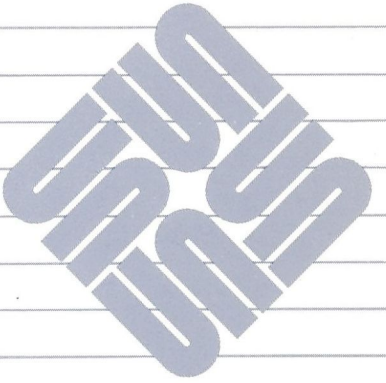
sun® microsystems

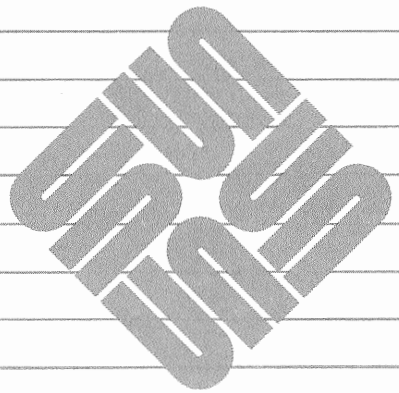
Sun™ System Introduction



Part Number: 800-1702-10
Revision A, of 9 May 1988

sun®
microsystems

Sun™ System Introduction



Part Number: 800-1702-10
Revision A, of 9 May 1988

Contents

Tables

Figures

Preface

This document presents an overview of Sun Microsystems' hardware product family, operating system software, application oriented software, and technical documentation. This book is a guide; it tells you *what is available* and *where to find* more information. It is *not* a tutorial or a ''how to'' guide for any of the software.

Who Should Read This Book

You should read this book if:

- you are considering the purchase of a Sun workstation(s) and want to know what is available;
- you already have purchased a Sun workstation, or your boss dropped one on your desk and you don't know where to start; and,
- you are new to the SunOS operating system or any of the Sun software.

Contents of Chapters

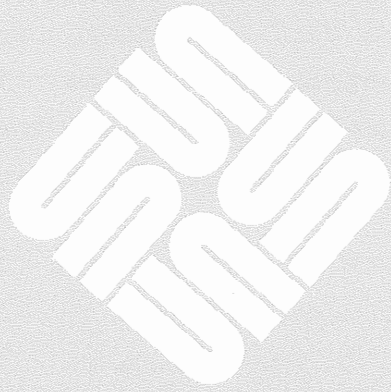
The chapters in this manual include:

- Chapter 2, *Sun Product Family*, presents an overview of the workstation systems.
- Chapter 3, *SunOS — User Features*, describes the SunOS, and discusses the major features of the SunOS operating system.
- 4, *Sunview — A Window Environment*, describes the user and programming interfaces of the SunView window environment.
- 5, *Using the Shells*, discusses the *shells*. The shell is the principal command interface for SunOS system utilities.
- Chapter 6, *User Access and Commands*, describes how to gain access to the system and its application software.
- Chapter 7, *Working with Files*, discusses the facilities for working with files and for using the hierarchical file system.
- Chapter 8, *Editing Text Files*, describes the programs used for creating, editing, and manipulating text.
- Chapter 9, *Formatting Documents*, covers the application software for producing documents.

- Chapter 10, *Communication Facilities*, describes the local area network facilities and the remote communication facilities.
- Chapter 11, *SunOS — Internal Features*, describes the programming interfaces to the SunOS operating system.
- Chapter 12, *System Administration*, describes the tools for the system administrator.
- Chapter 13, *Software Development*, describes the programming languages and language tools that are standard in the SunOS system, and discusses programming tools for debugging and maintaining software.
- Chapter 14, *Graphics Software*, describes an overview of Sun's graphic libraries.
- Chapter 15, *Unbundled and Third Party Software*, contains pointers to available unbundled products, the *Catalyst* third-party software program and the Sun Users' Group.
- Chapter 16, *Sun Technical Documentation*, describes the organization of the Sun suite of technical manuals.

1

Introduction



1

Introduction

The Sun workstation is a high-performance, bitmapped workstation complete with a monochrome or color screen display, a keyboard, and mouse pointing device. The workstation is oriented for applications in engineering, Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Engineering (CAE), Computer Aided Publishing (CAP), graphics, software development, etc.

There are three families of Sun workstations: the Sun-4 series, the Sun-3 series and the Sun386i. Each workstation in these three families is described in this manual. All three families of workstations run the SunOS operating system.

1.1. Basic Hardware

The Sun-4 family of workstations are built around the Scalable Processor ARChitecture (SPARC). SPARC is a RISC architecture; a style of computer architecture designed for simplicity and efficiency.

The Sun386i family uses an AT bus and a private 32-bit interconnect.

The Sun-3 family of workstations is built around the Motorola MC68020 processors and the high-speed 32-bit VMEbus. The monochrome and color displays have 1152 by 900 pixels. The display screen has its own memory — the *frame buffer* providing high-speed data transfer to the screen.

