painter, H. Pham, H. Thomas	TSR	2,2	June 1986	83937
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
Peripheral Device Interfaces	J. Blakkan	TSR	3,2	Aug. 1987	83940
Plated Media Technology Used in the XL8 Storage Facility	D.S. Ng	TSR	2,2	June 1986	83937
Streaming Tape Drives	J. Blakkan	TSR	3,2	Aug. 1987	83940
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	3,2	Aug. 1987	83940
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PROCESSORS					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250
The High-Performance NonStop TXP Processor Transaction Processing	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
SECURITY					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SYSTEM CONNECTIVITY					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
Terminal Connection Alternatives for Tandem Systems	J. Simonds	TSR	5,1	April 1989	18662
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
SYSTEM MANAGEMENT					
Configuring Tandem Disk Subsystems	S. Sitler	TSR	2,3	Dec. 1986	83938
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
SYSTEM MANAGEMENT *(cont.)*					
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
UTILITIES					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Index by Product

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
3207 TAPE CONTROLLER					
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
5120 TAPE SUBSYSTEM					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
5200 OPTICAL STORAGE					
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	4,1	Feb. 1988	11078
6100 COMMUNICATIONS SUBSYSTEM					
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
6530 TERMINAL					
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
6600 AND TCC6820 COMMUNICATIONS CONTROLLERS					
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
Ada					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
BASIC					
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
C					
State-of-the-art C Compiler	E. Kit	TSR	2,2	June 1986	83937
CIS					
Customer Information Service	J. Massucco	TSR	3,1	March 1987	83939
CLX					
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
COBOL85					
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
COMINT (CI)					
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CYCLONE					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
DAL SERVER					
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DP1 AND DP2					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
DSM					
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
DYNAMITE					
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
ENABLE					
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
ENCOMPASS					
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
ENCORE					
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
ENSCRIBE					
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
FASTSORT					
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
FOX					
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
FUP					
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
GUARDIAN 90					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
B00 Software Manuals	S. Olds	TSR	1,2	June 1985	83935
C00 Software Manuals	E. Levi	TSR	4,1	Feb. 1988	11078
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Multiprocessor Approach	A. Borr	TSR	1,2	June 1985	83935
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83934
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
INTEGRITY S2					
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
MEASURE					
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
MULTILAN					
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
NETBATCH-PLUS					
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NONSTOP SQL					
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32986
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
OSI					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
PATHFINDER					
PATHFINDER–An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930
PATHWAY					
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
POET					
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
PS MAIL					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
RDF					
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848
RESPOND					
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
RSC					
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
SAFEGUARD					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SNAX					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
SPOOLER					
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11078
TACL					
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
TAL					
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
TCM					
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
TLAM					
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
TMDS					
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
TMF					
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934
TNS/R					
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
TRANSFER					
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TXP					
The High-Performance NonStop TXP Processor	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
V8					
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935
VIEWSYS					
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
VLX					
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
XL8					
Data-encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Plated Media Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937

TandemSystemsReview*OrderForm*

Use this form to order new subscriptions, change subscription information, and order back issues.

☐ I am a Tandem customer. My Tandem sales representative is _____.

☐ I am not a Tandem customer and am enclosing a check or money order for the requests indicated on this form. (Subscriptions are $75 per year and each back issue is $20. Make checks payable to Tandem Computers Incorporated.)

Subscription Information

☐ New subscription

☐ Update to subscription information
Subscription number: _____

Your subscription number is in the upper right corner of the mailing label.

