# *Sun Technical Bulletin*

## *July 1992*

*Technical Information Services*

Sun Microsystems Computer Corporation
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 812-9207-01
Issue 1992 - 07

The *Sun Technical Bulletin* is distributed to customers with software/hardware or software-only support contracts. Send comments or corrections by electronic mail to *stb-editor@Sun.COM*, or by mail to:

Sun Technical Bulletin
Sun Microsystems Computer Corporation
2550 Garcia Avenue , M/S MPK02-04
Mountain View, CA 94043-1100  U.S.A.

# *Contents*


**1. STB Editor's Introduction** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **1-1**

*STB Overview* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1-1

*Updated Sun Operating System Support Matrix* . . . . . . . . . . . . . 1-4

*Updates* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1-6

Updated Sun Operating System Support Matrix (March-May 1992). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1-6

Using Maximum Tape Capacity (February 1992) . . . . . . . 1-7

Availability of Perl (December 1991, April 1992) . . . . . . . 1-7

**2. Multiprocessing** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-1**

*Multiprocessing Implementation in SunOS 4.1.2*. . . . . . . . . . . . 2-1

The Perils of Sharing . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-2

Uniformity and Uniqueness . . . . . . . . . . . . . . . . . . . . . . . 2-5

Interprocessor Communication and Control . . . . . . . . . . 2-6

Debugging 600MP Problems. . . . . . . . . . . . . . . . . . . . . . . 2-6

Further Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-7

**3. System Administration** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-1**

*Backup Copilot™: The Next Step in Backup and Restore Technology* . . . . . 3-1

Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-1

Analysis of Requirements . . . . . . . . . . . . . . . . . . . . . . . . . 3-2

A Step Forward: Backup Copilot . . . . . . . . . . . . . . . . . . . . 3-7

*The Perl Language* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-15

**4. Hardware** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-1**

*101A Keyboard* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-1

Product Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-1

Window System Overview . . . . . . . . . . . . . . . . . . . . . . . . 4-2

Questions & Answers . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-3

References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-9

**5. Languages** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-1**

*ANSI C Compliance* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5-1

Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5-2

Compiling with Compatibility Flags . . . . . . . . . . . . . . . . 5-3

Supplying Function Prototypes . . . . . . . . . . . . . . . . . . . . 5-3

Converting to ANSI C Function Declarations . . . . . . . . . 5-6

Correcting Compiler Warnings and Errors . . . . . . . . . . . . 5-7

Correcting Run-Time Inconsistencies . . . . . . . . . . . . . . . . 5-9

ANSI C Notes and Code Samples . . . . . . . . . . . . . . . . . . 5-10

*Unbundled Compilers* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5-25

**6. Software Release Levels** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **6-1**

*Current Sun Software Products and Release Levels* . . . . . . . . . . . 6-1

Operating Systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-1

Window Systems & Desktop Utilities . . . . . . . . . . . . . . . . 6-1

Sun Server & Database Software . . . . . . . . . . . . . . . . . . . . 6-2

Graphics Software . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-2

Network Management. . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-2

Software Development . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-3

Other Products . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-3

Connectivity Products . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6-4

**7. Notes & Comments**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-1**

*World Hotlines* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7-1

*Reporting Bugs* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7-3

Submitting Software Bugs: United States and Canada . . 7-3

Submitting Software Bugs: Europe . . . . . . . . . . . . . . . . . . 7-4

Submitting Software Bugs: Intercon. . . . . . . . . . . . . . . . . . 7-9

*Your STB Subscription* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7-12

**Index**

# STB Editor's Introduction 1

**NOTICE** — For the new Fiscal Year 1993, the *Software Technical Bulletin* (STB) is changing its name to the "*Sun Technical Bulletin*". This is a change that we have been anticipating for some time, due to the fact that the STB occasionally covers hardware issues for the interest of our software customers.

This name change does not signify any change in the general purpose of the STB, nor in its audience, nor in its contractual obligations. The STB is still primarily a software document, delivered to customers with software (or software-and-hardware) support contracts. The Bulletin still covers primarily software-related topics, but with periodic coverage of hardware issues that may be of concern to software users—documentation problems, new commands and capabilities, performance benchmarks, and so on.

Thank you for your continued interest in...the *Sun Technical Bulletin*.

## STB Overview

✍ This month's STB contains articles of general interest in the topics outlined below.

In this chapter, we have brought back a revised updating of Sun's support matrix for Operating System release levels and platforms.

Also, please note the *Updates* section at the end of the chapter. This section features new or updated information about articles from past issues of the STB. We appreciate the participation of our readership in calling this information to our attention.

## Multiprocessing

This month, "Multiprocessing Implementation in SunOS 4.1.2" describes the technical concepts involved in adapting traditional UNIX to a multiprocessing environment. The covered issues include shared global memory and kernel locks, CPU-unique memory, interprocessor communication, and debugging problems.

## System Administration

"Backup Copilot: The Next Step in Backup and Restore Technology" is a white paper produced by SunSoft, Inc., describing its Backup Copilot product. The paper describes the many inadequacies of the standard UNIX dump programs, and details how Backup Copilot can improve the efficiency and flexibility of a system administrator's backup scheme.

We also include "The Perl Language," a condensed version of a common information file on the Perl programming language.

## Hardware

"101A Keyboard" is a collection of frequently encountered issues regarding use of the 101A keyboard in the SunView and OpenWindows environments. The various setup commands are described, and a Question and Answer session addresses common problems with the keyboard and its distributed software.

## Languages

This month, we present the first in a series of articles relating to conversion of C programs for Solaris 2.0. "ANSI C Compliance" outlines the migration path from common C in SunOS 4.1.*x* to Sun ANSI C in Solaris 2.0. Included in the discussion are ANSI C compatibility flags, function prototypes, and miscellaneous changes in compiler behavior, illustrated by numerous code samples.

"Unbundled Compilers" briefly details some of the reasons behind Sun's decision to unbundle the C compiler from Solaris 2.0.

## Software Release Levels

This section includes the current Sun software products and release level tables. As of this issue, the product lists have been revised to be in closer relationship with the End User Price List.

## Notes & Comments

### World Hotlines

For Sun customers world-wide served by your local service groups, use the customer service telephone numbers listed in this monthly item. Also, look to this section during the upcoming year for details on your local support call policies and procedures.

### Reporting Bugs

The information in this monthly section continues to be expanded as Sun software service centers are added world-wide. The section includes a list of Sun service centers, addresses, e-mail hotlines, telephone hotlines, and FAX numbers.

### Your STB Subscription

This notice is published monthly, giving customers useful information regarding ordering multiple STB hardcopy subscriptions, and duplicating additional STB copies from a single "master" copy. This duplication permission is limited, as detailed in the note.

# Updated Sun Operating System Support Matrix

Sun's policy is to provide hardware and software support for out-of-production or transitioned systems and the latest compatible software for those systems for a minimum period of five years from the last order date.

The details of Sun's software support policy have been published in a Sun Operating System Support matrix in past STB issues. As illustrated in the support matrix, older versions of the operating system generally receive lower levels of software support than newer versions. Thus, we recommend that you upgrade your system to the most current versions to ensure you receive the best quality of support from Sun.

Since we continually enhance the operating system to meet our customers' needs, we promised to keep you abreast of updates to the support matrix via the STB. Now that SunOS 4.1.2 is available, we provide an updated matrix.

Sun uses "support levels" to describe its software support policy. Please note that the services described in these levels are subject to local availability—contact your local service manager should you have any questions.

## Level 1 Support

Level 1 offers the complete array of services contained in Sun's Personal AnswerLine Service, AnswerLine Service, and Software Update Service (see Note ①). Level 1 support also includes the full availability of options such as SunStart™, Sun's educational services, and supplementary consulting products and services.

## Level 2 Support

Level 2 includes Sun's Personal AnswerLine and AnswerLine levels of phone support, limited availability of software and documentation, bug report submission and tracking, and emergency fixes and workarounds (see Note ①). Level 2 support also includes availability of supplementary consulting products and services.

## Level 3 Support

Level 3 provides Time and Materials (T&M) support only.

---

**Note ① —** Software updates will only be available for the most current release. When a version of software has been designated as the final release for a platform, only critical bug fix releases will be available for that platform.

---

## Sun Operating System Software Support Matrix

The following tables illustrate Sun's OS software support matrix. See Note ② below regarding this section.

| **For Sun-2** | **3.5** | **4.0/4.0.1** | **4.0.3** | **(see Note ③)** |
|---|---|---|---|---|
| 1/91-1/92 | level 3 | level 3 | level 2 | |
| 1/92-1/94 | level 3 | level 3 | level 3 | |

| **For Sun-3** | **3.5** | **4.0/4.0.1** | **4.0.3/4.0.3e** | **4.1** | **4.1.1** | **(see Note ③)** |
|---|---|---|---|---|---|---|
| 1/91-1/92 | level 3 | level 3 | level 2 | level 1 | level 1 | |
| 1/92-1/95 | level 3 | level 3 | level 3 | level 2 | level 2 | |
| 1/95-1/96 | level 3 | level 3 | level 3 | level 3 | level 3 | |

| **For Sun386i** | **4.0.1i** | **4.0.2i** | **(see Note ③)** |
|---|---|---|---|
| 1/91-1/92 | level 2 | level 1 | |
| 1/92-1/95 | level 3 | level 2 | |
| 1/95-1/96 | level 3 | level 3 | |

| **For SPARC** | **3.5** | **4.0/4.0.1** | **4.0.3/4.0.3e** | **4.1** | **4.1.1** | **4.1.2** |
|---|---|---|---|---|---|---|
| 1/91-present | level 3 | level 3 | level 3 | level 2 | level 1 | level 1 |

**Note** ② — The above levels 1 and 2 are for contract customers only. Non-contract customers have the option to purchase support on a Time and Materials basis or use Sun's consulting services. The then-current unbundled products will be supported at the same level as the customer's operating system. Support for emergency fixes and workarounds may require the customer to upgrade to a more current version of the product.

**Note** ③ — These will be the final releases for these product lines, with the exception of critical bug fix releases. Sun's future SVR4-based operating system will function primarily on SPARC processor-based systems (Intel-based support for SVR4 is available with SunSoft's INTERACTIVE UNIX product line). Interoperability between Sun-3 or Sun386i systems and SPARC SVR4 systems will be fully supported through services such as NFS, NIS, remote login, and the like.

In addition, Sun is committed to supporting Sun-3 diskless clients running SunOS 4.1.1 on Solaris 2.0 servers.

# Updates

This section presents updates on subjects covered in past issues of the Software Technical Bulletin.

## Updated Sun Operating System Support Matrix (March-May 1992)

The text accompanying the support matrix published in three earlier issues contained some inaccurate or misleading information:

> Sun's future SVR4-based operating system will function only on SPARC processor-based systems.

The recent release of SunSoft's INTERACTIVE UNIX System V/386 Release 3.2 Operating System, Version 3.0, provides a path to Solaris 2.0 on Intel platforms. These products are sold by SunSoft directly to developers and are not available through the standard Price List.

> In addition, Sun is committed to supporting Sun-3 diskless clients running SunOS 4.1.2 on Solaris 2.0 servers.

This should have read "SunOS 4.1.1", which is the final supported SunOS release for Sun-3 systems.

This passage led to questions about how a Solaris 2.0 server can support a Sun-3 diskless client, if Sun-3 binaries are not included in the Solaris 2.0 distribution.

While Solaris 2.0 is a SPARC-only release, some customers who wish to upgrade to Solaris 2.0 SPARC servers may have diskless boot clients which cannot be upgraded. As a transition aid, Solaris 2.0 servers will support diskless boot clients running any SunOS 4.1.*x*-based release. The configuration takes extra steps and requires space on the server for the entire SunOS 4.1.*x* distribution. The SunOS 4.1.*x* release must be installed from CD-ROM, not tape.

Diskless boot clients of any architecture that runs a SunOS 4.1.*x* release can be served in this fashion, either Sun-3 or Sun-4. Solaris 2.0 has the generic capability to do this with any SunOS 4.1.*x*-based release, from SunOS 4.1 through SunOS 4.1.3, including Sun-3s running SunOS 4.1.1.

The restrictions above apply to diskless *boot* clients. All diskful ONC systems can use NFS to cross-mount filesystems, including home directories, as they have been able to in all previous SunOS releases. The NFS and TCP/IP protocols are unchanged in Solaris 2.0.

## *Using Maximum Tape Capacity (February 1992)*

This article described the use of the `tar` and `mt` commands to place multiple files on a single magnetic tape. Under the heading “Read/Retrieve Multi-File Archive Tape,” the article described the commands necessary to locate a file, including the following command:

```
% mt -f /dev/nrst0 bsf
```

In SunOS 4.1.1, executing this command on 1/4” SCSI tape drives generates the following error:

```
/dev/nrst0 bsf failed: Inappropriate ioctl for device
```

The `bsf` option is not supported under SunOS 4.1.1 for 1/4” SCSI tape drives (like the QIC-150). The `st` device driver in SunOS 4.1.2 is an improved version that should support the `bsf` option on SCSI drives. If you wish to pursue patching SunOS 4.1.1 rather than upgrading to 4.1.2, contact your Customer Support representative and reference bug numbers 1078062 and 1082260, and patch number 100581.

## *Availability of Perl (December 1991, April 1992)*

Many readers have requested online copies of the Perl scripts that have appeared in the STB. Some readers are unfamiliar with this language and have inquired about its availability.

In the *System Administration* chapter of this issue, we are including the article “The Perl Language,” which reprints excerpts from a common information file on the language. The entire file is available to readers upon request.

Further information can be obtained from Larry Wall, the author of the language, at e-mail address *lwall@netlabs.COM*.

We would like to encourage our readership to e-mail us any comments or suggestions regarding the content of the Sun Technical Bulletin, including requests for articles on specific subject matters. We are aiming at a broad spectrum of topics that are of interest to all sectors of our audience, both highly experienced technical people and relatively inexperienced users who are new to desktop workstations. We are not the appropriate channel for researching any and all technical problems that may come up in the field, but we will do our best to follow up on questions about articles that we have printed in the recent past.

Please let us know what you think, and we will do our best to accommodate you in future issues.

Thanks.

The STB Editor
*sun!stb-editor* or *stb-editor@Sun.COM*

# Multiprocessing 2

## Multiprocessing Implementation in SunOS 4.1.2

**By –** Jack Vogel, SMCC Corporate Technical Escalation – OS

The introduction of the SPARCserver 600MP class of machines is a significant milestone for Sun, for a couple of reasons. It is the first machine to use the SPARC International Version 8 Reference Memory Management Unit (SRMMU), as well as the Mbus module technology. But more importantly, it is the first multiprocessor (MP) system Sun has shipped.

