Architecture Differences 14

Solaris 2.x for x86

Solaris for x86 is a complete 32-bit distributed computing environment that runs on machines based on a 386DX-class of microprocessor or higher.

The Solaris for x86 environment is intended for power users who need to communicate with others through a distributed computing environment and who use performance-critical applications that ned a graphical user interface or employ client-server capabilities.

The unique advantage of Solaris for x86 is that it provides a proven environment integrated with a graphical user interface, desktop applications, heterogeneous networking, multitasking, multiprocessing, network security, and an industry standard operating system. No other single product in the x86 32-bit market can provide all of these features.

Solaris SunOS Standards and Specifications

The standards and specifications supported by SVR4 make this release unique. They include SVID3, SCD 2.0, X/Open, XPG-3 (POSIX 1003.1), IEEE 754 ANSI C, and DDI/DKI. The SunOS 5.x interfaces are the standard SVR4 interfaces.

SunSoft's goal for the Solaris environment is to provide a uniform distributed computing environment across the industry's leading volume platforms: SPARC RISC and the 80x86 CISC.

The SPARC and x86 releases of Solaris are similar. SunSoft's intent is that Solaris on the x86 architecture be identical to the SPARC port. In particular the API's of the two implementations are identical, other than some minor details of the DDI/DKI interface mandated by differences in the architectures. There are two possible exceptions to the policy of uniformity between implementations of Solaris software:

1. Solaris for x86 may vary where differences in the underlying x86 hardware environment requires an alternate treatment.

For example, device configuration varies somewhat between SPARC and x86. In the SPARC workstation environment, most hardware devices, such as SBus cards, are self-identifying. In the x86 marketplace, ISA peripherals are not self-identifying, and EISA and MCA configuration procedures provide only limited self-identifying features.

Few ISA devices have port and IRQ interrupt vector information that can be read or set from software. EISA systems can provide identification and option information for each EISA card on the bus, but not for ISA cards. A micro-channel system can provide identification and optional information for every card on the MCA bus. When EISA and ISA cards are mixed, there is the potential for misconfiguration and overlapping IRQ settings.

To handle device configuration all manufacturers of EISA systems provide an EISA utility to configure devices initially, or later when devices are added.

Consequently, Solaris for x86 contains an additional utility for device configuration (`drvconfig`).

Note – Developers may refer to the SunSoft white paper entited *Solaris 2.x for x86 Platforms* (part number 93033-001) for a list of compatible devices/peripherals.

---

2. Solaris for x86 may vary where the customer's expectations of typical target users vary from that of SPARC users.

   Solaris for x86 enables 80x86 systems to run multiple concurrent DOS® 5.0 and Microsoft® Windows 3.1 applications in the Solaris desktop environment. This is provided to satisfy the requirements of end users who currently have 80x86 systems.

   This capability is not an emulation; applications benefit from the underlying x86 hardware. Emulators such as SunPC™ from SunSelect are appropriate when the native instruction set on the platform, such as SPARC, is different from that used by the application. The ability to run a DOS application in Solaris for x86 is more comparable to the current SunSoft INTERACTIVE VP/ix, a product that runs MS-DOS applications under INTERACTIVE UNIX System V 3.2.

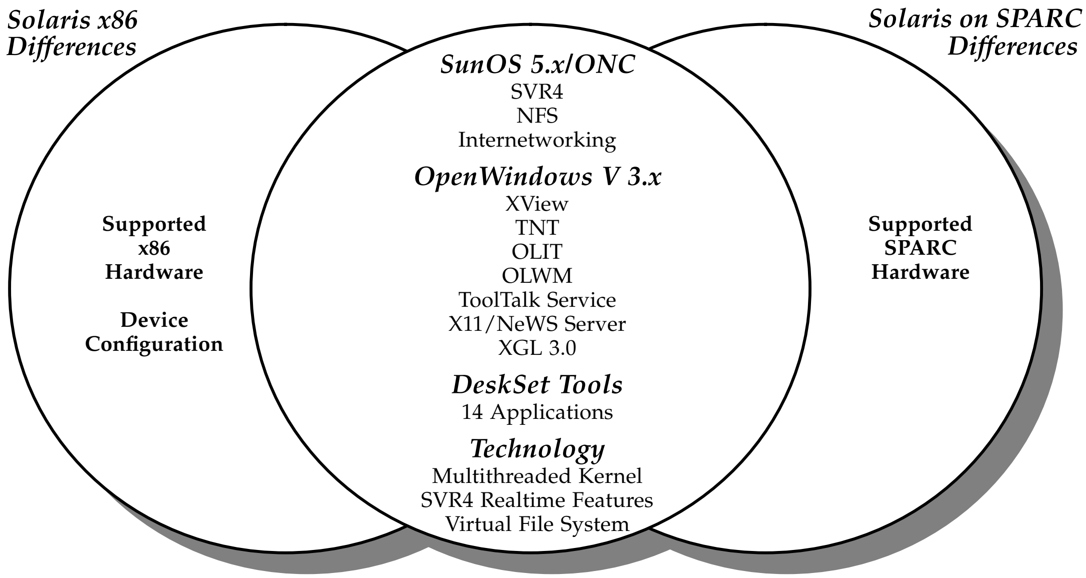

Figure 14-1 Solaris for x86 and SPARC Have Few Differences

Guidelines for Migrating Applications

Source Code Compatibility

The Solaris for x86 environment conforms to the same UNIX source code specifications as the SPARC version, including:

- POSIX P1003.1-1990
- X/Open Portability Guide 3
- System V Interface Definition, Issue 3
- MIT X11/Release 4

SPARC Solaris 2.x Migration to Solaris for x86

One common migration path for applications will be from Solaris 2.x on SPARC to Solaris 2.x for x86. For end-user applications, a recompilation of the code usually suffices.