1.2. Basic Software

The SunOS operating system is a heavily enhanced version of the 4.2 BSD and 4.3BSD UNIX system derived from the University of California at Berkeley. It also includes features of AT&T, System V.3 UNIX, coupled with other Sun enhancements. The SunOS operating system provides:

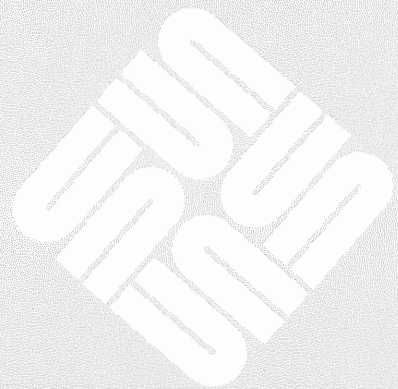
- a host of software as a base level package;
- support for C and assembler languages;
- tools for software development applications;
- utilities for performing statistical text processing and for document preparation.
- a virtual memory management scheme;
- a shared library facility; and,
- support for interprocess communication and local area networking.

To these basics, Sun Microsystems has added a *user interface* package. This package is based on overlapping windows, and a comprehensive library of graphics packages to support the more common graphics standards.

The remainder of this book introduces the Sun family of workstations and features/applications of the SunOS operating system.

2

Sun Product Family



2

Sun Product Family

The Sun family of workstations are built around industry standard chips, bus architectures, and network architectures.

- The Sun-4 family of supercomputing workstations use Sun's implementation of RISC technology, the Scalable Processor ARChitecture (SPARC).
- The Sun386i family uses an AT bus for user-expandable I/O, and a private 32-bit interconnect.
- The Sun-3 family of workstations are based on the MC680x0 processors and the high-speed 32-bit VMEbus.

The Sun-4 and Sun-3 families of workstations use the Sun Operating System (SunOS), a heavily enhanced version of 4.2BSD and 4.3BSD UNIX. The Sun386i uses an extended version of the SunOS operating system.

An earlier family of Sun workstations, the Sun-2 family, is still supported, but unavailable for purchase. The Sun-2 family is based on the MC680x0 processors, and can be upgraded to the new SunOS system.

The Sun-4, Sun386i, and the Sun-3 families of workstations include monochrome and color workstations with a wide choice of peripheral equipment.

Numerous configurations are possible in the Sun-4, Sun386i, and Sun-3 families, such as standalone workstations with disk and tape, servers on a network, and diskfull or diskless workstations on a network. The Sun-4 and Sun-3 families also include rack-mountable configurations.

2.1. Sun-4 Product Family

The Sun-4 family of supercomputing workstations use Sun's RISC technology SPARC. SPARC is source-code compatible with Sun's current Sun-3 family of workstations. SPARC architecture provides mainframe-equivalent power (10-MIPS) to the Sun-4 workstations.

Sun-4/200 Series

The Sun4/200 series offers workstations with high-resolution pixel density screens (Sun 4/260HM), a workstation with a monitor which can simultaneously present 256 colors from a palette (Sun 4/260C), and a workstation with specially enhanced graphics capabilities (Sun4/260CXP).

These workstations (and others in the 4/200 series) can be configured in numerous ways from standalone workstations, to network-servers with up to 2.3 gbytes of mass storage.

Sun-4/110 System

The Sun-4/110 is a supercomputing desktop machine also based on SPARC architecture. It is a 7-MIPS machine, for applications such as artificial intelligence and computer-aided software engineering. The Sun-4/110 can use software running on other Sun-4 machines without change, and with some modification, can use software running on Sun-3 and Sun-2 workstations.

2.2. Sun386i Product Family

The Sun386i workstations, based on Intel's 80386 microporcessor, run the SunOS operating system and are source-compatible with the Sun-4 and Sun-3 product lines. In addition, the Sun386i systems feature integrated DOS functionality, and an AT-compatible bus. Ease-of-use features built into the Sun386i system include:

- factory-installed system software;
- automatic network installation;
- context-sensitive help; and,
- window-based system and network administration.

Each Sun386i workstation includes a SCSI disk, 1.44 MB diskette, Ethernet, and an Intel 80387 floating point processor. Sun386i systems can be configured as diskless, diskfull, or server workstations.

Sun386i/150

The Sun386i/150 workstation contains 4 MB of memory, expandable to 8, 12, or 16 MB, and runs at 20 MHz.

Sun386i/250

The Sun386i/250 workstation contains 8 MB of memory, expandable to 12 or 16 MB, and runs at 25 MHz.

2.3. Sun-3 Product Family

The Sun-3 product family, like the Sun-4 family, run the SunOS operating system. The Sun-3 product family, however, is based on the Motorola MC68020 chip and the high-speed 32-bit VMEbus.

Sun-3/60 Series

The Sun-3/60 can be used as a general-purpose workstation or server. It is a 3-MIPS machine with a range of monitor options. This workstation is used for CAD/CAM, AI delivery and computer-aided publishing.