Multiprocessing introduces a new level of complexity for both hardware and software. As the multiprocessing landscape is a varied one, it would be useful to position the 600MP within it. The machine is *tightly coupled*, meaning that all processors share access to the same physical memory. This is in contrast to *loosely coupled* designs, in which each processor has its own local memory. Tightly coupled systems also typically execute a single OS image, whereas loosely coupled machines execute individual iterations of an OS.

The 600MP system is of a *symmetric* design, meaning that each processor's hardware is equally capable of performing all the same functions as any other processor. This is in contrast to *asymmetric* designs, in which the hardware of one processor limits it to some specific functionality unlike that of the other processors. The best example is a master-slave design in which only one processor runs the core OS, and other processors only perform subservient tasks. [This design of the hardware should not be confused with the reference to SunOS 4.1.2 as “asymmetric” due to its single kernel lock: this is a separate issue of which we will say more later.]

Traditional UNIX was created in a uniprocessor environment, so adapting it to multiprocessing is a non-trivial task, and different vendors have approached the challenges differently. What we hope to accomplish in this article is to look at the key issues in the multiprocessing implementation in SunOS 4.1.2:

- sharing global memory and how the kernel lock allows this
- the unique memory of each CPU and how this was designed
- a quick overview of interprocessor communication
- practical issues in debugging multiprocessor problems

## *The Perils of Sharing*

As mentioned in the introduction, a multiprocessor system opens up a whole new world of computing capability but also brings with it a new class of problems as well. From the beginning, traditional UNIX was a “multitasking” operating system, meaning that different processes would share varying time-slices of a processor, giving the illusion of parallel processing. With a multiprocessor, this is no longer an illusion: with four processors, the machine is simultaneously executing four streams of instructions, potentially the same ones. It is sometimes difficult to picture this and to understand deeply the constraints on programming that it imposes.

It may be useful at this point to distinguish the multiprocessor from the parallel or massively parallel processor. The latter, such as Thinking Machines’ Connection Machine, is a specialized machine that uses compiler technology to break a single program into a multitude of tasks that can be run in parallel to produce a single result. In contrast, a multiprocessor, as we are referring to it here, only parallelizes the kernel, not the individual program (at least not until such parallel compilers are available).

The traditional UNIX kernel has the notion of critical sections of code. This means that system data is being modified and must be completed before further access. In many cases in which this involves a small number of instructions, it is accomplished by raising the priority to be uninterrupted during its execution. When larger bodies of code are involved in which you want to allow interruption, a kernel semaphore is created. This is usually coded in the following way:

```
int s = splvm();
while ( lock_is_not_free){
        set_the_lock_wanted;
        sleep(lock_address,some_priority);
}
set_the_lock_busy;
splx(s);
```

In the case of a multiprocessor, neither of these traditional schemes will work in general. Just raising the `spl` level will not do, because the interrupt can easily come in on another processor, causing the critical code to run in parallel there. The semaphore will also be inadequate, for it is possible for two processors to run this code simultaneously, both seeing the lock free and dropping through the `while` loop to enter the same code. What is needed is a new type of lock, called a *mutex* (mutual exclusion) or atomic lock, so that we are guaranteed that only one processor at a time can get through.

Fortunately, the SPARC instruction set provides the needed hardware support to do this. It has two instructions—`ldstub` and `swap`—that are guaranteed atomic between processors. Because `swap` is not implemented in all hardware, it is not used for critical kernel locks. The `ldstub` (load and store unsigned byte) instruction atomically sets all bits of the specified address *on*, and loads the previous value into a specified register. So the following simplified assembler creates an atomic lock:

```
set     _klock,%g1        ! get lock address
ldstub  [%g1],%g2
cmp     %g2,%g0           ! did we get it?
bne     1b                ! oops, nope try again
nop                       ! yep
```

This code fragment is referred to as a *spin-lock*: it sits in a tight loop until it gets the desired lock. Locking routines could be coded to try only once (or only some finite number of attempts) to get the lock.

Given the general notion of an atomic lock, the issue arises of how to implement such locks in the kernel. SunOS 4.1.2 and the 600MP machines themselves are sometimes referred to as “asymmetric” because there is a single kernel lock. As mentioned in the introduction, the machine hardware is fully symmetric, so this usage is not wholly accurate (perhaps we should say the kernel is asymmetric—or drop the terminology altogether).

In any case, it is true that the system has a single lock: its symbolic name is `klock`, and it is manipulated by the individual processors to determine which can run the kernel code protected by the lock. There are three different routines for dealing with the lock (we will see in a moment how each is used):

- `klock_enter()` is a spin-lock in the strict sense: it will not return until the lock is acquired.
- `klock_knock()` is a routine that only attempts to get the lock once, and returns success or failure.
- The special routine `klock_intlock()` is used in handling interrupts.

All entries to the kernel, or transitions from user mode to kernel mode, happen via *traps*. A trap is either a hardware or software event that causes the current instruction stream to be interrupted, and causes the processor to "jump" to some other address. The location of this jump is determined by the trap table and the numeric trap type. The table's address is contained in the hardware register `%tbr` (trap base register), and the trap number indexes into the 16-byte member of the table. Instructions at that address are executed upon receiving that trap. So while it is true in general that only one processor can run the kernel, this low-level code (which is in `locore.s` in the `sun4m` source directory) does run in parallel before acquiring the lock.

There are three general types of traps—think of them as three doorways into the kernel. *Hardware traps* are events like a page fault or execution of an illegal instruction. This type of event vectors to `trap()`, and at the very beginning of that routine there is a call to `klock_enter()` so only one processor at a time can get beyond that point.

Secondly, there are *software traps*, like system calls. In this case, the code vectors to the routine `syscall()`, and as in the case of `trap()`, the first routine is another call to `klock_enter()`.

Finally, *interrupts* are the most complex case. When an interrupt occurs, it calls `klock_intlock()`, which determines whether the lock must be acquired at all. Hard interrupts level 14 or 15 run without the lock. For interrupts that require the lock, the attempt is made to acquire it, but not as a spin-lock. If the lock cannot be obtained, the interrupt is forwarded to the CPU that has it. So for instance, clock interrupts occur on every CPU (level 10), but only one CPU ever runs the `hardclock()` routine, which processes the necessary information for all the other CPUs.

There are two special cases in which the kernel lock must be manipulated. These are kernel `lwp` (lightweight process) and STREAMS processing. This processing potentially occurs on the way out of the kernel back to user mode. In the case of a lightweight process, there is a call to `klock_enter()` to guarantee the lock. STREAMS processing is currently the one case in which `klock_knock()` is used. This call tries to get the lock, and if it fails, it simply defers doing the STREAMS processing.

## *Uniformity and Uniqueness*

We noted that the 600MP is tightly coupled, which means that all processors have shared access to all physical memory. All processors must also share a uniform virtual memory map. But it is equally true that some data must be unique to each processor. After all, each will be running a unique process, and thus must have a different `masterprocp` pointer, a different `uunix` pointer, and so forth. So the question is: how does the system implement this unique per-processor information?

SunOS 4.1.2 implements this information via unique mappings in the highest-level page table. Specifically, there is a new kernel structure declared: it is called the `PerCPU` structure. There is one for each possible CPU (there are four such structures on the 600MP). Each structure is a megabyte in size. This structure contains things like the pointers mentioned, multiple stack areas, and other data. The four copies of the structure will, of course, exist in different physical pages of memory.

For the system to work properly there are two requirements:

1. Each CPU must access its own unique data, but at the same virtual addresses.
   - There is what might be called a “magic” segment, `0xfd0`*xxxxx*. For whatever processor you are running on, this megabyte address range will map in the page tables to that processor’s unique `PerCPU` area.

2. Each CPU must also be able to access any other processor's data at some known kernel virtual address. This is accomplished by a dual mapping of these pages.
   - So that all processors can get access to any other processor's unique data, the virtual mapping `0xfe`***Y****xxxxx* (where ***Y*** is the `cpuid` (0-3) and *xxxxx* is the offset) will map to that unique memory for all processors. This is the manner in which uniqueness as well as uniformity is managed.

## *Interprocessor Communication and Control*

With multiple processors executing independent streams of execution, it is essential to be able to communicate and possibly to control from one CPU what the others are doing. For example, if a `panic` occurs on one CPU, the others must be made to halt execution so that shared memory can be fixed during a system dump. As another example, if the status of a page of shared memory is being changed, all the processors must be informed so that possibly cached information can be `sync`ed. This is accomplished at two levels:

- When running in the kernel, communication takes place via a number of routines called *cross calls*. These are the highest level "soft" interrupts that cause the other CPUs to do some task. In many cases, this task may be to just stop and wait; in other cases, it may be to `sync` shared cached information, or to perform some task another CPU needs to get accomplished.
- Each CPU has memory in its `PerCPU` area for communication with the PROM. This is called the "PROM mailbox." If one CPU suffers a fatal error or enters the PROM for whatever reason, other CPUs can be instructed what to do via this mailbox. It is examined when returning from a trap, and will determine if a CPU should cease execution or just pause (e.g., a user aborted by hitting `L1-A`, and everybody should wait; or one CPU had a "watchdog reset" and everybody should halt).

## *Debugging 600MP Problems*

In debugging problems on the 600MP series machines, there are a number of tips to keep in mind. First and foremost should be the complications introduced by multiprocessing. If a problem occurs on a 600MP, the system can be made to run in uniprocessor mode by altering the global variable `uniprocessor` to a non-zero value and rebooting. If the problem disappears, then it is MP-specific. If not, then it is `sun4m`-specific but not MP-specific (assuming it does not occur on any other SPARC platforms).

Additionally, when an MP problem is encountered, it is important to keep in mind the unique addresses of each CPU's `PerCPU` area. So for instance, looking at the `proc` structure pointer in each of the CPUs and other data. If one CPU `panic`ed, it may be important and relevant to examine what the other CPUs were doing at the time. With the first shipped SunOS 4.1.2 system, there are a couple of problems with using `adb` on core dumps. These problems are fixed with a forthcoming patch.

## *Further Study*

- To understand most fully the details of what is discussed here, there is no substitute for the source code. Most of the MP-specific changes to the OS are in the `sun4m` source directory. Pay particular attention to `machdep.c`, `locore.s`, and `trap.c`. In the general source, any MP changes are indicated by `#ifdef MULTIPROCESSOR` and can be fairly easily located.
- As a prerequisite, it is important to understand the SPARC general architecture. Obtain and study *The SPARC Architecture Manual*, SPARC International, Inc., Menlo Park, CA, 1992.
- Many available texts on the general principles and issues of multiprocessing should be consulted.

# System Administration 3

## Backup Copilot™: The Next Step in Backup and Restore Technology

Every UNIX site must protect its data through some kind of backup mechanism. The data must be protected from three kinds of loss: media loss (i.e., disk failure), user error (e.g., accidental removal), and catastrophe (e.g., a tornado destroys the entire building). Note that solutions such as disk arrays mitigate only one kind of data loss. Thus, backup is a way of life at UNIX sites. Unfortunately, some UNIX backup programs seem to suffer from a common set of problems. These include:

- Inability to work correctly with partitions mounted and in use (i.e., offline dumps are intrusive to the user community)
- Requirements for extensive operator intervention
- Lack of a good catalog of which files are on which tapes
- Poor end-of-media handling
- Slow dump speed
- Slow restore speed
- Tape management
- Inability to sequence automatically through several local and remote partitions

### Introduction

Automatic execution and sequencing of dumps coupled with a rudimentary tape management system dramatically eases the operations load of performing dumps. This paper describes a system that can free operators almost entirely from the daily dump burden and considerably ease the burden of file restoration. Many third-party vendors have addressed some of the problems listed above, but none has been able to solve them all. In this paper, the SunSoft Backup Copilot system is introduced specifically to alleviate these limitations. Backup Copilot is a member of the Commercial Extensions Products series that enhances the versatility and security of Solaris-based systems. Solaris is a single, integrated distributed computing solution. It is comprised of SunOS, ONC, OpenWindows, DeskSet, and OPEN LOOK.

## *Analysis of Requirements*

The dump program is the heart of a backup system. Whether one uses `dump` (sometimes known as `ufsdump`), `tar`, `cpio`, `bar`, or some other program, the goals are similar: copy a set of files (or an entire partition) to a backup medium in hopes that the files can later be read back if needed.

One major problem with dump programs, however, is that without kernel support (or a new file system), ensuring the validity of a dump tape is a difficult matter indeed. In fact, analyzing the behavior of a dump scheme when presented with moving subtrees inside the file system structure often exposes problems of safety. The popular `dump` program, when run with a file system mounted, can get into trouble when inodes contain a directory when the first scan occurs but are re-used in the interim and become regular files. It can produce a tape which is unreadable by `restore`.

Of course, all online schemes must contend with files being written precisely while they are dumped. TransArc's Andrew File System product solves this problem by creating copy-on-write inodes — but this requires that an entire new file system be used on your disks. Otherwise, backup programs do the best they can, sometimes warning the operator when a file was modified during backup, sometimes silently failing.

The above remarks merely point out the pitfalls of running the traditional `dump` program while file systems are mounted. The probability of some of these kinds of failures is very small — nonzero, but small. The risk is often acceptable when weighed against the cost of file systems being unavailable (often for hours). Several sites have used `dump` in online mode and never had a problem. Other sites were occasionally bitten by taking the chance.

At any rate, it is desirable to make certain promises about the quality of backup tapes produced. Programs with stronger guarantees (e.g., “never creates a tape which can’t be restored,” “always copies every file that is on disk when the dump began”) are more desirable than those with weaker guarantees (“always succeeds in favorable circumstances”).

### *Speed and Throughput Issues*

Another problem seen too often in the world of backup programs is speed (or lack of it). Some backup programs run as slowly as a few tens of kilobytes per second — certainly too slow in a world of multiple gigabyte disk systems. While affordable 9-track tapes have ranged from 450 KB/sec to 750 KB/sec in speed, the capacity of 9-track tapes is rapidly falling behind other peripherals, especially in light of the frequency of operator intervention required to change tapes.

Exabyte 8200 drives can achieve sustained throughput of just under 250 KB/sec — slower than 9-tracks but with a 2+ GB capacity. The new Exabyte 8500 drive runs twice as fast and has double the capacity — thus affording relatively intervention-free backups. Sites with 10 GB of storage that desire intervention-free backups can probably afford two drives. New compression devices claim compression ratios that range from 2 to 5 — thus extending the capacity of a single high-density drive to between 10 GB and 25 GB (with concomitantly higher transfer rates). Of course, `dump` should be able to drive these devices at high speed.

Optical disks and quarter-inch tapes are two other backup media in use — but they’re relatively slow when contrasted with 9-track drives. A backup program should be able to exploit 500 KB/sec drives in order to perform full backups on a 12 GB disk system in eight hours or so (four hours if one has two 8mm drives).