However, software with hardware or kernel dependencies usually require changes to the source code. One class of dependency is when software relies on x86-specific or release-specific portions of the kernel such as:

- x86 device drivers (DDI/DKI)
- I/O bus architecture
- PC memory management techniques

Applications which use or generate hardware-dependent data files should allow for the difference in the byte-ordering between the x86 systems and other platforms.

INTERACTIVE UNIX to Solaris 2.x Migration

The migration path for applications coming from the System V Release 3 environment of INTERACTIVE UNIX versions 3.0 and 3.0.1 to the SVR4 environment of Solaris is similar to the transition between the SVID2 (System V Interface Definition 2) environment of SunOS 4.x to the SVID3 environment of Solaris 2.x. Often only a recompilation of the application is required; occasionally minor cleanup is needed.

Applications using the MOTIF toolkit should transition to the OPEN LOOK Intrinsic Toolkit (OLIT). OLIT provides a programming interface based on its Xt intrinsics definition. It enables easy migration to other Xt toolkits. It includes three dimensional user interface components, allows for customized applications, and provides enhanced intrinsics performance.

Changes to source code are needed in some unusual cases. XENIX calls must be replaced with calls to native services.

Binary Compatibility

The following binary compatibility specifications apply to Solaris 2.x.

SVR4 x86 Application Binary Interface (ABI)

The System V Release 4 Application Binary Interface (ABI) established a standard binary interface for application programs on systems that implement the interfaces defined in the System V Interface Definition, Issue 3 (see Chapter 8, SVID3 Compliance). This corresponds to UNIX System V Release 4.

The SVR4 ABI consists of two parts: a generic portion that describes the interfaces that are constant across all machine implementations, and a processor-specific portion describing the interface specific to a particular hardware architecture.

Solaris for x86 will conform to the SVR4 ABI and its x86 processor specification supplement (iABI).

BCS-1 (Binary Compatibility Specification)

The System V Binary Compatibility Specification (BCS) is a binary interface for applications that is based upon System V Release 3. Its purpose is to establish a standard binary interface for application programs on systems that have implemented the older System V Interface Definition, Issue 2. This includes implementations of System V Release 3.2.

Like the SVR4 ABI, the BCS consists of a generic portion and a processor-specific portion. Solaris for x86 includes most of the BCS-1, but not those portions relating to XENIX.

For further information, see the *System V Binary Compatibility Specification*.

Object Files

The native object file format for Solaris for x86 is ELF (Executable and Linking Format). In the same way that the SPARC port of the Solaris software allows dynamically linked SunOS 4.x executables using `a.out` object file formats to execute in a binary compatibility mode, Solaris for x86 allows most INTERACTIVE UNIX object files using the COFF object file format to run in a binary compatibility mode.

The method for providing binary compatibility for COFF object files is similar to the library mapping scheme used for `a.out` object files. Some of the calls are provided in the kernel, some in shared libraries, and possibly some user level commands.

System V Interface Definition

UNIX itself provides insight into the proper structure of code for portability by being available on a wide variety of platforms. To understand the architecture of UNIX, the components of the System V Interface Definition (SVID) are examined.

Some modules cannot be defined generically. They are defined as the System V ABI to a specific processor.

Existing iABI applications will run without recompilation on Solaris 2.x for x86.

Processor Architecture Differences

All programs consist of data structures and algorithms. As long as a program creates and consumes all of its data structures within the context of the same processor architecture, there is no problem. However, there are certain differences that exist between the SPARC and x86 processors. These differences become apparent when attempting to exchange data between the two.

Most significantly, there are differences between the ordering of the individual bytes within a data structure. Within a given 32-bit word, individual bytes are allocated from the "high" byte towards the "low" byte on the SPARC processor, but on an x86 processor, individual bytes are allocated from the "low" byte towards the "high" byte, as shown in the following illustration:

Figure 14-2 Byte Ordering Differences

SPARC Processor				**x86 Processor**			
0	1	2	3	3	2	1	0
High byte			Low byte	High byte			Low byte

Hence, the SPARC architecture is called a "Big Endian" machine, and the x86 architecture is called a "Little Endian" machine. This is the primary incompatibility in data structures between the SPARC and x86 architectures; it is at the heart of the porting issue.