_____
COMPANY

_____
NAME

_____
JOB TITLE

_____
DIVISION

_____
ADDRESS

_____

_____
COUNTRY

_____
TELEPHONE NUMBER (include all codes for U.S. dialing)

Title or position:

☐ President/CEO
☐ Director/VP information services
☐ MIS/DP manager
☐ Software development manager
☐ Programmer/analyst
☐ System operator
☐ End user
☐ Other: _____

Your association with Tandem:

☐ Tandem customer
☐ Third-party vendor
☐ Consultant
☐ Other: _____

Back Issue Requests

Number of copies

Tandem Systems Review

____ Vol. 1, No. 1, Feb. 1985
____ Vol. 1, No. 2, June 1985
____ Vol. 2, No. 1, Feb. 1986
____ Vol. 2, No. 2, June 1986
____ Vol. 2, No. 3, Dec. 1986
____ Vol. 3, No. 1, March 1987
____ Vol. 3, No. 2, Aug. 1987
____ Vol. 4, No. 1, Feb. 1988
____ Vol. 4, No. 2, July 1988
____ Vol. 4, No. 3, Oct. 1988
____ Vol. 5, No. 1, April 1989
____ Vol. 5, No. 2, Sept. 1989
____ Vol. 6, No. 1, March 1990
____ Vol. 6, No. 2, Oct. 1990
____ Vol. 7, No. 1, April 1991
____ Vol. 7, No. 2, Oct. 1991
____ Vol. 8, No. 1, Spring 1992
____ Vol. 8, No. 2, Summer 1992
____ Vol. 8, No. 3, Fall 1992
____ Vol. 9, No. 1, Winter 1993
____ Vol. 9, No. 2, Spring 1993

Tandem Journal

____ Vol. 1, No. 1, Fall 1983
____ Vol. 2, No. 1, Winter 1984
____ Vol. 2, No. 2, Spring 1984
____ Vol. 2, No. 3, Summer 1984

For questions or ordering information, call 800-473-5868 in the U.S. and Canada or +1-408-285-8150 in other countries.

Send this form to:
Tandem Computers Incorporated
Tandem Systems Review, Loc 208-65
10400 Ridgeview Court
Cupertino, CA 95014-0723
FAX: +1-408-285-0840

Tandem employees must order their subscriptions and back issues through Courier.

Menu sequence: Marketing Information→ Literature Orders→ Technical Marketing Pubs (TSR)

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD

TandemSystemsReview*ReaderSurvey*

The purpose of this questionnaire is to help the *Tandem Systems Review* staff select topics for publication. Postage is prepaid when mailed in the United States. Readers outside the U.S. should send their replies to their nearest Tandem sales office.

1. How useful is each article in this issue?

Product Update
01 ☐ Indispensable 02 ☐ Very 03 ☐ Somewhat 04 ☐ Not at all

A Designer's Overview of the D-Series Guardian 90 Operating System
05 ☐ Indispensable 06 ☐ Very 07 ☐ Somewhat 08 ☐ Not at all

Migration Planning for D-Series Systems
09 ☐ Indispensable 10 ☐ Very 11 ☐ Somewhat 12 ☐ Not at all

Application Code Conversion for D-Series Systems
13 ☐ Indispensable 14 ☐ Very 15 ☐ Somewhat 16 ☐ Not at all

Technical Information and Education
17 ☐ Indispensable 18 ☐ Very 19 ☐ Somewhat 20 ☐ Not at all

2. I specifically would like to see more articles on (select one):

21 ☐ Overview discussions of new products and enhancements
22 ☐ Performance and tuning information
23 ☐ High-level overviews on Tandem's approach to solutions
24 ☐ Application design and customer profiles
25 ☐ Technical discussions of product internals
26 ☐ Other _____

3. Your title or position:

27 ☐ President, VP, Director 28 ☐ Systems analyst 29 ☐ System operator
30 ☐ MIS manager 31 ☐ Software developer 32 ☐ End user
33 ☐ Other _____

4. Your association with Tandem:

34 ☐ Tandem customer 35 ☐ Tandem employee 36 ☐ Third-party vendor 37 ☐ Consultant
38 ☐ Other _____

5. Comments

_____

_____

_____

_____

NAME _____

COMPANY NAME _____

ADDRESS _____

_____

4/93

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD

TANDEM

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599

MARC BRANDIFINO
LOC NUM 56-00
NEW YORK NY DOWNTOWN DISTRICT

Part No. 89805

4/93 Printed in USA