### *What* `dump` *Should Support*

The `dump` program should also keep a catalog of files it backs up. Furthermore, the backup system should include a means to sequence through local and remote partitions. (See the section entitled “The Sequencer” below.)

Additionally, the `dump` program should support label verification to reduce operator error and data loss associated with mounting incorrect tapes. Of course, producing tapes that are backward-compatible with the old `dump` program would be a definite plus.

dump should also reduce operator intervention by recognizing and dealing correctly with end-of-media marks. The current dump program relies on knowledge about tape length in order to know when to change tapes. A program which correctly recognized end-of-media could, in fact, enable packaging many file systems onto a single tape.

Any redesign of the dump program should investigate implementation of other niceties that users have requested for many years: mail notification in case of errors, true incremental dumps, and automatic device switchover. Furthermore, some sites request support for “parallel dumps” in which two independent sets of dumps (fulls and incrementals) are interleaved in time (e.g., on alternate days). This insulates a site against loss of data due to dump media failure. Support for this option requires dump to be able to use different dumpdates files.

### *Requirements for Restoring Files*

The UNIX restore program interface is notoriously poor. While dump tapes have an index preceding their data, tar tapes (and others) must be read completely to learn their contents.

A good UNIX backup scheme will support a catalog that is updated by each dump. Users should be able to peruse the catalog (using the familiar cd and ls commands) in order to locate versions of backed-up files and request their retrieval. Operators should intervene only to the extent required by security. Of course, the backup catalog must also protect against users seeing filenames they are not normally able to see.

While it is clear that users should be able to see information about files that is normally given by the ls or ls -algs commands, the question arises: how many versions of a file should a user see? The answer to the question should include enabling the user to:

- See the file system as it existed at any point in time
- See all versions of a given file
- See files back to the most recent level 0 dump
- See files all the way back to the first dump ever

Specifying what to restore (files, directories, an entire hierarchy) should be also simple and intuitive.

Another significant problem with `restore` is its speed. Many systems see a restore speed of eight hours per gigabyte of disk. This is intolerably slow — especially since situations which require large restores are precisely those in which users want to get back online quickly.

Why are `restore` and its brethren so slow? It's those synchronous writes. The operating system likes to make absolutely sure that file systems are not corrupted badly when a machine crashes and all the data stored in its memory are lost. The kernel carefully ensures that data from directory operations (like "create a file") is physically transferred to disk before other operations proceed. It is this synchrony that slows `restore`.

Providing a per-file system switch to disable this security feature (and increase the probability of totally scrambled disks after a crash) is certainly acceptable in the world of full restores. Since data transfer rates appear to increase by a factor of about four, restarting restores from scratch after the occasional crash still results in no net loss of time.

### *Requirements for Simple Tape Management*

Avoiding the overwriting of tapes is especially important for backups. A simple tape management scheme that utilizes such a simple approach as the label field in dump headers would be a dramatic improvement. Augmenting it with a small database to keep track of expiration dates would help even more by automating the choice of "which backup tape(s) to use today."

### *Requirements for Dump Sequencing*

Even novice UNIX system administrators regularly create customized programs and procedures for performing dumps. These programs are usually simple shell scripts, `awk` scripts, or C code.

The scripts sequence the dumps, remind the operator what's going on, and sometimes spread the workload of full dumps throughout the week. Some of the more sophisticated software attempts to perform low-level tape management using UNIX. Other administrators use tape drives (or other secondary storage devices) hosted by other machines (e.g., IBM 370 series machines) and rely on their tape management.

The scripts attack a well-defined (and commonly known) set of problems. They attempt to:

- Minimize operator intervention. It is much easier for an operator or administrator to spend ten contiguous minutes setting up a dump than ten minutes spent one minute at a time walking to the machine room, changing a tape, labeling a tape, and restarting the dump process.
- Sequence through file systems automatically. The `dump` program is not particularly good at this without assistance.
- Sequence through tapes (and tape drives) automatically. Few currently available programs can save files across tapes with correct end-of-media handling.
- Automate (where possible) tape handling and labeling. Overwriting a dump tape is one of the easiest ways to lose data. Good dump schemes attempt to guarantee that some minimal number of recent dumps is saved and that dumpsets have a minimal lifetime before they are recycled.
- Be simple and understandable. Complex schemes — particularly in multi-administrator environments — can be error-prone (especially in environments where errors are least tolerable).
- Support the “remote device user” security model so that clients do not require `root` access on servers in order to use the servers’ tape drives.

Some scripts go so far as to attack the problem of choosing tapes to store offsite. This is a difficult problem when, for instance, tapes contain many different dumpsets.

Supplying a program to ease setup and dumping of file systems is desirable for installations of all sizes. New sites (presumably with novice administrators) will benefit from dramatically eased installation and configuration. Sophisticated sites can exploit a good execution system to ensure that their dump requirements are met.

Attempting to create a single program that encompasses all the functionality of the hundreds of scripts in the world is extremely difficult. Nevertheless, it is important to be able to configure the software to emulate the most popular schemes. The richness of configuration options leads to high flexibility — and high complexity. This means that setting up the configurations becomes an error-prone or difficult process in itself.

### Requirements for Network Support

Of course, all the above requirements should function in a networked environment with file systems and tape drives spread throughout a network. Additionally, operators should be able to interact with all the components of an integrated dump system from anywhere within the network.

## A Step Forward: Backup Copilot

Backup Copilot enhances the `dump` and `restore` programs in addition to providing three components which provide an easy-to-configure, easy-to-run, and reliable scheme for dumping a site's file systems. These components include a *configurator*, a *sequencer*, and a *catalog*.

The *configurator* runs once to set up each configuration file and then later may run in edit mode as configurations change (e.g., adding a new disk or a new diskful system). The configurator presents a simple set of choices that administrators easily discern.

The *sequencer* interprets configuration files created by the configurator. It deduces parameters with which to invoke the `dump` program and then iterates those invocations (potentially both locally and remotely) to save a site's files. The sequencer is extremely robust and knows how to sequence dumps in the face of crashes, holidays, and unknown (frequent or occasional) invocations.

Backup Copilot keeps a *catalog* of dumped files and information about them. The `dump` program updates the catalog; the `recover` program reads the catalog and provides users with a hierarchical view of all backed-up files.

### The Sequencer

The sequencer runs on a machine which may have several configuration files (one for each set of dumps to be performed in a single group). Each configuration file uses its own `/etc/dumpdates.*` file. The file systems to be dumped need not be mounted on the machine on which the sequencer runs, though network communications must connect the machines. One can run as many sequencers as desired, but only one copy of each configuration file concurrently. It may be advantageous for sites owning multiple dump devices (e.g., two Exabyte drives) to split their dumping chores into two dumpsets in order to maximize use of the drives.

The sequencer not only interprets these configuration files but also interacts with the tape database. The `dump` program itself knows how to verify tape labels and creates new tape labels (using information supplied through command line arguments and potentially through a data file created for this purpose).

The sequencer also keeps complete logs of its activity so that administrators can gain confidence that dumps are being performed as expected. The logs reside in the `/var/adm/dumplog` directory and include: tape request messages, all dump output, tape usage information, and error information. A typical site will see from 5 KB to 50 KB of data logged daily. Each message includes the date, time, machine, and configuration file that was running to generate the log message.

As one would expect, the sequencer performs both full and incremental dumps. The sequencer, the `dump` program, and the file catalog support a switch that specifies true incrementals ("files changed since the most recent of any level 0-9 dump") rather than the standard level 9 incremental ("any files changed since the most recent of any level 0-8 dump").

The sequencer supports cyclical dump schemes (using the level numbers of the standard `dump` program). These are typified by a daily sequence like `05555` which means: on Monday ("the first day"), do a full dump, then on Tuesday, Wednesday, Thursday, and Friday perform level 5 incrementals. Some file systems in a dumpset may use a `50555` or `55550` cyclical dump in order to spread the full-dump load across different days ("staggering").

A more complex cyclical dump scheme includes one very similar to Epoch's baseline dump: `08888588885888858888`. In this scheme, a full dump is performed only every four weeks (that is, every 20 invocations). Daily incrementals and weekly "super-incrementals" follow the full dump. Extending the sequence increases the full-dump interval — not necessarily advisable since susceptibility to tape errors and data loss is increased.

The configuration and sequencer programs run without knowledge of the precise frequency that dumps are to be performed. Knowledge of holidays is complex and requires operator intervention. This configuration file design obviates the need to know anything other than "this program is being run today and should perform the next dumps in sequence."

```
tapelib            tapedata
dumpmach           indexhost
dumpdevs           /dev/nrst16 remote_system:/dev/nrst16
block              64
tapesup            2
notify             smith
rdevuser           rdev
longplay

#     level    multiple    days    min available
keep     0         1        15           3
keep     0        26        -1           3
keep     5         1         5           3

mastercycle 00014

fullcycle 00001 -/            >0555555555
fullcycle 00001 -/usr         >5055555555
fullcycle 00001 -/mnt         >5505555555
fullcycle 00001 -/usr2        >5550555555
```

*Figure 3-1* Sequencer Configuration File

Figure 3-1 depicts a sample dump configuration file. Here's how it works:

Blank lines and comments preceded by '`#`' are ignored.

The `tapelib` parameter names the tape library to use. See the section below on tape libraries.

The `dumpmach` parameter tells which machine owns the online catalog of dumped files. This can be the local machine or a remote machine.

The `dumpdevs` line has a whitespace-separated list of tape devices to use. This list should be homogeneous in type (e.g., all 9-track tapes or all Exabyte tapes).

The `block` line specifies the block size that `dump` should use.

The `tapesup` keyword specifies the default number of tapes to "plan ahead" for potential use. This number of tapes will be "reserved" (for 23 hours) in the tape library each time the sequencer is invoked. The operator is informed which tapes are reserved and hence will be used as the dump session progresses.

The notify line specifies the name of the user(s) who should be notified by e-mail in case of errors.

The rdevuser line specifies which user should be used as the login user of choice for the rmt remote tape protocol (thus relieving the server from having to trust clients with root access).

The longplay specification indicates that tapes will be left in the drive between successive invocations of the sequencer (thus reducing operator intervention even more, since tapes need only be changed every several days instead of daily).

### *How Long to Keep Tapes*

The next section describes how long to keep dump tapes before they are reused. Each keep line has four fields besides its keyword: the level described by this line, the multiple, the minimum number of days to keep this kind of tape, and the minimum number of this kind of tape to keep available on the shelf before being reused. The multiple works like this: every dump has a fullcycle number (0 for the first time through the list of file systems, 1 for the second, 2 for the third, and so on). When the fullcycle number is a multiple of the number in the second column, then that particular column applies. A multiple of 1 represents every dump; a multiple of 2 represents every other dump, and so on.

Of course every dump tape fullcycle number is a multiple of 1. The sequencer figures the most restrictive set of keep criteria for any given tape (so each 26th tape is kept a long time, even though the fullcycle number is a multiple of 1 in addition to 26).

Finally, the last lines describe not only the scheme to be used for dumping the file systems but also contain state information about current and recent dumps.

The mastercycle line tells how many times the sequencer has completely traversed this particular dumpset. Its initial value is 0. Additionally, the tape database keys expiration dates off both the date and the mastercycle number. It calculates the longest sequence of tapes that can intervene before, say, three copies of a level 5 incremental. Adding this length to the mastercycle number gives an upper bound of a “safe” expiration cycle for tapes in the database.

Each subsequent line contains information on file systems to be dumped:

- The keyword `fullcycle`
- The number of times this particular file system has completed a full dump
- A +, –, or * with the following interpretations:

  \+ means this file system has been dumped during this cycle

  – means it hasn't

  * means a dump was attempted but failed
- The name of the disk device to dump (potentially including a machine name to indicate remote dumps)
- A sequence of digits (including `x` for true incremental), indicating dump levels
- An indicator (>) that shows the next dump to be performed on this file system; it is advanced at the same time the "–" changes to a "+" in a given `mastercycle` and resets to the beginning of the level sequence before pointing to the newline

The sequencer traverses the dumpset using the configuration information to format dump commands and execute them. It updates the configuration file in place by moving the indicator (>) to show which is the next dump to perform and changing the "–" to a "+" (or "*" in the case of failure).

This sequencing scheme has several interesting properties:

- No knowledge of holidays or other service interruptions is required.
- No knowledge of frequency of execution is necessary: the sequencer always does "the next right thing."
- Crash behavior is quite benign. Sequencing resumes where it left off (potentially dumping a partition twice).
- It is simple to understand and implement.
- It is easily extensible to "parallel dump sets" in which two set of `dumpdates` files and sequences are used to increase resiliency in the face of tape loss.

## Configuration

Two different configuration programs create configuration files. The resulting configuration files are not necessarily different; the configurators are differentiated by the complexity of options they present to the user.

The simplest, `dumpconfez`, takes about five minutes to run, requires relatively low sophistication, mostly sets defaults for configuration parameters, and sets up all other required configuration files. This section will concentrate on `dumpconfez`.

The technically demanding `dumpconfig` takes much longer to run (15-30 minutes), requires relatively sophisticated users, and has few defaults.

Any configuration file is easily edited with `curses`-based editor (`dumped`). The `dumpconfez` configuration program asks additional explicit questions about:

- File catalog machine
- Dump sequences
- List of people to notify of successes and failures
- Long-term drive residency
- Tape blocking factor
- Remote device user
- Expiration criteria
- Parallel dumpsets
- Tape drives to use
- File systems to dump

Once these questions are answered, `dumpconfez` creates both a configuration file for daily backups and a configuration file for the occasional archival backup. It also sets up all other required configuration files (optionally including `root`'s `cron` entry) and starts the daemons required by the backup system.

## Tape Management

Both the `dump` program and the sequencer know how to manipulate named tapes in a rudimentary way. Falling far short of the protection and interchangeability afforded by ANSI labeled tapes, the tape library nevertheless ensures that tapes are not used before their expiration date has elapsed and that tapes are not inadvertently overwritten because of a tape selection or mounting error.

After tapes are initially used (and the labels are written), no maintenance is required.

## Disaster Recovery

Once catastrophe strikes, it is important to be able to recover an entire file system hierarchy — even if the catalog of files (with the names of the tapes upon which they are kept) is lost.

To cope with this potential problem, `dumpex` copies tape library information from the database as the last dump each time `dumpex` is invoked. The tape name and file number is mailed to the notification list. Administrators are additionally encouraged to keep the "last tape written" in a special place (until a new tape replaces it). This way, the tape is easily found.

A recovery program reads the tape and instructs an administrator with the sequence of tapes to load and programs to run to recover the entire database. Once the database is recovered, file recovery is simple.

## Security Issues

Security issues for any dump system revolve around access to tapes and tape drives. Access to a tape implies access to any dumped file on that tape, hence invalidating any security notion. Access to tape drives is often a weak point, since dump tapes may be left on the drive after a dump is complete. Obviously, physical access to tapes invalidates security for a site.