Sun-3/200 Series

This series of Sun workstations are 4-MIPS machines and are available with color, grayscale or the same high-resolution monochrome screen of the Sun-4 family. The Sun-3/200 series are used for artificial-intelligence development, electrical and mechanical CAD, and rapid software prototyping.

Sun-3/100 Series

The Sun-3/100 series is a midrange line of workstations and servers. These machines provide 2-MIPS of performance and offers a wide array of solutions for technical professionals.

Sun-3/50 Series

The Sun-3/50 runs at 1.5 MIPS and can be configured as a machine for a network, or as a standalone workstation. This machine is valuable for applications in software engineering, computer-aided design, and computer-aided publishing.

2.4. Sun-3 Optional Hardware

Optional hardware for the Sun-3 family includes a floating point accelerator. The floating point accelerator conforms to IEEE754 standards for floating point calculations and is useful for engineering applications, MCAD, ECAD, architectural computer-aided design, and scientific research.

There are also workstations which enhance graphics processing. The Sun-3/160CXP and the Sun-3/260CXP improve performance for 2-D and 3-D transformations, clipping, scaling and rendering.

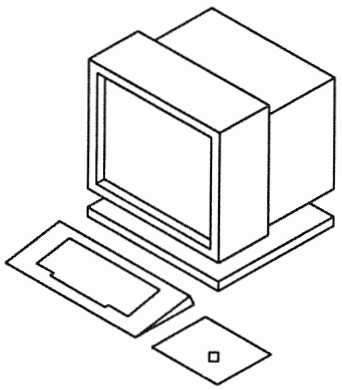
2.5. Pictures of Sun Machines

All Sun workstations and servers share the following standard features:

- a keyboard and mouse pointing device (optional for all servers);
- ethernet 10 Mbits/sec local area network interface;
- two RS-423 serial ports with modem control; and,
- 8-bit color on all color systems.

The pictures below shows desk top and desk side Sun machines.

Desk Top Systems



The desk top Sun workstation can be assembled in two different ways.

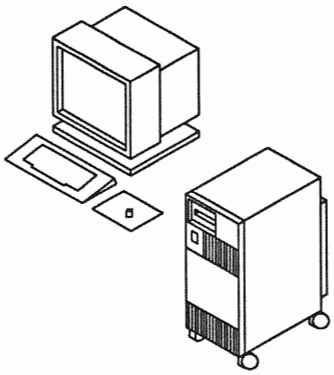
- The first type of desk top Sun is a diskless machine serviced by the network for files, data, etc.
- The second type of desk top Sun is a workstation with a local disk. This type of machine can act as a standalone or can also sit on a network. Generally, a workstation with a local disk is a three slot machine containing memory and hardware.

Sun machines which can be configured as desk top systems include:

- in the Sun-4 family of workstations, the Sun-4/110M, Sun-4/110FC, and the Sun-4/110C;
- in the Sun386i family of workstations, the Sun386i/150 and the Sun386i/250.
- in the Sun-3 family of workstations, the Sun-3/50, Sun-3/60 series, and the Sun-3/100 series, except the Sun-3/140S to Sun-3/180S systems.

Desk Side Systems

The desk side Sun system has faster and more powerful processors. It typically is configured with a disk and six or twelve slots for adding more memory or hardware.

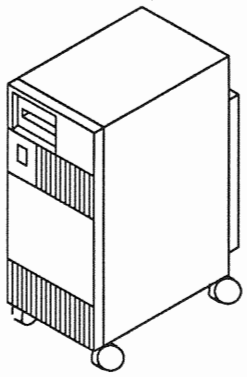


Sun systems which can be configured as desk side systems, include:

- in the Sun-4 family of workstations include the Sun-4/260HM, Sun-4/260C, Sun-4/260CXP, and the Sun-4/260G;
- in the Sun-3 family of workstations include the some workstations from the Sun-3/100 series, and workstations from the Sun-3/200 series except the Sun-3/260S and the Sun-2/80S.

File Servers

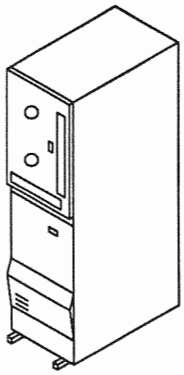
A file server is used for exporting files. Generally, it is configured with a disk and acts as a file server for an individual or a small number of people.



Systems used as file servers include:

- in the Sun-4 family of workstations, the Sun-4/110S and the Sun-4/260S.
- in the Sun-3 family of workstations, the Sun-3/140 to Sun-3/150S, and the Sun-3/260S.

Rack-Mountable File Servers



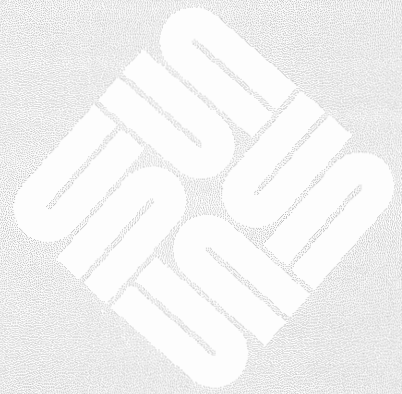
The rack-mountable file server supports high volumes of disks. It is not a workstation, but a system for servicing network file service (NFS) requests and up to about ten diskless workstations.