In addition, there are also differences between the byte alignment of `double` and `long double` floating-point variables. The SPARC processor enforces 8-byte alignment on `double float` variables and utilizes a quad-word, 128-bit format for its `double float` variable, which are aligned upon quad-word (16 byte) boundaries. In contrast, the x86 processor enforces only 4-byte alignment upon its `double float` variables and utilizes a 96-bit IEEE extended precision arithmetic, which is aligned upon 12-byte boundaries.

What are the implications of these differences? First, the amount of data allocated for a structure is affected by how much space is allocated for its members. Second, unions must assume the alignment of their most strictly aligned member. The size of any object must always be a multiple of the object's alignment. Arrays of these elements will use the same alignment for all its variables. To illustrate, let's examine how identical data structures from a programmer's perspective get translated into the two architectures.

The following table shows that the corresponding data types double and long double do not agree.

Table 14-1 Processor Alignment Differences

	`sizeof` (bytes)		**Alignment (bytes)**		
Type	**SPARC**	**x86**	**SPARC**	**x86**	
`double float`	8	-	8	-	double-precision
		8		4	double-precision (IEEE)
`long double float`	16	-	8	-	quad-precision
		12		4	extended-precision (IEEE)

The critical area lies in the size and alignment of the two data types. As mentioned earlier, on the SPARC processor, the value of a double-precision occupies one doubleword, and the value of a quad-precision occupies two doublewords. The addresses of these values are in multiples of eight (a doubleword boundary). On the x86 processor, there is no requirement for doubleword alignment for a double-precision value. The double-precision values occupy one doubleword, and their addresses occur in multiples of four.

In Figure 14-2, you learned how bits are allocated. It shows that in the simplest case, there is no problem with porting between the SPARC processor and the x86 processor because the structures should align themselves the same on both architectures.

Summary

Solaris 2.x for x86 brings the benefits of the leading distributed computing operating system platform to the world's most common processor architecture, x86. With Solaris, SPARC and x86 users can share the same mature and robust distributed computing environment. Solaris for x86 further brings a new paradigm for computing to the desktops of PC users everywhere.

Uniformity of the operating software across platforms in a distributed computing environment is an enormous benefit to end users, application developers and system administrators. SPARC systems users will find that Solaris for x86 is almost identical to their SPARC-based environments. Solaris for x86 will put thousands of powerful workstation applications on personal computer desktops, and still run DOS and Microsoft Windows applications.

Software developers wishing to port applications between x86 and SPARC platforms will find that their code migrates uniformly and without difficulty. Developers report that the average migration time is only one day, and in many cases applications run with only a simple recompile.

With the exception of some configuration facilities and devices, Solaris for x86 and SPARC are the same. System administrators will find that Solaris for x86 simplifies their lives by minimizing the number of operating systems running on their networks. System administrators also will find that Solaris for x86 supports most popular PC hardware configurations.

Solaris provides the foundation for x86 users who want to build on the power of distributed computing today.

Bibliography

Solaris 2.x for x86 Platforms White Paper, Part Number 93033-001, SunSoft, Inc., Mountain View, CA.

Writing Portable Applications for the Solaris Operating Environment: x86 and SPARC Processors, Part Number 92196-001, SunSoft, Inc., Mountain View, CA.

System V Binary Compatibility Specification, ISBN 0-13-877614-8, Prentice Hall, Englewood Cliffs, NJ.

SPARCworks Technical White Paper: SPARCworks Toolset, Part Number 800-6943-01, SunPro, Inc., Mountain View, CA.

Solaris OpenWindows: OpenWindows V3 Collection, Part Number 91021-001, SunSoft, Inc., Mountain View, CA.

Solaris Commercial Extensions Products: Introduction to SunOS Security, Part Number 91029-002, SunSoft, Inc., Mountain View, CA.

Solaris SunOS: SunOS 5.0 System Administration, Part Number 91026-002, SunSoft, Inc., Mountain View, CA.

Solaris SunOS 5.0: SunOS 5.0 Multithreaded Architecture, Part Number 91033-002, SunSoft, Inc., Mountain View, CA.

Design and Implementation of Transport-Independent RPC, Part Number 91028-001, SunSoft, Inc., Mountain View, CA.

Solaris SunOS: SunOS 5.0 Release Report, Part Number 91002-002, SunSoft, Inc., Mountain View, CA.

Solaris SunOS 5.0: Multithreading and Realtime in Solaris—Terms and Concepts, Part Number 91023-002, SunSoft, Inc., Mountain View, CA.

ToolTalk Service Technical Report, Part Number 91031-002, SunSoft, Inc., Mountain View, CA.

Solaris OpenWindows: Introduction to the ToolTalk Service, Part Number 91031-002, SunSoft, Inc., Mountain View, CA.

Network Information Service Plus (NIS+), Part Number 91027-001, SunSoft, Inc., Mountain View, CA.

Project DOE: Distributed Objects Everywhere, Part Number 91035-001, SunSoft, Inc., Mountain View, CA.

Object Oriented Technology, Sun Microsystems Computer Corporation, Inc., Mountain View, CA.

Catalyst: A Catalog of International Third-Party SPARCware Solutions, Part Number 92019-001, SunSoft, Inc., Mountain View, CA.