`Root` access is another problem with dump programs which must be able to read every file on the system. The `dump` program only dumps devices for which the real `uid` and real `gid` of the invoker have read permission. The `restore` and `recover` programs run `setuid` "`root` only" while making network connections. The restoration of `setuid` programs follows rules designed to inhibit security problems.

In the catalog of dumped files, viewing of filenames is restricted by file permissions, just like UFS file systems. Users only see the catalog for the machine upon which they are executing (thus avoiding impersonation problems).

In the world of network security, Backup Copilot does not use encrypted data, and thus is vulnerable to programs which use the network interface tap. Spoofing (i.e., of the operator monitor) is quite possible. No special user authentication is performed; secure RPC is not used.

## *Backup Copilot Limitations*

Highly secure customers (e.g., C2 kernels) should not use any remote dump program. Customers with less than 1 gigabyte of disk who already have a dump scheme will benefit most from ease of frequent backups and ease of `restore`.

Customers with difficult requirement (e.g., staging dumps to empty partitions for a day, then dumping the partitions) may find Backup Copilot a bit of a bother.

Customers must be running SPARC machines and SunOS 4.1.*x* to take advantage of Backup Copilot.

## *Conclusion*

While this program is a logical next step in bringing backups into the modern age, it is understood that new media (e.g., optical disk jukeboxes, digital optical tape jukeboxes, and other peripherals) will enable dumps and restores to be completely automated.

# The Perl Language

Perl is a programming language written by Larry Wall (*lwall@netlabs.COM*). The current Perl version is 4.019 (version 4, patchlevel 19). This article contains answers to some of the most frequently asked questions about Perl.

1. *What is Perl?*

   Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It combines some of the best features of C, `sed`, `awk`, and `sh`, so people familiar with those languages should have little difficulty with it. (Language historians will also note some vestiges of `csh`, Pascal, and even BASIC-PLUS.) Expression syntax corresponds quite closely to C expression syntax.

   Unlike most UNIX utilities, Perl does not arbitrarily limit the size of your data—if you've got the memory, Perl can slurp in your whole file as a single string. Recursion is of unlimited depth. And the hash tables used by associative arrays grow as necessary to prevent degraded performance.

   Perl uses sophisticated pattern matching techniques to scan large amounts of data very quickly. Although optimized for scanning text, Perl can also deal with binary data, and can make `dbm` files look like associative arrays (where `dbm` is available). `Setuid` Perl scripts are safer than C programs through a dataflow tracing mechanism which prevents many stupid security holes.

   If you have a problem that would ordinarily use `sed` or `awk` or `sh`, but it exceeds their capabilities or must run a little faster, and you don't want to write the job in C, then Perl may be for you. There are also translators to turn your `sed` and `awk` scripts into Perl scripts.

2. *Where can I get Perl?*

   From any `comp.sources.misc` archive. Initial sources were posted to Volume 18, Issues 19-54 at patchlevel 3. The Patches 4-10 were posted to Volume 20, Issues 56-62.

The following machines, at the very least, definitely have it available for anonymous FTP:

```
ftp.uu.net                        137.39.1.2
archive.cis.ohio-state.edu        128.146.8.52
jpl-devvax.jpl.nasa.gov           128.149.1.143
```

If you are in Europe, you might using the following site (this information thanks to "Henk P. Penning" at *henkp@cs.ruu.nl*):

> Perl stuff is in the PERL directory on `archive.cs.ruu.nl` [`131.211.80.5`]
>
> Email: Send a message to *mail-server@cs.ruu.nl* containing:
>
> ```
> begin
> path your_email_address
> send help
> send PERL/INDEX
> end
> ```
>
> The `path` line may be omitted if your message contains a normal `From:` line.
>
> You will receive a help-file and an index of the directory that contains the Perl stuff.

3. *How can I get Perl via UUCP?*

You can get it from the site `osu-cis`; here is the appropriate information, thanks to J. Greely (*jgreely@cis.ohio-state.edu* or *osu-cis!jgreely*).

> E-mail contact:
>
> `osu-cis!uucp`
>
> Get these two files first:
>
> `osu-cis!~/GNU.how-to-get.`
>
> `osu-cis!~/ls-lR.Z`
>
> Current Perl distribution:
>
> `osu-cis!~/perl/4.0/kits@10/perl.kit`***xx***`.Z` (***xx***=01-37)

How to reach `osu-cis` via `uucp` (L.sys/Systems file lines):

```
#
# Direct Trailblazer
#
osu-cis Any ACU 19200 1-614-292-5112 in:--in:--in: Uanon
# Micom port selector, at 1200, 2400, or 9600 bps.
# Replace ##'s below with 12, 24, or 96
#   (both speed and phone number).
#
osu-cis Any ACU ##00 1-614-292-31## "" \r\c Name? osu-cis
nected \c GO \d\r\d\r\d\r in:--in:--in: Uanon
```

Modify as appropriate for your site, of course, to deal with your local telephone system. There are no limitations concerning the hours of the day you may call.

Another possiblity is to use UUNET, although they charge you for it. You have been duly warned. Here's the advertisement:

> Anonymous Access to UUNET's Source Archives
>
> 1-900-GOT-SRCS
>
> UUNET now provides access to its extensive collection of UNIX-related sources to non-subscribers. By calling 1-900-468-7727 and using the login "`uucp`" with no password, anyone may `uucp` any of UUNET's online source collection. Callers will be charged 40 cents per minute. The charges will appear on their next telephone bill.
>
> The file `uunet!/info/help` contains instructions. The file `uunet!/index//ls-lR.Z` contains a complete list of the files available and is updated daily. Files ending in `Z` need to be uncompressed before being used. The file `uunet!~/compress.tar` is a `tar` archive containing the C sources for the `uncompress` program.
>
> This service provides a cost-effective way of obtaining current releases of sources without having to maintain accounts with UUNET or some other service. All modems connected to the 900 number are Telebit T2500 modems. These modems support all standard modem speeds including PEP, V.32 (9600), V.22bis (2400), Bell 212a (1200), and Bell 103 (300). Using PEP or V.32, a 1.5 megabyte file such as the GNU C compiler would cost $10 in

connect charges. The entire 55-megabyte X Window system V11 R4 would cost only $370 in connect time. These costs are less than the official tape distribution fees and they are available now via modem.

UUNET Communications Services
3110 Fairview Park Drive, Suite 570
Falls Church, VA 22042
+1 703 876 5050 (voice)
+1 703 876 5059 (fax)
*info@uunet.uu.net*

4. *Where can I get more documentation and examples for Perl?*

If you've been dismayed by the ≈75-page Perl `man` page, you should look for "The Camel Book," written by Larry and Randal L. Schwartz (*merlyn@iwarp.intel.COM*), published as a Nutshell Handbook by O'Reilly & Associates and entitled *Programming Perl*. Besides serving as a reference guide for Perl, it also contains tutorial material and is a great source of examples and cookbook procedures. The code examples contained therein are available via anonymous FTP from `ftp.uu.net` in the file `/published/nutshell/perl/perl.tar.Z` for your retrieval. Corrections and additions to the book can be found in the Perl `man` page right before the `BUGS` section under the heading `ERRATA AND ADDENDA`.

If you can't find the book in your local technical bookstore, the book may be ordered directly from O'Reilly by calling 1-800-DEV-NUTS (1-800-338-6887) in North America or 1-707-829-0515. Autographed copies are available from TECHbooks by calling 1-503-646-8257 or mailing `info@techbook.COM`. Cost is ≈30$US for the regular version, 40$US for the special autographed one. The book's ISBN is 0-937175-64-1.

For other examples of Perl scripts, look in the Perl source directory in the `eg` subdirectory. You can also find a good deal of them in the `pub/perl/scripts` subdirectory on `tut.cis.ohio-state.edu` .

Another source for examples, currently only for anonymous FTP, is on `convex.com` [`130.168.1.1`]. This contains, among other things, a copy of the newsgroup up through August 1991, a text retrieval database for the newsgroup, a rather old and short `troff` version of Tom Christiansen's Perl tutorial (this was the version presented at the Washington D.C. USENIX), and quite a few of Tom's scripts. You can look at the `INDEX` file in

/pub/perl/INDEX for a list of what's in that directory. In the future, monthly updates of all the newsgroup's articles will be placed there, and the by-subject indexing into subfolders will be completed.

Larry Wall has published a 3-part article on Perl in *Unix World* (August through October of 1991), and Rob Kolstad also had a 3-parter in *Unix Review* (May through July of 1990).

A nice reference guide by Johan Vromans (jv@mh.nl) is also available; it is distributed in LaTeX (source) and PostScript (ready-to-print) forms. Obsolete versions may still be available in TeX and troff forms, although these don't print as nicely. The official kit includes both LaTeX and PostScript forms, and can be FTP'ed from archive.cs.ruu.nl [131.211.80.5], in the file DOC/perlref-4.010.2.1.tar.Z. The reference guide comes with the O'Reilly book in a nice, glossy card format.

Additionally, USENIX and SUG have been sponsoring tutorials of varying lengths on Perl at their system administration and general conferences, taught by Tom Christiansen (*tchrist@convex.com*) and/or Rob Kolstad (*kolstad@usenix.org*); you might consider attending one of these. Special cameo appearances by these folks may also be negotiated.

You should definitely read the USENET comp.lang.perl newsgroup for all sorts of discussions regarding the language, bugs, features, history, humor, and trivia. In this respect, it functions both as a comp.lang.* style newsgroup and also as a user group for the language; in fact, there's a mailing list called "perl-users" that is bidirectionally gatewayed to the newsgroup. Larry Wall is a very frequent poster here, as well as many (if not most) of the other seasoned Perl programmers. It's the best place for the very latest information on Perl.

# Hardware 4

## 101A Keyboard

**By –** Alan Mark Levinson, U.S. Support Center

This article is a collection of frequently encountered issues regarding use of the 101A keyboard in the SunView and OpenWindows environments. A Question and Answer section is included to address frequently asked 101A keyboard questions.

### Product Overview

The AT101 (Marketing part number X912, Spares part number 320-1070-01) is an IBM PC AT101-style keyboard that has become the quasi-standard among PC customers.

#### Software Requirement

The required software for this keyboard is included in SunOS releases 4.1.2 and subsequent releases. Software is also available for SunOS releases 4.1 and later on a CD which is shipped with the keyboard in the 101A Accessory Kit (part number 595-2221-01).

The 101A software (part number 704-2839-10) included in the distribution is required for SunView, OpenWindows 2.0, and DOS Windows; it is *not* required for OpenWindows 3.0.

### Supported Platforms

The AT101 keyboard is supported on all Sun-3 and Sun-4 machines. Software can be accessed via a CD or exported from an NFS mounted file server.

## Window System Overview

The 101A keyboard functions under SunView, OpenWindows 2.0, and OpenWindows 3.0. The default directory for the installed software is `/usr/101a_kbd`, but the user can change this directory during installation. References below to the directory should be changed if necessary.

### SunView Setup

After installing the 101A software, refer to the instructions regarding use of the 101A keyboard found in the following file:

```
/usr/101a_kbd/Mail.101A.SunView
```

You should execute one of the setup commands described below, depending upon whether you would like your keyboard set up as Sun-compatible or PC-compatible.

#### Sun-Compatible

To set up the Sun-compatible keyboard, execute the following setup command upon login:

```
% loadkeys /usr/101a_kbd/loadkeys.101sun
```

The Sun-compatible layout will remap some of the keys on the edit (middle) keypad to support acceleration of window and selection service operations. For additional information regarding these accelerator keys in SunView, refer to the *SunView User's Guide* (part number 800-3831-10) sections on `Cut`, `Paste`, `Copy`, `Undo`, `Props`, `Find`, `Front`, and `Open`.

Middle keypad operations are set as follows:

| LABEL | | OPERATION |
|---|---|---|
| `Printscreen` | → | Front |
| `ScrollLock` | → | Open |
| `Insert` | → | Paste |
| `Home` | → | Copy |
| `PageUp` | → | Props |
| `Delete` | → | Cut |
| `End` | → | Find |
| `PageDown` | → | Undo |

You can customize the layout of your keyboard using the `loadkeys` options. For additional information, refer to the `man` page for `loadkeys`.

#### *PC-Compatible*

To set up the PC-compatible keyboard, execute the following setup command upon login:

```
% loadkeys /usr/101a_kbd/loadkeys.101
```

---

**Note –** You could execute the appropriate setup command upon logging in or put the appropriate command in your shell startup file (e.g., `.login` or `.cshrc` for the C shell, `.profile` for the Bourne shell).

---

### *OpenWindows 2.0*

For instructions regarding use of the 101A keyboard while running OpenWindows 2.0, refer to the instructions in the following file:

```
/usr/101a_kbd/Mail.101A.OpenWindows
```

### *OpenWindows 3.0*

The 101A software distributed on CD is not used by OpenWindows 3.0. This release of OpenWindows has built-in keyboard support.

## *Questions & Answers*

1. **How can I verify that OpenWindows has determined that it is running with a 101A keyboard? How can I verify that the layout intended, either Sun or PC, is in effect?**

   There is no easy way to determine this under OpenWindows 2.0. A bug has been filed (bugid number 1072683). There is no workaround.

   OpenWindows 3.0, on the other hand, has two NeWS operators called *keyboardtype* and *keyboardlayout* which can be used to determine which keyboard type is running. The NeWS operator functions are as follows:

   | | |
   |---|---|
   | *keyboardtype* | Determines the keyboard ROM-stored type |
   | *keyboardlayout* | Determines the DIP switch setting |

To obtain information from the NeWS operators, issue the following PostScript/NeWS shell (`psh`) command at the prompt:

```
% psh -i
```

where `-i` specifies an interactive session. The following interactive exchange with `psh` will determine the type and layout (**`bold font`** indicates the commands you would enter):

```
Welcome to X11/NeWS Version3
keyboardtype ==
4
keyboardlayout ==
19
quit
%
```

The responses indicate a keyboard type of `4` and a keyboard layout of `19`.