Systems used as file servers, include:

- in the Sun-4 family of workstations, the Sun-4/280S;
- in the Sun-3 family of workstations, the Sun-3/180S.

3

SunOS — User Features



3

SunOS — User Features

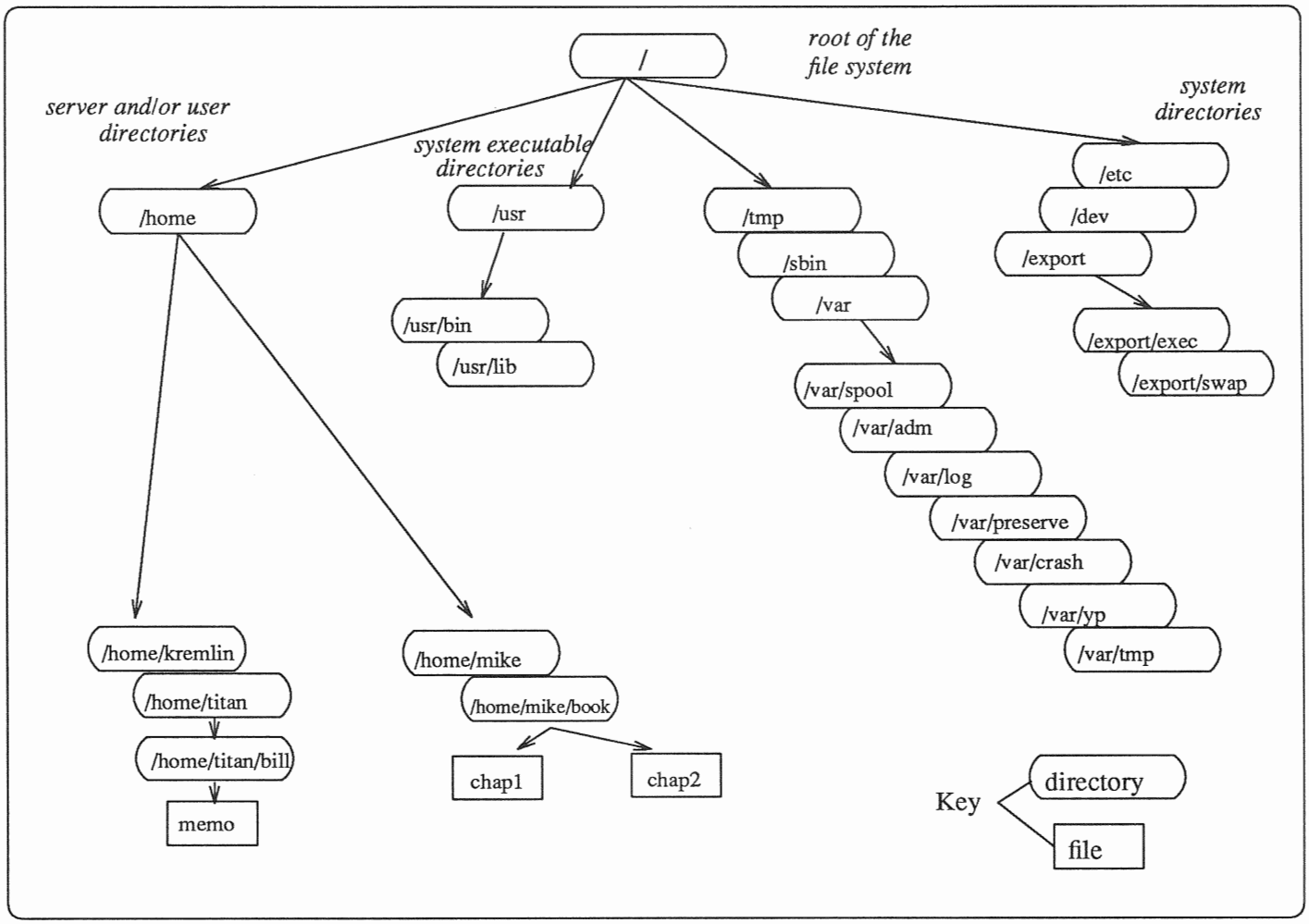
The SunOS operating system is based upon the UNIX operating system as derived from the University of California, Berkeley 4.2 and 4.3 systems. It also includes features of AT&T System V.3, and other Sun enhancements. The SunOS operating system is a virtual memory system. This chapter presents the basic user features of the system, including the hierarchical file system, the hierarchical file system in the network, and the user interface features — the SunView Window system, and the C shell command interpreter.

3.1. Hierarchical File System

The hierarchical file system is used to store and to retrieve data. It is the most visible aspect of the SunOS operating system. At the user (command) level, the basic object which you manipulate is a *file*. A file is a collection of data, such as words, characters, numbers, and attributes that are stored together. The most important attribute is the file's *name*.