Located in the `$OPENWINHOME/etc/keytables` directory is a file named `keytable.map`. The type and layout numbers are decoded therein. The file includes the following lines:

```
Type   Layout  Filename
0       0      US4.kt # Default keytable
#
2       0      US2.kt
3       0      US3.kt
#
#Type   4      Keyboards
#
4       0      US4.kt # Sun US Type 4 keyboard
4       1      US4.kt # Also Sun US Type 4 keyboard
4       2      FranceBelg4.kt
4       3      Canada4.kt
4       4      Denmark4.kt
4       5      Germany4.kt
4       6      Italy4.kt
4       7      Netherland4.kt
4       8      Norway4.kt
4       9      Portugal4.kt
4       10     SpainLatAm4.kt
4       11     SwedenFin4.kt
4       12     Switzer_Fr4.kt
4       13     Switzer_Ge4.kt
4       14     UK4.kt
```

```
4       16    Korea4.kt
4       17    Taiwan4.kt
#
# Uncomment either the PC compatible or Sun compatible layout
4       19    US101A_PC.kt # Sun 101A -- PC compatible layout
#4      19    US101A_Sun.kt # Sun 101A -- Sun compatible layout
#
4       32    Japan4.kt
#
# Type 5 Keyboards
#
#
# Type 5 keyboards identify themselves as type 4 for backward
compatibility
#
4       33    US5.kt
4       34    US_UNIX5.kt
4       35    France5.kt
4       36    Denmark5.kt
4       37    Germany5.kt
4       38    Italy5.kt
4       39    Netherland5.kt
4       40    Norway5.kt
4       41    Portugal5.kt
4       42    Spain5.kt
4       43    Sweden5.kt
4       44    Switzer_Fr5.kt
4       45    Switzer_Ge5.kt
4       46    UK5.kt
4       47    Korea5.kt
4       48    Taiwan5.kt
4       49    Japan5.kt
```

You will notice that almost everything shows a keyboardtype of `4` (including type 101 and type 5 keyboards), so you have to look at the layout column as well to figure out which keyboard is actually in use. Search for the lines that show `4` in the `Type` column and `19` in the `Layout` column:

```
4       19    US101A_PC.kt # Sun 101A -- PC compatible layout
#4      19    US101A_Sun.kt # Sun 101A -- Sun compatible layout
```

The line containing `US101A_Sun.kt` (indicating a Sun-compatible layout) is "commented out" with a pound (`#`) sign. The uncommented line indicates a PC-compatible layout. Therefore, this system is running the 101A PC keyboard type.

2. **Under OpenWindows 2.0, how can I change from Sun layout to PC layout or vice versa?**

   Execute one of the setup commands below, depending upon whether you want the 101A keyboard to be setup as PC-compatible or Sun-compatible.

   ```
   % $OPENWINHOME/lib/setup.101A.pc
   % $OPENWINHOME/lib/setup.101A.sun
   ```

   You must restart OpenWindows 2.0 after running the appropriate setup command to invoke the new layout.

3. **Under OpenWindows 3.0, how can I change from Sun layout to PC layout or vice versa?**

   By default, OpenWindows 3.0 comes up with the keyboard in PC layout.

   To switch to the Sun layout, change the commented-out line (#) in the `$OPENWINHOME/etc/keytables/keytable.map` file as follows:

```
#4      19    US101A_PC.kt # Sun 101A -- PC compatible layout
4       19    US101A_Sun.kt # Sun 101A -- Sun compatible layout
```

   You must restart OpenWindows 3.0 after making the change to invoke the new layout.

4. **Are there differences between OpenWindows 2.0 and OpenWindows 3.0 that would cause the keyboard to work in one but not the other? What are the required steps to set up and run the 101A Keyboard under OpenWindows 3.0?**

   Yes. OpenWindows 3.0 has support for the keyboard built in, unlike OpenWindows 2.0 which required installation of the 101A software. However, there are a number of issues which need to be addressed when upgrading from OW 2.0 to OW 3.0.

   If you installed the 101A software needed to run OpenWindows 2.0 and then upgraded to OpenWindows 3.0, the following changes need to be made:

   a. **Delete the following two lines from your `~/.Xdefaults` file:**

   ```
   OpenWindows.FrontKey: F32
   OpenWindows.OpenKey: F33
   ```

   [To activate the OpenWindows 3.0 counterparts to `FrontKey` and `OpenKey`, see question #7 below.]

b. **Remove the following lines which invoke the `.xmodmaprc` file that the 101A software added to your `$HOME/.xinitrc` file:**

```
if [ -f $HOME/.xmodmaprc ]; then
xmodmap $HOME/.xmodmaprc
fi
```

Removing the `$HOME/.xmodmaprc` file is also recommended but not required.

5. **How can I get the gray Arrow Keys to autorepeat under OpenWindows 2.0 and OpenWindows 3.0?**

   The gray arrow keys will not autorepeat under OpenWindows 2.0. Arrow key functionality has been added to OpenWindows 3.0.

   Under OpenWindows 3.0, the gray arrow keys work fine but do not autorepeat by default.

   Use the `xev` (*print contents of X events*) command to get the keycodes of the keys (refer to `man` page for `xev`). Use the `xset` (*user preference utility for X*) command to enable autorepeat (refer to `man` page for `xset`).

   To prevent having to type these statements each time the window system is invoked, put lines like the following in your `$HOME/.xinitrc` file:

```
xset r 27
xset r 31
xset r 34
xset r 35
```

6. **How can I institute keys to `Front/Back` and `Open/Close` windows under OpenWindows 2.0?**

   Refer to the *101A Keyboard User Guide* for details on what each of the two layouts offer. The PC layout includes no `Open/Close` or `Front/Back` keys, but establishing the following resource settings in `$HOME/.Xdefaults` will enable them:

```
OpenWindows.FrontKey: F32
OpenWindows.OpenKey: F33
```

   To activate these keys, you must either restart the window system or run the following command:

```
% xrdb -load $HOME/.Xdefaults
```

7. **How can I institute keys to `Front/Back` and `Open/Close` windows under OpenWindows 3.0?**

   To activate these keys in OpenWindows 3.0, you must establish the appropriate resource settings by adding the following lines in the `$HOME/.Xdefaults` file, regardless of whether the PC layout or the Sun layout is in use:

   ```
   olwm.KeyboardCommand.OpenClosePointer: Scroll_Lock
   olwm.KeybardCommand.RaiseLower: Print
   ```

   To activate these keys, you must either restart the window system or run the following command:

   ```
   % xrdb -load $HOME/.Xdefaults
   ```

8. **How do I establish the `Copy`, `Paste`, `Find`, `Again`, and `Undo` keys in the Sun layout?**
   These keys are not mapped correctly in the OpenWindows 3.0 version of the `US101A_Sun.kt` keytable file. Another problem with this keytable is that the `Meta` key does not work. These problems are both known bugs (bug numbers 1076203 and 1077802).

   However, adding the following `xmodmap` statements into your `$HOME/.xinitrc` file will enable the `Copy`, `Paste`, `Find`, `Again`, and `Undo` keys with the Sun layout under OpenWindows 3.0:

   ```
   # Gray LeftArrow is Paste
   xmodmap -e 'keycode 31 = L8 Left SunXK_Paste'
   # Gray RightArrow is Cut
   xmodmap -e 'keycode 35 = L10 Right SunXK_Cut'
   # Gray PageDown is Copy
   xmodmap -e 'keycode 80 = L6 Next SunXK_Copy'
   # Gray UpArrow is Find
   xmodmap -e 'keycode 27 = L9 Up SunXK_Find'
   # Gray DownArrow is Again
   xmodmap -e 'keycode 34 = L2 Down SunXK_Again'
   # Gray PageUp is Undo
   xmodmap -e 'keycode 58 = L4 Prior SunXK_Undo'
   ```

   Use "`Shift-ArrowKey`" to get the cursor motion functionalities from these gray arrow keys.

9. **Under OpenWindows 3.0, the modifier map for the 101A Keyboard under the Sun layout is wrong. How can it be corrected?**

   One symptom of this error is that the `Meta` key (which is mapped to the left `Alt` key on the keyboard) does not work. (Note that this has been corrected in the next OpenWindows release, which is included in the Solaris 2.0 release.)

   Use `xmodmap` to fix the modifier map. The following command sets the modifier map correctly:

```
% xmodmap -e 'clear mod1' \
  -e 'keycode 127 = Meta_L' \
  -e 'add mod1 = Meta_L' \
  -e 'clear mod4' \
  -e 'keycode 129 = Alt_L' \
  -e 'add mod4 = Alt_L'
```

10. **Are there errors in the *101A Keyboard User Guide* of which I should be aware?**

    Yes. There are known bugs in both OpenWindows 2.0 and OpenWindows 3.0. The section "Enabling the Keyboard" on Page 26 of the *101A Keyboard User Guide* says that, in order to switch from SunView keyboard mapping to Openwindows mapping, you need to reboot the system. This is misleading and needs to be corrected.

    When switching from SunView to OpenWindows or vice versa, all you have to do is to logout and then log back in.

## References


1. *101A Keyboard User Guide* (part number 800-5794-10), revision A, April 1991

2. The OpenWindows 3.0 file:

   `$OPENWINHOME/etc/keytables/keytable.map`

3. `man` pages for `xmodmap`, `xev`, `loadkeys`, `keytables`, `olwm`, `xview`, and `psh`

4. Symptom/Resolution Data Base (SRDB) articles:

```
4215      101a keyboard arrow keys don't autorepeat under owv3
4224      General US 101A keyboard issues
```

5. Customer Bugs

| | |
|---|---|
| 1068744 | `101A keyboard does not function correctly on OpenWindows3.0` |
| 1068817 | `Stand-alone arrow keys don’t repeat for 101A keyboard` |
| 1071670 | `Misleading doc bug in 101A Keyboard User Guide` |
| 1072683 | `keyboardtype returns 4 instead of 23 for a type 10` |
| 1076203 | `Open/Close and Front/Back keys on 101A don’t work` |
| 1077802 | `Copy, Paste, Cut or Delete functions in OW3 on 101A` |
| 1077836 | `There is a need for documentation on OW3 and the 101A` |
| 1085811 | `"xset r somekeycode" and "xset -r somekeycode" broken in OW2` |

**Note –** Customer Bugs and SRDB articles may be examined via the Online Support System (OSS), more recently named *SunSolve*. If you need information on how to access SunSolve, call 1-800-USA-4-SUN.

# *Languages* 5

## *ANSI C Compliance*

As part of the move toward standardization with System V Release 4 (SVR4), Solaris 2.0 provides a C compiler that complies with the ANSI C standard X3.159-1989 (ANSI C), rather than the common Kernighan/Ritchie (“K & R”) C compiler previously supplied.

The “ANSI C standard” or “ANSI C” refers to standard C as defined by ANSI. “Sun ANSI C” is Sun’s ANSI C implementation with extensions that comply with the ANSI C standard. “Common C” refers to Sun’s implementation of common C.

To ease the migration to Solaris 2.0, the ANSI C compiler has been backported to run under SunOS 4.1.*x* as part of the unbundled C 1.1 compiler package. This package consists of two compilers: `cc`, a common C compiler, and `acc`, an ANSI C compiler with compatibility flags that allow it to accept certain common C constructs.

This article outlines the migration path from common C in SunOS 4.1.*x* to Sun ANSI C in the C 1.1 package and in Solaris 2.0. Because SunOS 4.1.*x* is not a SVID III-compliant platform, software that is compiled with the C 1.1 ANSI C compiler must also be made compliant with SVID III to run successfully under Solaris 2.0.[1] For additional information, refer to the *ANSI C Transition Guide* (part number 800-5116-10).

---

1. Future issues of the STB will contain further information on System V Compliance Definition III (SVID III).

## Overview

The migration process from common C to full compliance under the Sun ANSI C compiler takes place in the stages illustrated by Figure 5-1. These include:

- choosing the level of backward compatibility that best suits a given development schedule, and compiling the code with the appropriate option
- supplying ANSI C function prototypes
- converting to ANSI C data declarations
- converting to ANSI C function declarations
- correcting compile-time warnings, error messages, and incorrect system calls and library routines
- correcting run-time inconsistencies between common C and Sun ANSI C semantics

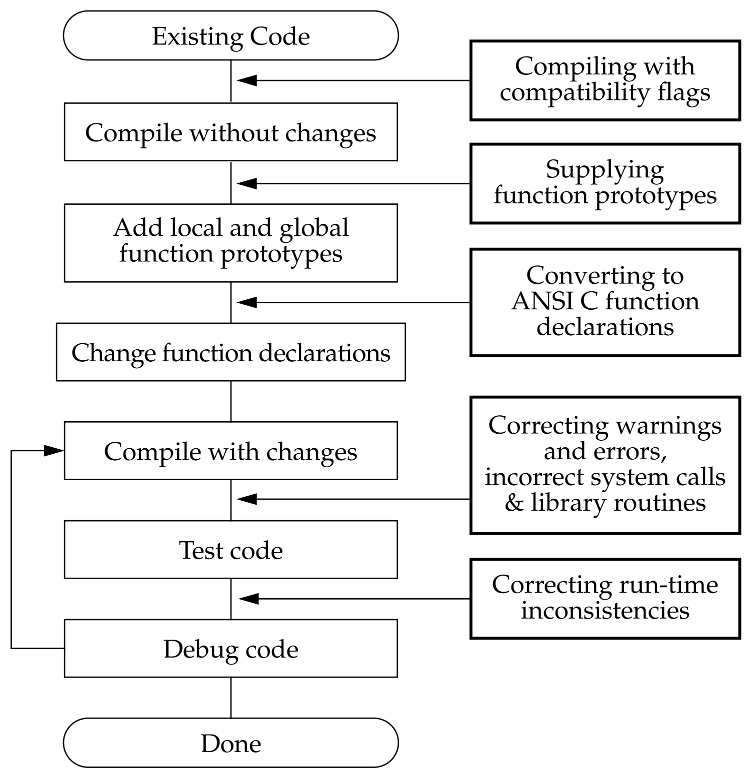


*Figure 5-1* ANSI C Porting Process

## Compiling with Compatibility Flags

The Sun ANSI C compiler provides four compatibility flags for use with code containing common C constructs:

`-Xs` This option features ANSI C along with common C (non-ANSI) extensions. The compiler warns about all language semantics that have different behaviors between Sun ANSI C and common C, and defaults to the common C semantics when possible. This flag is *not* supported by the SunOS 4.1.*x* version of the ANSI C compiler.

`-Xt` This option provides the greatest degree of compatibility with common C. Where common C and Sun ANSI C specify different semantics for the same construct, the compiler issues a warning and uses the common C interpretation.

`-Xa` This option provides an intermediate level of compatibility with common C. Where common C and Sun ANSI C specify different semantics, the compiler issues a warning and uses the ANSI C interpretation.

`-Xc` This option provides maximal conformance with the ANSI C standard. With this option, the compiler rejects programs that contain constructs not supported by Sun ANSI C.

For the general purpose options, refer to the *Sun ANSI C Programmer's Guide*.

## Supplying Function Prototypes

A *function prototype* is a function declaration or definition that follows the ANSI C prototype syntax, or a function written in that syntax. Like a traditional function declaration, a function prototype declares the data type of the value returned. Unlike a traditional declaration, a function prototype also declares the number and type of the function's formal parameters. Function prototypes are typically specified in header files or at the beginning of a source file.

There are three classes of prototypes. The class of a prototype is determined by whether the function has zero, a fixed number, or a variable number of parameters:

- A function that has no parameters must have a parameter type list consisting of the single type specifier `void`. For example:

```
extern int random_generation(void);
static void do_nothing(void) {return;}
```

- A function that has a fixed number of parameters indicates the type of those parameters in the parameter type list. If the prototype appears in a function declaration, parameter names may be included or not, as desired, which helps in documenting the function. Parameter names must appear in the definition. For example:

```
double square (double x) { return x*x; }
extern char *strncpy(char *, char *, int);
```

- A function that has a variable number of parameters or parameters of varying types indicates the types of any fixed parameters as above and follows them by the notion "`,...`" (which is composed of four tokens: a comma followed by three periods):

```
extern in fprintf (FILE *file, char *format,...);
```

  There must be at least one fixed parameter, or the parameter list cannot be accessed using the standard library facilities from `stdarg.h` .

Although mixing a prototype declaration and a non-prototype declaration for the same function is not recommended, ANSI C does specify conditions under which the two kinds of declarations are compatible.

In general, there are many different prototypes that are individually compatible with a non-prototype declaration. For instance, Table 5-1 shows some prototype declarations that would be compatible with the following non-prototype declaration:

```
extern int f();
```

*Table 5-1* Prototype Declarations Compatible with `extern` Declaration

| **Prototype** | **Compatible with** `extern int f()` **?** | **Reason** |
|---|---|---|
| `extern double f(void);` | no | The parameter list is acceptable, but the return types are not compatible |
| `extern int f(int, float);` | no | `float` is not compatible with `double` |
| `extern int f(double x);` | yes | The parameter type does not change upon conversion |
| `extern int f(float *);` | yes | The argument is a pointer that is not converted |

In common C, each argument is converted just before being passed to the called function according to the default argument promotions. These specify that all integral types narrower than `int` are promoted to `int`, and any `float` arguments are promoted to `double`. This simplifies both the compiler and libraries. ANSI C's function prototypes are more expressive—the specified parameter type is what is passed to the function. Thus, if a function prototype is written for an existing (common C) function definition, there should be no parameters in the function prototype with any of the following types:

```
char      signed char      unsigned char      float
short     signed short     unsigned short
```

In general, there is only one prototype that matches a non-prototype function definition. This prototype is sometimes referred to as the function's *Miranda* prototype, since it is "appointed" to a function definition that otherwise would not have one. Table 5-2 lists some compatible and incompatible prototype declarations for the following definition:

```
int f(x, y)
       float x;
       int y;
{ ...}
```

*Table 5-2* Prototype Compatibility with Function Definition

| **Prototype** | **Compatible with** `int f(x, y)` **?** | **Reason** |
|---|---|---|
| `extern double f(double x, int y);` | no | The parameter list is acceptable, but the return types are not compatible |
| `extern int f(float, int);` | no | The first parameter must have prototype `double` |
| `extern int f(double a, int b);` | yes | This is the only compatible prototype; the parameter names do not matter |

## Converting to ANSI C Function Declarations

Most modern programming languages such as Pascal and Ada check the agreement of the formal and actual parameter to functions; that is, both the number of arguments and the types of the individual arguments must agree. This checking is also performed in ANSI C when a function is declared with a prototype.

In the following example, the `sqrt1` function call is not governed by a prototype; therefore the C compiler is not required to warn the programmer that the actual parameter to `sqrt1` is of type `long` even though the formal parameter is declared to have type `double`. The `sqrt1` function will simply return an incorrect value.

When a call, such as `sqrt2`, is governed by a prototype in ANSI C, the actual arguments are converted to the corresponding formal type. Only if this conversion is impossible, or if the number of arguments does not agree with the number of formal arguments, will the C compiler reject the program. Therefore, by adding a prototype to the definition of `sqrt1` above, the example will work correctly: the `long` argument will be converted to the `double` transparently.

```
#include <stdio.h>
double sqrt2(double);
double sqrt1(x)
double x;
{
       return(x);
}
main()
{
       long x=10, y=10;
       printf("Because sqrt1 uses the old-style declaration, argument\n");
       printf("is not promoted to double. The return is still long.\n");
       printf("sqrt1(%d, %d) = %d (long)\n", x, y, sqrt1(x*x+y*y));
       printf("sqrt1(%d, %d) = %f (not double)\n\n", x, y, sqrt1(x*x+y*y));
       printf("Because sqrt2 uses the new-style declaration, argument\n");
       printf("is promoted to double. The return becomes double.\n");
       printf("sqrt2(%d, %d) = %f (double)\n", x, y, sqrt2(x*x+y*y));
       printf("sqrt2(%d, %d) = %d (not long)\n", x, y, sqrt2(x*x+y*y));
}
```

When run, this program produces the following results:

```
% acc -Xt proto.c
% a.out
Because sqrt1 uses the old-style declaration, argument
is not promoted to double. The return is still long.
sqrt1(10, 10) = 200 (long)
sqrt1(10, 10) = 0.000000 (not double)

Because sqrt2 uses the new-style declaration, argument
is promoted to double. The return becomes double.
sqrt2(10, 10) = 200.000000 (double)
sqrt2(10, 10) = 1080623104 (not long)
```

## Correcting Compiler Warnings and Errors

The following examples illustrate warning and error messages from the ANSI C compiler when used with the various compatibility options. For additional examples, refer to "ANSI C Notes and Code Samples" later in this article.

### Keyword Errors

The following program shows errors involving the keywords `const`, `volatile`, and `signed`.

```
#include <stdio.h>

main()
{
        int const=1, volatile=2, signed=3;
        printf("const=%d, volatile=%d, signed=%d\n",
                const, volatile, signed);
}
```

Under common C, the program compiles and runs without errors. Under ANSI C with the `-Xs`, `-Xt`, `-Xa`, and `-Xc` options, the compiler gives the following messages:

```
% acc -Xc key.c
"key.c", line 5: syntax error before or at: =
"key.c", line 7: syntax error before or at: const
Compilation failed
```

### Semantics of Unsigned Expressions

The following program shows how the semantics of unsigned expressions are preserved:

```
#include <stdio.h>

main()
{
      printf(" return = %d\n", test1());
      printf(" return = %d\n", test2());
}
int test1()
{
      int i = -2;
      unsigned char uc = 1;
      return (i+uc)<17;
}
int test2()
{
      int i = -2;
      unsigned char uc = 1;
      return (i+(int)uc)<17;
}
```

Under common C, the program compiles and runs without errors.

Under ANSI C with the `-Xs` and `-Xt` options, the compiler gives the following warning:

```
% acc -Xt unsign_preserve.c
"unsign_preserve.c", line 14: warning:
      semantics of "<" change in ANSI C;
use explicit cast
% a.out
return =0
return =1
```

Under ANSI C with the -Xa and -Xc options, the compiler gives the following warning and produces a different run-time result:

```
% acc -Xc unsign_preserve.c
"unsign_preserve.c", line 14: warning:
        semantics of "<" change in ANSI C;
use explicit cast
% a.out
return =1
return =1
```

## Correcting Run-Time Inconsistencies

In certain cases, differences between common C and ANSI C semantics do not trigger warnings or errors from the ANSI C compiler. Typical potential run-time problems include:

- character constants
- differences in unsigned expression semantics
- implementation-specific initialization of aggregates

The following example shows that the order of multi-byte character constants differs between the common C and ANSI C compilers. This type of problem can only be detected at run-time. You can use a debugging tool, such as dbxtool(1), to correct these problems.

### Character Constants

The following program illustrates the behavior of character constants.

```
#include <stdio.h>

main()
{
        int mc = 'abcd';
        printf("mc = %c %c %c %c\n" mc, mc>>8, mc>>16, mc>>24);
}
```

Under common C and ANSI C with -Xs, the program behaves as shown:

```
% cc multi_byte.c
% a.out
mc = d c b a
```

Under ANSI C with the `-Xt` flag:

```
% acc -Xt multi_byte.c
% a.out
mc = a b c d
```

Under ANSI C with the `-Xc` flag:

```
% acc -Xc multi_byte.c
% a.out
mc = a b c d
```

## ANSI C Notes and Code Samples

This section discusses a number of incompatibilities between Sun C and ANSI C.

The `-Xs` compiler option is not available in SunOS 4.1.*x*; otherwise, the compatibility flags noted in this article behave consistently between SunOS 4.1.*x* and Solaris 2.0.

### `envp` *Argument to* `main()`

Common C allows `envp` as a third argument to `main()`. Sun ANSI C also allows `envp` as a third argument to `main()`, although this usage does not strictly conform to the ANSI C standard.

### `extern` *and* `static` *Function Declarations Inside a Block*

Common C promotes extern and static function declarations within blocks to file scope. The ANSI standard does not guarantee that block scope function declarations are promoted to file scope.

### *Identifier with '*`$`*'*

Common C allows dollar signs (`$`) in identifiers, while ANSI C does not.

```
#include <stdio.h>
main()
{
       int $i;
       $i = 100;
       printf("$i = %d\n", $i);
}
```

Common C and ANSI C `-Xs`:

```
% cc iden.c
% a.out
$i = 100
```

ANSI C `-Xt`:

```
% acc -Xt iden.c
"iden.c", line 5: warning: "$" used in identifier $i
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc iden.c
"iden.c", line 5: invalid source character: '$'
"iden.c", line 7: invalid source character: '$'
"iden.c", line 8: invalid source character: '$'
Compilation failed
```

### `long float` *Type*

Common C accepts `long float` declarations and treats these as type `double`. ANSI C does not accept these declarations.

```
#include <stdio.h>
main()
{
        long float x;
        x = 100;
        printf ("x = %d\n", x);
}
```

Common C:

```
% cc long_float.c
% a.out
x = 100
```

ANSI C `-Xs`, `-Xt`, `-Xa`, `-Xc`:

```
% acc -Xc long_float.c
"long_float.c", line 5: warning:
        use "double" instead of "long float"
```

### Integer Constants

Common C accepts 8 or 9 in octal escape sequences; ANSI C does not.

```
#include <stdio.h>
main()
{
       int i;
       i = '\9';
       printf("i = %d\n", i);
}
```

Common C:

```
% cc escape.c
% a.out
i = 57
```

ANSI C -Xs, -Xt, -Xa, -Xc:

```
% acc -Xc escape.c
"escape.c", line 7: warning: dubious escape: \9
```

### Assignment Operators

Common C treats operator pairs (i.e., assignment operators such as "*=") as two tokens, and as a consequence, permits white space between them. ANSI C treats them as a single token, and therefore disallows white space between them.

```
#include <stdio.h>
main()
{
       int i=2;
       i * = 10;
       printf("i = %d\n", i);
}
```

Common C:

```
% cc operator.c
% a.out
i = 20
```

ANSI C -Xs, -Xt, -Xa, -Xc:

```
% acc -Xc operator.c
"operator.c", line 7:
      whitespace between two character assignment operators
acc: Fatal error in acomp: Bus error (core dumped)
Compilation failed
```

### *Single/*`double` *Precision Calculations*

Common C promotes the operands of floating point expressions to `double`. ANSI C allows operations on `floats` to be performed in single-precision calculations.

### *Name Space of* `struct/union` *Members*

Common C allows `struct`, `union`, and arithmetic types using member selection operators (‘`.`’, ‘`->`’) to work on members of other `structs` or `unions`. ANSI C requires that every unique `struct/union` have its own unique namespace.

### *Cast as an* `lvalue`

Common C supports casts as lvalues. ANSI C does not support this feature.

```
#include <stdio.h>
main()
{
       char ch = 'a';
       int *ptr;
       (char *)ptr = &ch;
}
```

ANSI C -Xs, -Xt, -Xa, -Xc:

```
% acc -Xc cast.c
"cast.c", line 8: warning: a cast does not yield an lvalue
```

### *Implied* `int` *Declarations*

The common C compiler supports declarations without an explicit type specifier. A declaration of `num` is treated as implied `int num`. In ANSI C, a `num` declaration (without the explicit type specifier `int`) is not supported and generates a syntax error.

### Empty Declarations

The common C compiler accepts empty declarations. The ANSI C compiler issues a warning.

```
#include <stdio.h>
main()
{
        int a=10;
        int;
        printf("a = %d\n", a);
}
```

Common C:

```
% cc empty_int.c
% a.out
a = 10
```

ANSI C -Xs, -Xt, -Xa, -Xc:

```
% acc -Xc empty_int.c
"empty_int.c", line 6: warning: useless declaration
% a.out
a = 10
```

### Type Specifiers on Type Definitions

The common C compiler allows type specifiers such as `unsigned`, `short`, and `long` on `typedef` declarations. ANSI C does not allow type specifiers to modify `typedef` declarations.

```
#include <stdio.h>
main()
{
        typedef short small;
        unsigned small x;
        x = 10;
}
```

ANSI C -Xs, -Xt:

```
% acc -Xt typedef.c
"typedef.c", line 6: warning:
        modifying typedef with "unsigned";
        only qualifiers allowed
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc typedef.c
"typedef.c", line 6: warning:
      ANSI C behavior differs;
      not modifying typedef with "unsigned"
"typedef.c", line 6: identifier redeclared: small
"typedef.c", line 6: syntax error before or at: x
"typedef.c", line 8: undefined symbol: x
```

### *Types Allowed in Bit Fields*

Common C allows bit fields of all integral types, including unnamed bit fields. ANSI C supports bit fields only of the type `int`, `unsigned int`, and `signed int`. Other types are undefined.