The hierarchical *file system* groups related files within a *directory*. A directory is, itself, simply a special file. A directory can contain other directories as well as files and can be grouped within directories to arbitrary limits in the SunOS operating system. The figure below represents a file system containing directories and files for both the system and several users:

Figure 3-1 *Hierarchical File System*



The SunOS hierarchical file system differs from most UNIX systems. The SunOS filesystem makes administration for disk and diskless machines easier since the file system is consistent for disk/diskless machines and server/non-server machines.

The following is a brief description of the system directories in the hierarchical file system.

- The *root (/)* directory is a small directory, with files that do not change often.
- The *export* directory contains files of all exported files for diskless clients.
- The *etc* directory contains database files pertinent to each machine.
- The *sbin* directory contains files used for booting the system.
- The *var* directory contains files that change often. This directory was created to control files that grow.
- The *home* directory includes server and user directories.

For more information see *Getting Started with SunOS: Beginner's Guide* and the `hier` and `filesystem` manual pages.

3.2. The Hierarchical File System in the Network

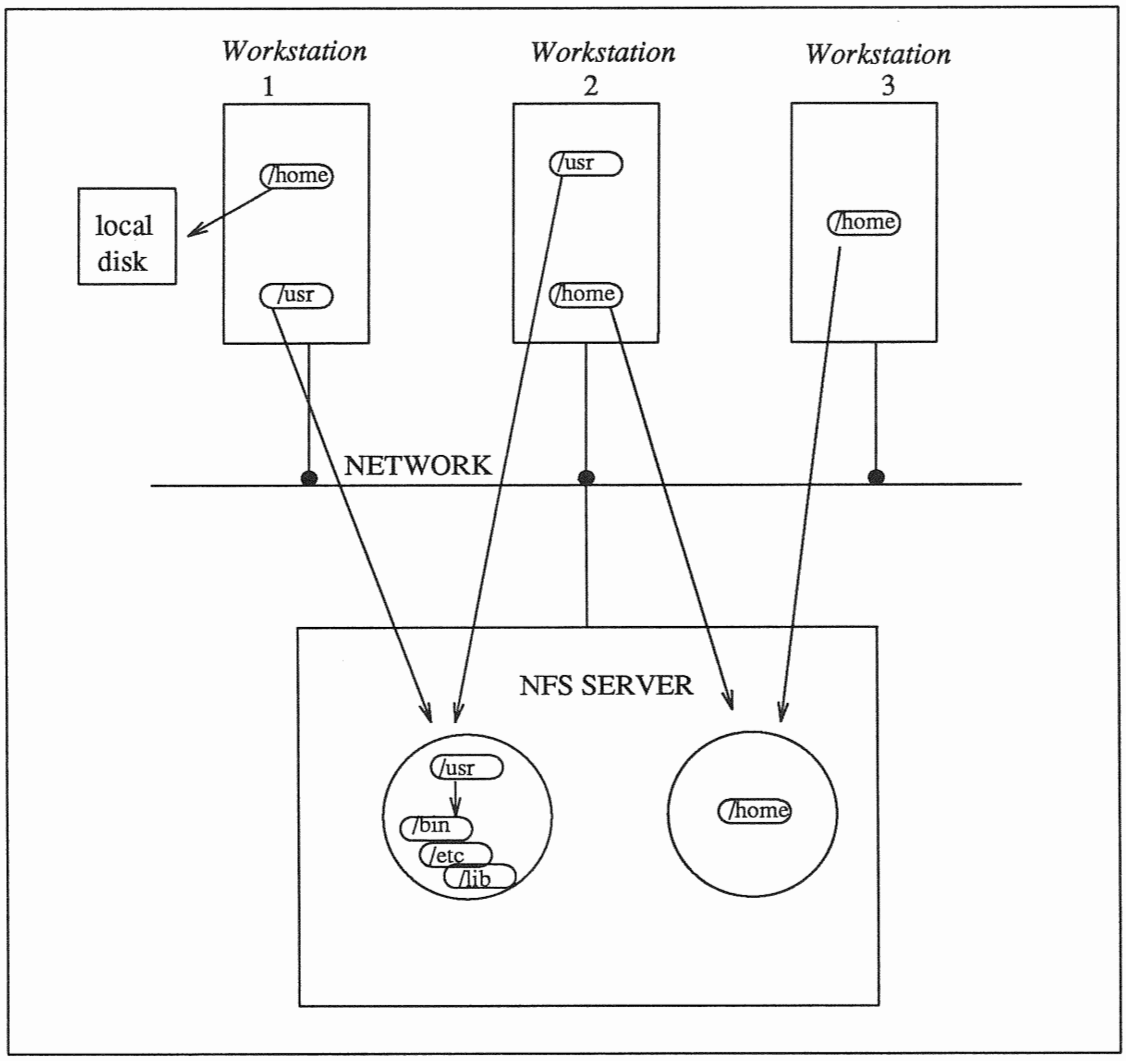
The SunOS operating system extends the file system concept a major step further with the *Network File System.* The Network File System, (NFS), is a facility for transparently sharing files in a heterogeneous environment of machines, operating systems, and networks. Sharing occurs by mounting a remote filesystem, then reading or writing files in place. The network file system avoids duplicating resources since each machine in a network does not maintain its own file systems.

In the network file system, a file system does not need to be physically present on a local disk or on your own areas of your server. File systems are mounted from *other machines* in the network, as well as from disks attached to the local machine. Once mounted, a remote file system appears as part of the existing file system hierarchy.

In the network, a machine that provides resources to the network is called an NFS *server*, while a machine that employs these resources is called an NFS *client*. A machine may be both a server and a client. A user logged in on a client machine is a *user*, while a program or set of programs running on a client machine is an *application*.

The figure below shows three workstations with their public `/usr` file systems mounted from a NFS server. All file transfers take place over the underlying network.

Figure 3-2 *Network File System*



Yellow Pages Network Service

For machines on a network there is certain information such as password, group, network, host information, etc. that should be the same for each machine. The Yellow Pages (YP) is a replicated, read-only, database service, used to administer this information. The Yellow Pages replicates this information on multiple machines, making the information highly available for server and client machines. If a server crashes, a client machine can find the information on another server.

For more information on the file system, Network File System, and the Yellow Pages, see *Using the Network: Beginner's Guide*, and *System Services Overview*.

3.3. User Interface — Window System

The SunOS operating system *user interface* uses a combination window system and command interface for doing work on the system. The SunView window system is based on multiple overlapping windows which provides a 'desktop' environment. Instead of typing commands, objects are manipulated using a pointing device called a mouse.

The SunView facility is described in further detail in chapter 3, *SunView — A Window Environment*. For further information see *SunView 1 Beginner's Guide* and *SunView 1 Programmer's Guide*.

3.4. Command Interface — C Shell

The command interface uses the C shell command interpreter. The shell is the principal *command-oriented* interface to the operating system and utilities outside of the window- and mouse-based user interface. The shell reads and interprets user commands, passes the interpreted commands to the appropriate utility software, and waits for the command to finish before accepting another command.

The C shell was implemented by Bill Joy at the University of California, Berkeley. On Sun systems the C shell is normally used for interactive use. Another standard shell available on the SunOS is the Bourne shell named after its creator, S. R. Bourne.

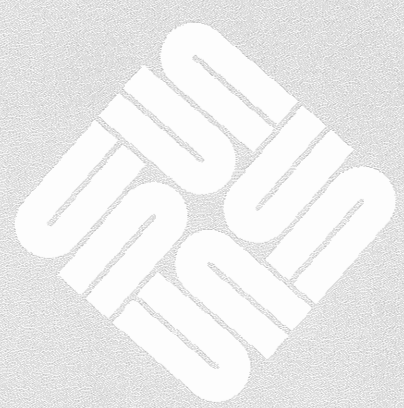
The C shell is a powerful interactive command interpreter. Major features of the C shell are listed below.

- Background and foreground running of jobs.
- Displaying the status of jobs.
- Maintaining a *history* of user commands both during a login session and across login sessions. A history list can be displayed and previous commands can be accessed either by event number or by the first few unique characters of the command. Substitutions can be made to repeated commands.
- Tailoring command sets using the C shell *alias* facility — a macro-like facility providing synonyms for commands.
- Tailoring the command environment using the C shell *startup* facilities for setting standard features of the environment such as the path the shell follows for locating commands.
- The C shell can be used as a programming language. Sequences of commands are placed in a file and the command sequences is executed just by typing the name of the file. The programming features of the shell can perform often complex tasks without creating new software.

For further information see chapter 5, *Using the Shells*, *Getting Started with SunOS: Beginner's Guide*, *Doing More with SunOS: Beginner's Guide*, and the `csh` manual page.

4

SunView — A Window Environment



4

SunView — A Window Environment

SunView — the Sun Visual/Integrated Environment for workstations is a window-based user environment which uses the capabilities of a high-resolution bit-mapped screen.

SunView has two interfaces; a user interface and a programming interface.