### *Treatment of Tags in Incomplete Declarations*

In an ANSI conforming implementation, an incomplete `struct` or `union` type specifier hides an enclosing declaration with the same tag. Common C ignores the incomplete type declaration.

```
#include <stdio.h>
struct x {
       int num1;
       int num2;
} s1;

main()
{
       struct x;
       struct y; {
              struct x f1;
              int num3;
       } s2;
       s1.num1 = 1;
       s1.num2 = 2;
       s2.f1.num1 = 3;
       s2.f1.num2 = 4;
       s2.num3 = 5;
       printf("s1.num1 = %d, s1.num2 = %d\n", s1.num1, s1.num2);
       printf("s2.f1.num1 = %d, s2.f1.num2 = %d\n, s2.f1.num1, s2.f1.num2);
       printf("s2.num3 = %d\n", s2.num3);
}
```

Common C:

```
% cc tag.c
% a.out
s1.num1 = 1, s1.num2 = 2
s2.f1.num1 = 3, s2.f1.num2 = 4
s2.num3 = 5
```

ANSI C `-Xs`, `-Xt`:

```
% acc -Xt tag.c
"tag.c", line 10: warning:
       declaration introduces new type in ANSI C: struct x
% a.out
s1.num1 = 1, s1.num2 = 2
s2.f1.num1 = 3, s2.f1.num2 = 4
s2.num3 = 5
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc tag.c
"tag.c", line 12: incomplete struct/union/enum x: f1
"tag.c", line 18: warning: improper member use: num1
"tag.c", line 19: warning: improper member use: num2
"tag.c", line 23: warning: improper member use: num1
"tag.c", line 23: warning: improper member use: num2
Compilation failed
```

### *Mismatch on* `struct/union/enum` *Declarations*

Common C allows mismatches on the `struct/union/enum` type of a tag in nested `struct/union` declarations. ANSI C will treat the inner declaration as a new declaration, hiding the outer tag.

```
#include <stdio.h>
struct x {
        int num1;
        int num2;
} s1;

main()
{
        union x s2;
        s1.num1 = 1;
        s1.num2 = 2;
        s2.num1 = 5;
        printf("s1.num1 = %d, s1.num2 = %d\n", s1.num1, s1.num2);
        printf("s2.num1 = %d, s2.num2 = %d\n", s2.num1, s2.num2);
}
```

Common C:

```
% cc mis_match.c
% a.out
s1.num1 = 1, s1.num2 = 2
s2.num1 = 5, s2.num2 = 0
```

ANSI C `-Xs`, `-Xt`:

```
% acc -Xt mis_match.c
"mis_match.c", line 10: warning:
        base type is really "struct x": s2
"mis_match.c", line 10: warning:
        declaration introduces new type in ANSI C: union x
% a.out
s1.num1 = 1, s1.num2 = 2
s2.num1 = 5, s2.num2 = 0
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc mis_match.c
"mis_match.c", line 10: incomplete struct/union/enum x: s2
"mis_match.c", line 14: warning: improper member use: num1
"mis_match.c", line 17: warning: improper member use: num1
"mis_match.c", line 17: warning: improper member use: num2
Compilation failed
```

### Labels in Expressions

Common C treats labels as (`void *`) lvalues. ANSI C does not allow labels in expressions.

### Switch Condition Type

Common C allows `float` and `double` expressions as conditional types by converting them to type `int`. ANSI C evaluates only integral types (`int`, `char`, and `enumerated`) for the switch condition.

### Syntax of Conditional Inclusion Directives

The common C preprocessor ignores trailing tokens after an `#else` or `#endif` directive. ANSI C disallows such constructs.

```
#include <stdio.h>
#define test
#ifdef test comment area
#define NUM 10
#endif comment area
main()
{
      printf("NUM = %d\n", NUM);
}
```

Common C and ANSI C `-Xs`:

```
% cc dir_tail.c
% a.out
NUM = 10
```

ANSI C `-Xt`, `-Xa`:

```
% acc -Xt dir_tail.c
"dir_tail.c", line 5: warning:
      tokens ignored at end of directive line
% a.out
NUM = 10
```

ANSI C -Xc:

```
% acc -Xc dir_tail.c
"dir_tail.c", line 5: warning:
      tokens ignored at end of directive line
"dir_tail.c", line 7: warning:
      tokens ignored at end of directive line
% a.out
NUM = 10
```

### Token Pasting and the ## Preprocessor Operator

ANSI C defines ## as the preprocessor operator that performs token pasting. Common C does not recognize the ## operator.

```
#include <stdio.h>
#define INC "++"
#define TAB "internal_table"
#define INCTAB "table_of_increments"
#define CONC(x,y) x/**/y
main()
{
      printf("%s\n", CONC(INC,TAB));
}
```

Common C and ANSI C -Xs:

```
% cc pp_token.c
% a.out
table_of_increments
```

ANSI C -Xt:

```
% acc -Xt pp_token.c
"pp_token.c", line 7: warning: comment is replaced by "##"
% a.out
table_of_increments
```

ANSI C -Xa, -Xc:

```
% acc -Xc pp_token.c
"pp_token.c", line 7: warning:
      comment does not concatenate tokens
% a.out
++internal_table
```

### Preprocessor Rescanning

The common C preprocessor will recursively substitute. In ANSI C, a macro will not be replaced if it is found in the replacement list during the rescan.

### `typedef` Names in Formal Parameter Lists

Common C allows one to use `typedef` names as formal parameter names in a function declaration, and would, in effect, hide the `typedef` declaration. ANSI C disallows this use.

```
#include <stdio.h>
typedef int yy;
main()
{
	yy num=10;
	test(num);
	test(100);
}
test(yy)
int yy;
{
	printf("yy = %d\n", yy);
}
```

Common C:

```
% cc typedef_param.c
% a.out
yy = 10
yy = 100
```

ANSI C `-Xs`, `-Xt`, `-Xa`, `-Xc`:

```
% acc -Xc typedef_param.c
"typedef_param.c", line 14:
	parameter not in identifier list: yy
"typedef_param.c", line 15:
	formal parameter lacks name: param #1
```

### Implementation-Specific Initialization of Aggregates

Common C uses a bottom-up algorithm when parsing and processing partially elided initializers within braces. ANSI C uses a top-down parsing algorithm.

```
#include <stdio.h>
struct { int a[3]; int b; } w[] = { {1}, 2};
struct { int a[3]; int b; } y[] = { 3,0,0, 4};

main()
{
	printf("w[0].a[0]=%d, w[0].a[1]=%d, w[0].a[2]=%d, w[0].b=%d\n",
		w[0].a[0], w[0].a[1], w[0].a[2], w[0].b);
	printf("w[1].a[0]=%d, w[1].a[1]=%d, w[1].a[2]=%d, w[1].b=%d\n",
		w[1].a[0], w[1].a[1], w[1].a[2], w[1].b);
	printf("y[0].a[0]=%d, y[0].a[1]=%d, y[0].a[2]=%d, y[0].b=%d\n",
		y[0].a[0], y[0].a[1], y[0].a[2], y[0].b);
	printf("y[1].a[0]=%d, y[1].a[1]=%d, y[1].a[2]=%d, y[1].b=%d\n",
		y[1].a[0], y[1].a[1], y[1].a[2], y[1].b);
}
```

Common C:

```
% cc init_agg.c
% a.out
w[0].a[0]=1, w[0].a[1]=0, w[0].a[2]=0, w[0].b=2
w[1].a[0]=3, w[1].a[1]=0, w[1].a[2]=0, w[1].b=4
y[0].a[0]=3, y[0].a[1]=0, y[0].a[2]=0, y[0].b=4
y[1].a[0]=2002464861, y[1].a[1]=778132272,
	y[1].a[2]=1564288356, y[1].b=740325211
```

ANSI C `-Xs`, `-Xt`, `-Xa`, `-Xc`:

```
% acc -Xc init_agg.c
% a.out
w[0].a[0]=1, w[0].a[1]=0, w[0].a[2]=0, w[0].b=0
w[1].a[0]=2, w[1].a[1]=0, w[1].a[2]=0, w[1].b=0
y[0].a[0]=3, y[0].a[1]=0, y[0].a[2]=0, y[0].b=4
y[1].a[0]=2002464861, y[1].a[1]=778132272,
	y[1].a[2]=1564288356, y[1].b=740325211
```

### Comments Spanning Include Files

Common C allows comments which start in an include file to be terminated by the file that includes the first file. ANSI C does not support this behavior.

In file comment.h:

```
#define number 111
/* This is the first part of comment. Comment will be terminated in main()
```

In file comment.c:

```
#include <stdio.h>
#include "comment.h"

This is the second part of comment */
main()
{
      printf("number = %d\n", number);
}
```

Common C and ANSI C -Xs:

```
% cc comment.c
% a.out
number = 111
```

ANSI C -Xt, -Xa, -Xc:

```
% acc -Xc comment.c
"./comment.h", line 3: warning: EOF in comment
"comment.c", line 5:
      syntax error, probably missing ",", ";" or "="
"comment.c", line 5: syntax error before or at: is
"comment.c", line 5:
      syntax error, probably missing ",", ";" or "="
Compilation failed
```

### Formal Parameter Substitution Within a Character Constant

The common C preprocessor substitutes a character constant when it matches the replacement list macros. In ANSI C, the character is not replaced.

```
#include <stdio.h>
#define string(c) "c"
main()
{
      printf("char = %s\n", string(testing));
}
```

Common C and ANSI C `-Xs`:

```
% cc param.c
% a.out
char = testing
```

ANSI C `-Xt`:

```
% acc -Xt param.c
"param.c", line 3: warning:
      macro replacement within a string literal
% a.out
char = testing
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc param.c
"param.c", line 3: warning:
      no macro replacement within a string literal
% a.out
char = c
```

### *Formal Parameter Substitution Within a String Constant*

The common C preprocessor will substitute a formal parameter when enclosed with a string constant. The behavior is not supported in ANSI C. For string substitution in ANSI C, the # preprocessor operator should be used.

```
#include <stdio.h>
#define string(cc) "cc"
main()
{
        printf("string = %s\n", string(test));
}
```

Common C and ANSI C `-Xs`:

```
% cc str_sub.c
% a.out
string = test
```

ANSI C `-Xt`:

```
% acc -Xt str_sub.c
"str_sub.c", line 3: warning:
        macro replacement within a string literal
% a.out
string = test
```

ANSI C `-Xa`, `-Xc`:

```
% acc -Xc str_sub.c
"str_sub.c", line 3: warning:
        no macro replacement within a string literal
% a.out
string = cc
```

## Unbundled Compilers

Sun does not plan to “bundle” any compiler with the Solaris 2.0 operating system. The C compiler, which has traditionally been included as part of the Sun operating system, will be sold as a separate unbundled product.

Sun has based its decision on the needs of its customers. Sun’s decision to unbundle the compiler was based on the following:

- The compiler is not required to build the kernel.
- A large percentage of the customer base do not use or need the compiler.
- There is a cost saving in Solaris 2.0 by eliminating the compiler.
- Third-party compiler options are available.

### Compiler not required to build kernel

New features available in the SVR4-based Solaris 2.0 operating system eliminate the need for a compiler in a large percentage of Sun’s customer base.

Solaris 2.0 does not require a compiler to build the operating system during an installation, as was required in previous Sun operating systems. In previous SunOS releases, the compiler was needed to change the BSD-derived kernel. Other functions required that you remake the kernel as well, such as adding devices to a Sun system.

In Solaris 2.0, libraries are dynamically linked. One of the goals of Solaris 2.0 is to eliminate the necessity of reconfiguring the kernel, by using loadable modules, the setting of integer variables, and the dynamic allocation and configuration of kernel data structures. This enhancement eliminates the need for a compiler to effect changes to the kernel.

---

**Note –** Although the Solaris 2.0 software does not include a bundled C compiler, the system still includes everything an application writer needs for development, such as libraries and header files.

---

Key points in the Solaris 2.0 and SVR4 enhancements are as follows:

- Solaris 2.0 uses an improved native object file format called the Executable and Linking Format (ELF) to dynamically link libraries.
- The use of shared libraries in Solaris 2.0 has changed in two ways:
  - Use of minor version numbers is no longer supported.
  - The `.sa` files for shared initialized data are no longer required.

- The dynamic loader has been enhanced to load objects in both ELF format and previous `a.out` executables running in SunOS 4.*x* Binary Compatibility mode.
- Solaris 2.0 includes new signals relating to multithreading.

### *Large percentage of customer base do not use or need the compiler*

Sun has done extensive market studies to determine the needs of Sun users. The results concluded that a large percentage of Sun's customers do not use their Sun system to perform development requiring the use of a compiler. Unbundling the compiler in Solaris 2.0 allows this high percentage of Sun users to not be forced to pay for a compiler that they may never use.

### *Cost saving in Solaris 2.0 by eliminating compiler*

Sun offers an improved ANSI C compatible compiler that will be available with Solaris 2.0, but sold separately as an unbundled product. Selling the compiler as an unbundled product allows the Solaris system to be delivered at a cost savings to the customer. The ANSI C compatible compiler will be available at a competitive cost to Sun customers. Although users should not expect any reductions in system prices as a result of the unbundling, the new system would have been more expensive if the compiler had been included in the Solaris 2.0 distribution.

### *Third-party compiler options available*

Sun users now have the opportunity to select which compiler (including those developed by third-party vendors) best fits their development needs.

Sun's software development subsidiary, SunPro, has developed a series of high-end SPARC compilers (known as "SPARCompilers") for all Sun's core languages, including Sun C 1.1 for 4.*x* platforms. Additionally, SunPro offers the option of purchasing a compiler separately, or one bundled with SPARCworks 2.0 tools, including the Debugger, SourceBrowser, Analyzer, MakeTool and FileMerge. Opting to purchase the compiler only would cost less than purchasing the compiler bundled with SPARCworks.

---

**Note –** Other development utilities, including `yacc`, `awk`, `sccs`, and `lex` will remain bundled with Solaris 2.0.

---

# Software Release Levels 6

## Current Sun Software Products and Release Levels

The following tables contain lists of Sun software products and their respective release levels, current as of May 31, 1992. These tables appear monthly in the STB for your convenience.

Please contact your sales representative if you would like to update the release level of a Sun software product you already use, or wish to purchase another product. Use the tables to determine whether your release is the most current.

Please note the following software product change(s):

a. As of the release of Solaris 2.0, the C compiler will be unbundled.

### Operating Systems

| Product Name | Current Release |
|---|---|
| SunOS (sun-4, sun-4c) | 4.1.2 |
| SunOS (sun-3, sun-3x) | 4.1.1_U1 |

### Window Systems & Desktop Utilities

| Product Name | Current Release |
|---|---|
| OpenWindows | 3.0 |
| DeskSet Environment (SunView) | 1.1 |

## Sun Server & Database Software

| Product Name | Current Release |
|---|---|
| Sun Database Excelerator (SunDBE) | 1.2 |
| SimplifySQL | 1.1 |
| Sun SHIELD ARM and ASET | 1.0 |
| Backup Copilot | 1.0 |
| Online: DiskSuite | 1.0 |
| Online: IPI Dual Port Tools | 1.0 |

## Graphics Software

| Product Name | Current Release |
|---|---|
| SunVision | 1.2 |
| XGL | 2.0 |
| SunGKS | 4.1 |
| SunPHIGS | 2.0 |
| SunPEX | 1.0 |

## Network Management

| Product Name | Current Release |
|---|---|
| SunNet Manager | 2.0 |
| SunNet License | 1.1 |

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

## Software Development

| Product Name | Current Release |
|---|---|
| Sun Ada Development Environment | 1.1 |
| Sun C | 1.1 |
| Sun C++ | 2.1 |
| Sun COBOL | 1.0 |
| Sun COBOL Runtime System | 1.0 |
| Sun COBOL Forms-2 | 1.0 |
| NetISAM | 1.0 |
| OpenWindows Developer's Guide (DevGuide) | 3.0 |
| ONC / RPC Application Toolkit | 1.0 |
| Sun FORTRAN | 1.4 |
| Sun Pascal | 2.1 |

## Other Products

| Product Name | Current Release |
|---|---|
| NeWSprint | 2.0 |
| VideoPix | 1.0 |
| PC-NFS | 4.0 |
| PC-NFS LifeLine | 2.0 |
| PC-NFS Advanced Telnet | 1.0 |
| PC-NFS Programmer's Toolkit | 2.0 |
| SunPC | 3.0 |

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

## Connectivity Products

| Product Name | Current Release |
|---|---|
| SunLink CG3270 | 7.1 |
| SunLink SNA3270 | 7.0.1 |
| SunLink BSC3270 | 6.1 |
| SunLink BSCRJE | 6.0 |
| SunLink SNA Peer-to-Peer | 7.0.1 |
| SunLink Local 3270 | 6.1.1 |
| SunLink Channel Gateway | 7.1 |
| SunLink DNI | 7.0 |
| SunLink TE320 | 7.0 |
| SunLink OSI | 7.0.1 |
| SunNet MHS | 7.0 |
| SunLink IR | 6.0 |
| SunLink X.25 | 7.0 |
| SunLink TRI/S | 2.0 |
| SunLink HSI/S | 1.1 |
| SunLink FDDI/S | 1.0 |
| Serial Parallel Controller | 1.1 |
| SunLink MCP | 6.0 |
| SunLink FDDI/DX | 2.0 |
| DOS Windows | 1.0.1 |

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

# Notes & Comments 7

## World Hotlines

Sun Customers throughout the world have service hotlines available for both software and hardware support questions. The service hotlines are shown below. If your country is not shown in the table, please phone your local Sun sales office. World hotlines are divided into those for the U.S.A., Europe, and Intercon.