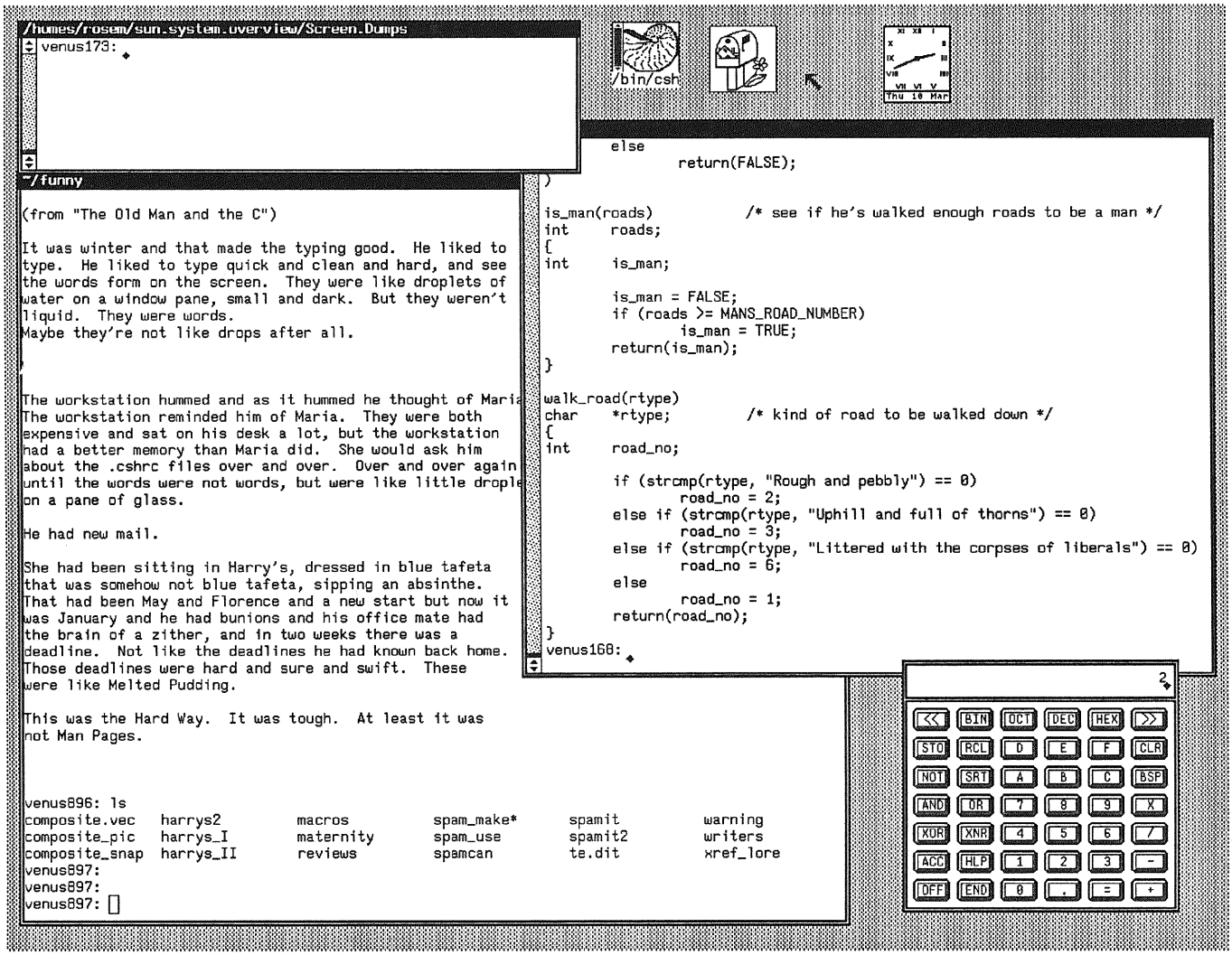
- The user interface is an environment of multiple, overlapping windows which appear on the workstation screen. The window environment is created with the `sunview` command. You can run a variety of window-based tools in SunView, and some of these tools let you run standard SunOS utilities. Each tool is a separate, independent application, and each window runs user tasks independent of the other windows on the screen.
- The programming interface is accessible through a collection of subroutine libraries. There are three major levels in the programming interface to SunView, and a programmer can use any or all of these three layers to write applications.

4.1. Window User Interface

The user interface of SunView is a collection of software utilities that use the graphical capabilities of the Sun hardware. The *mouse* (a pointing device) is a primary mechanism for interacting with the tools. Application software displays control panels with buttons that you 'click-on' with the mouse, and sliders (potentiometers) that you adjust with the mouse.

Windows occupy sections of the screen and can overlap each other. You can move windows around on the screen. You can move a window to the top of the pile by pointing at it, or you can move a window to the bottom of the pile. You can adjust the size of a window depending upon your needs. You can 'close' a window so that it occupies a minimum amount of the real-estate on the screen. When a window is closed it becomes an 'icon' — a small graphical object whose appearance sometimes suggests its function. The following picture depicts a typical screen with overlapping windows and some icons.

Figure 4-1 *Typical Screen with Overlapping Windows and Icons*



You can manipulate text in a window by using the mouse and clicking-on buttons or menu items. One application is to select text in one window and 'paste' it into the input stream of another window — this forms the basis of 'cut and paste' style editing.