### United States

| | |
|---|---|
| All | **1-800-USA-4SUN** (**1-800-872-4786**) |

### Europe

| | | |
|---|---|---|
| ***Finland*** | Sun Microsystems Oy | **+358-0-502-2700** |
| ***France*** | Sun Microsystems France SA | **+33-1-30-67-50-80** |
| ***Germany*** | Sun Microsystems GmbH | **+49-89-46008-321** |
| ***Italy*** | Sun Microsystems Italia SpA | **+39-6055228** |
| ***Netherlands*** | Sun Microsystems Nederland BV | **+33-501234** |
| ***Spain*** | Sun Microsystems Iberica SA | **+34-1-5560243** |
| ***Sweden*** | Sun Microsystems AB | **+46-8-623-9100** |
| ***Switzerland*** | Sun Microsystems (Schweiz) AG | **+41-1-825-7272** |
| | Sun Microsystems (Suisse) | **+41-22-64-4735** |
| ***United Kingdom*** | | |
| | Sun Microsystems UK Ltd | **+44-276-691974** |

### Intercon

| | | |
|---|---|---|
| ***Canada***: | <u>Quebec District</u><br>Montreal, Quebec | **(514) 744-9240**, **(800) 361-1554** |
| | <u>Ottawa District</u><br>Ottawa, Eastern Canada | **(613) 723-8112**, **(800) 465-4415** |
| | <u>Central District</u><br>Toronto, Central Ontario | **(416) 490-9490**, **(800) 263-1680** |
| | <u>Prairies District</u><br>Manitoba, Saskatchewan,<br>Alberta, Calgary, Western Canada | **(403) 262-6722**, **(800) 661-9256** |
| | <u>Edmonton District</u><br>Edmonton | **(403) 421-7264**, **(800) 661-3643** |
| | <u>B. C. District</u><br>Vancouver, British Columbia | **(604) 684-4120**, **(800) 663-0440** |
| | National CSS/Education | **(416) 477-6745** |
| ***Australia*** | Sun Australia | **(008) 02-4417** |
| ***New Zealand*** | Auckland<br>Wellington | **64-9-525-2749**<br>**64-4-499-2344** |
| ***Hong Kong*** | Sun Asia | **852-802-4188** |
| ***Japan*** | Nihon Sun | **044-819-3121** |

***Latin America***: Contact the Latin American hotline at ICON headquarters:
U.S. Office **(415) 688-9160**

Within ***Brazil***, you may also contact:
Sun do Brasil **55-11-887-9011**

***Caribbean***: Call your local distributor.

***All Other Countries***
Call **(415) 688-9372**, which is a U.S. number.

# Reporting Bugs

This article contains three sections for submitting bugs. The first section describes procedures to use within the United States and Canada. The second section describes procedures within Europe. The third section describes procedures within ICON.

## Submitting Software Bugs: United States and Canada

This section contains information on reporting bugs within the U.S., for customers holding and not holding support contracts.

Sun's United States Solution Center accepts software bug reports from Sun users via electronic mail and by phone. The method you use to submit a bug report varies with your needs.

U.S. users holding support contracts can report bugs to US Solution Center via the **(800) USA-4-SUN** phone hotline. Canadian users holding support contracts should call their local support center. The U.S. Solution Center phone hotline is the fastest way for a customer to find out if a problem is known and if a workaround exists. The status of previously-reported bugs can also be obtained in this way.

U.S. customers holding support contracts can also submit bug reports electronically to the address *sun!hotline* (or *hotline@Sun.COM*). Canadian customers need to use *hotline@Canada.Sun.COM*. This method generates a service order, and can be used when lines of code or other information difficult to relay over the phone is needed to describe the bug.

Customers who do not hold Sun software support contracts can report bugs via electronic mail to the address *sun!sunbugs* (or *sunbugs@Sun.COM*). These reports are reviewed periodically to determine proper disposition. Those reports determined to be from supported customers are forwarded to the U.S. Solution Center for handling. Reports from customers who cannot be verified as holding a support contract are reviewed by Sun's engineering and support personnel. An internal bug report is generated if the reported bug is new and verifiable.

Finally, customers not holding software support contracts may call the **(800) USA-4-SUN** phone hotline to report a problem and request support on a Time and Materials (T&M) basis. Canadian customers should call their local support center. In this case, please have a Purchase Order (PO) number for billing purposes.

### Summary: United States and Canada

For U.S. contract customers, **(800) USA-4-SUN** is the best method to report bugs. Canadian contract customers should report bugs to their local support center. The hotlines to use appear monthly in the note entitled "World Hotlines."

The electronic mail address *sun!hotline* (or *hotline@Sun.COM*) is available for U.S. customers to submit materials that are difficult to relay over the phone. Canadian customers should use *hotline@Canada.Sun.COM* .

For non-contract customers, the electronic mail address *sun!sunbugs* (or *sunbugs@Sun.COM*) is available to report bugs. To help us serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

## Submitting Software Bugs: Europe

This section contains information on reporting bugs within Europe, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

### Summary: Europe

To help European service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

### Finland

Report bugs to the Finland Answer Center at the postal address shown below.

Hotline
Sun Microsystems Oy
Sinikalliontie 12
SF-02630 Espoo
Finland

Hotline Telephone: +358-0-502-2700
Telefax: +358-0-502-2800

E-mail Address: *Hotline@Finland.Sun.COM*

Special Dispatch Arrangements:
Provide Dispatch with the Contract number.

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### France

Report bugs to the France Answer Center at the postal address shown below.

Service Hot Line
Sun Microsystems France SA
13, Avenue Morane Saulnier
BP 62
78142 Velizy
France

Hotline Telephone: +33-1-30-67-50-80
Telefax: +33-1-30-67-53-70
Minitel: 36 14 code SUNFRANCE

Special Dispatch Arrangements:
Please provide Dispatch with System serial number *or* Contract number.

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis, and order this support at the above address.

### Germany

Report bugs to the Germany Answer Center at the postal address shown below.

Hotline
Sun Microsystems
Bretonischer Ring 3
8011 Grasbrunn
Germany

Hotline Telephone: +49-89-46008-321
Telex: 5218197 sun
Telefax: +49-89-46008-320

E-mail Address: {*sunuk,unido*}*!sunde!hotline* or *hotline@Germany.Sun.COM*

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### Italy

Report bugs to the Italy Answer Center at the postal address shown below.

Hotline
Sun Microsystems Italia SpA
Centro Direzionale Colleoni
Palazzo Andromeda (Ingresso 1)
Via Paracelso 16
20041 Agrate Brianza
Milan
Italy

Telephone (Milan Answer Center): +039-6055228
Telex: 326423 MBOX SUNITA
Telefax: +039-6056764

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### *The Netherlands*

Report bugs to The Netherlands Answer Center at the postal address shown below.

Sun Microsystems Nederland BV
Computerweg 1
3821 AA Amersfoort
Netherlands

Hotline Telephone: +33-501234
Telefax: +33-553058
Telex: 79297 sun nl

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### *Spain*

Report bugs to Spain Answer Center at the postal address shown below.

Sun Microsystems Iberica SA
Plaza Pablo Ruiz Picasso s/n
Torre Picasso Planta 25
28020 Madrid
Spain

Hotline Telephone: +34-1-5560243
Telefax: +34-1-5564097

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### Sweden

Report bugs to the Sweden Answer Center at the postal address shown below.

Sun Microsystems AB
Box 51
S-164 94
Kista
Sweden

Hotline Telephone: +46-8-623-9100

E-mail Address: *hotline@Sun.se* or *hotline@Sweden.Sun.COM*

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### Switzerland

Report bugs to the Switzerland Answer Center at one of the postal addresses shown below.

(*German*)
Sun Microsystems (Schweiz) AG
SAPostfach
Eschenstrasse 8
CH-8603 Schwerzenbach
Switzerland

Telephone: +41-1-825-7272
Telefax: +41-1-825-7222

E-mail: *hotline@Swiss.Sun.COM*

(*French*)
Sun Microsystems (Suisse)
Route des Avouillons 12
CH-1196 Gland
Switzerland

Telephone: +41-22-64-4735
Telefax: +41-22-64-1176

E-mail: *answerline@Swiss.Sun.COM*

Special Dispatch Arrangements:
Provide Dispatch with the Contract number and System serial number.

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

***United Kingdom***

Report bugs to the UK Answer Center at the postal address shown below.

Sun Microsystems UK Ltd
Watchmoor Park
Riverside Way
Camberley
Surrey
GU15 3YL
England

Hotline Telephone: 0276-691974
Telefax: 0276-681546

Special Dispatch Arrangements:
Please provide Dispatch with either the System serial number *or* Contract number.

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

## *Submitting Software Bugs: Intercon*

This section contains information on reporting bugs within Intercon, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

## Summary: Intercon

To help Intercon service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Detailed information for individual countries follows.

### Australia

Report bugs to the Australian Answer Center at the postal address shown below.

Hotline
Sun Microsystems Australia Pty. Ltd.
PO Box 320
Artarmon, New South Wales 2064
Australia

Hotline Telephone: (008) 02-4417
Telefax: 02 436 1084

E-mail Address: *hotline@Aus.Sun.COM*

Special Dispatch Arrangements:
Please provide Dispatch with either the System serial number *or* Contract number

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### Japan

Report bugs to the Nihon Sun Answer Center at the postal address shown below.

Nihon Sun Microsystems K.K
KSP Technical Center - Answer Center
KSP R&D Bldg. B-1F-112
Sakato 100-1, Takatsu-ku,
Kawasaki, Kanagawa, 213
Japan

Hotline Telephone: 044-819-3121, 044-819-3111
FAX: 044-819-3120

Special Dispatch Arrangements:
Provide Dispatch with the Contract number and Serial Number

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

### New Zealand

Report bugs to the New Zealand Answer Center at the number shown below.

Hotline Telephone: 4-499-2344

# Your STB Subscription

Your company's software service contract includes a monthly issue of the STB. Each month, the copy of your STB is mailed to your company's primary contact person or department named in the contract. Sites with more than one contract may receive more than one STB copy, depending on how the contracts are arranged.

## *Duplicating Your STB*

Your primary contact person or department may duplicate this "master" STB copy for all Sun workstation end-users. As long as you duplicate copies and route them internally only, there are no copyright infringement problems.

This limited permission for duplication is for your convenience only, however, and does not include any duplication for resale, for distribution outside your company, or for distribution to employees of companies not having a Sun software service contract.

## *Multiple STB Subscriptions*

You can now order extra copies of the STB for a low monthly fee for each additional copy. These extra copies will sent to the same address as your "master" STB copy. Please contact the e-mail addresses shown below, or contact your sales representative for additional monthly STB copies.

## *How to Order Additional STB Subscriptions*

If you have any questions regarding your STB subscription or would like to order an additional subscription, please contact your local Sales/Service Office or Sun Authorized Distributor.

## *Further STB Questions*

If you have any questions or comments regarding the STB, including corrections, please send your comments to *stb-editor@Sun.COM* or *sun!stb-editor*.

# Index


## Numerics

101A keyboard, 4-1 to 4-10

## B

Backup Copilot, 3-1 to 3-14
- catalog, 3-7, 3-8, 3-9, 3-13
- configuration file, 3-7, 3-7 to 3-11, 3-12
- configurator, 3-7, 3-8, 3-12
- `recover`, 3-7, 3-13
- sequencer, 3-7, 3-7 to 3-11

## C

C, 5-1 to 5-26
- ANSI C, 5-1 to 5-24, 5-26
  - compatibility flags, 5-3
  - compiler errors, 5-7 to 5-9, 5-10 to 5-24
  - function prototypes, 5-3 to 5-6
- dynamic linking, 5-25

## D

DOS Windows, 4-1

`dump`, 3-2 to 3-14

## K

kernel semaphore, 2-3

## L

lightweight process (lwp), 2-5

## M

Mbus, 2-1

multiprocessing, 2-1 to 2-7
- asymmetric, 2-1
- cross calls, 2-6
- interprocessor communication, 2-6
- `ldstub`, 2-3
- locks, 2-2, 2-5
- massively parallel systems, 2-2
- multitasking, 2-2
- mutual exclusion (mutex) lock, 2-3
- parallel processing, 2-2
- `PerCPU` structure, 2-5 to 2-6, 2-7
- PROM mailbox, 2-6
- spin-lock, 2-3, 2-4
- symmetric, 2-1
- traps, 2-4 to 2-5

## N

NeWS, 4-3 to 4-4

## O

OpenWindows, 4-1, 4-3, 4-3 to 4-9

## P

PostScript/NeWS, 4-4

## R

`restore`, 3-1, 3-2, 3-4 to 3-5, 3-7, 3-13, 3-14

## S

`setuid`, 3-13
Solaris, 3-2, 4-9, 5-1, 5-25 to 5-26
SPARC, 3-14
SPARC Reference Memory Management Unit (SRMMU), 2-1
SPARCompilers, 5-26
SPARCserver 600MP, 2-1 to 2-7
SPARCworks, 5-26
STREAMS, 2-5
SunView, 4-1, 4-2 to 4-3
System V Compliance Definition (SVID III), 5-1
System V Release 4 (SVR4), 5-1, 5-25

## X

`xev`, 4-7
`xset`, 4-7