For more information on using SunView and its applications see *SunView 1 Beginner's Guide.*

4.2. SunView Tools

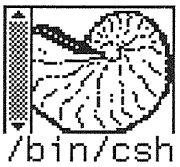
A major part of the SunView environment is the tools available for numerous applications. A *tool* is a shell, editor, or other application that runs in SunView. Some of the available tools of SunView include:

- command tool (`cmdtool`),
- shell tool (`shelltool`),

- text editor (`textedit`),
- defaults editor (`defaultsedit`),
- icon editor (`iconedit`),
- front editor (`fontedit`),
- mail tool (`mailtool`),
- Dbx (debug) editor (`dbxtool`),
- performance meter (`perfmeter`),
- clock (`clocktool`),
- lock screen (`lockscreen`),

Each of these tools is described below. A sample tool frame and tool icon representation accompany each description.

Command Tool



Command Tool (`cmdtool`) is a command interpreter that displays a standard SunView text-based command window with scrolling. It is an enhancement designed to replace Shell Tool. Command Tool can do everything that Shell Tool does, but it offers scrolling and a full Text menu.

An example of a Command Tool frame follows.

Figure 4-2 *Command Tool Frame*

```
cmdtool - /bin/csh
venus% ls *.doc
patrick_henry.doc       sched.doc
venus% head -3 patrick_henry.doc
I don't know why they always misquote me about liberty.
What they say I said ain't what I said at all.
Here are some of the remarks I've made recently on the subject.
venus% fgrep -n "Give me liberty" *.doc
patrick_henry.doc:4:Give me liberty or I'll give you what-for.
patrick_henry.doc:5:Give me liberty or I'll give you a piece of my mind.
patrick_henry.doc:7:Give me liberty or give me a salami on rye.
patrick_henry.doc:8:Give me liberty or give me a raise.
patrick_henry.doc:9:Give me liberty or give me a free lunch.
patrick_henry.doc:10:Give me liberty or give me a break.
patrick_henry.doc:11:What I want for my birthday is:  Give me liberty.
patrick_henry.doc:12:You know what I always say--Give me liberty!
patrick_henry.doc:13:Give me liberty or I'll go on strike.
patrick_henry.doc:14:"Give me liberty" seems to lose its meaning after a while.
patrick_henry.doc:15:Maybe the boy who cried wolf really cried, "Give me liberty
!"
venus% cp patrick_henry.doc liberty.doc
venus% ls *.doc
liberty.doc             patrick_henry.doc       sched.doc
venus% get_selection | cat
If liberty is all it's cracked up to be, gimme some.
Give me more
and more
and more
and more
and more
and more
venus%
```

For more information on Command Tool see *SunView 1 Beginner's Guide* and the `cmdtool` manual page.

Shell Tool

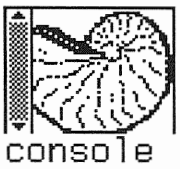
Shell Tool (`shelltool`) is a a window-based interface to the standard SunOS operating system command interpreters (shells). The following is an example of a Shell Tool frame.

Figure 4-3 *Shell Tool Frame*

```
shelltool - /bin/csh
venus% ls *.doc
liberty.doc             patrick_henry.doc       sched.doc
venus% head -3 patrick_henry.doc
I don't know why they always misquote me about liberty.
What they say I said ain't what I said at all.
Here are some of the remarks I've made recently on the subject.
venus% fgrep -n "Give me liberty" *.doc
liberty.doc:4:Give me liberty or I'll give you what-for.
liberty.doc:5:Give me liberty or I'll give you a piece of my mind.
liberty.doc:7:Give me liberty or give me a salami on rye.
liberty.doc:8:Give me liberty or give me a raise.
liberty.doc:9:Give me liberty or give me a free lunch.
liberty.doc:10:Give me liberty or give me a break.
liberty.doc:11:What I want for my birthday is:  Give me liberty.
liberty.doc:12:You know what I always say--Give me liberty!
liberty.doc:13:Give me liberty or I'll go on strike.
liberty.doc:14:"Give me liberty" seems to lose its meaning after a while.
liberty.doc:15:Maybe the boy who cried wolf really cried, "Give me liberty!"
patrick_henry.doc:4:Give me liberty or I'll give you what-for.
patrick_henry.doc:5:Give me liberty or I'll give you a piece of my mind.
patrick_henry.doc:7:Give me liberty or give me a salami on rye.
patrick_henry.doc:8:Give me liberty or give me a raise.
patrick_henry.doc:9:Give me liberty or give me a free lunch.
patrick_henry.doc:10:Give me liberty or give me a break.
patrick_henry.doc:11:What I want for my birthday is:  Give me liberty.
patrick_henry.doc:12:You know what I always say--Give me liberty!
patrick_henry.doc:13:Give me liberty or I'll go on strike.
patrick_henry.doc:14:"Give me liberty" seems to lose its meaning after a while.
patrick_henry.doc:15:Maybe the boy who cried wolf really cried, "Give me liberty
!"
venus% rm liberty.doc
rm: remove liberty.doc? y
venus%
```

For more information on Shell Tool see *SunView 1 Beginner's Guide* and the `shelltool` manual page.

Console



The Console is a special Command Tool that displays error messages and other information from SunView and the SunOS operating system. An example of a Console frame is shown below.

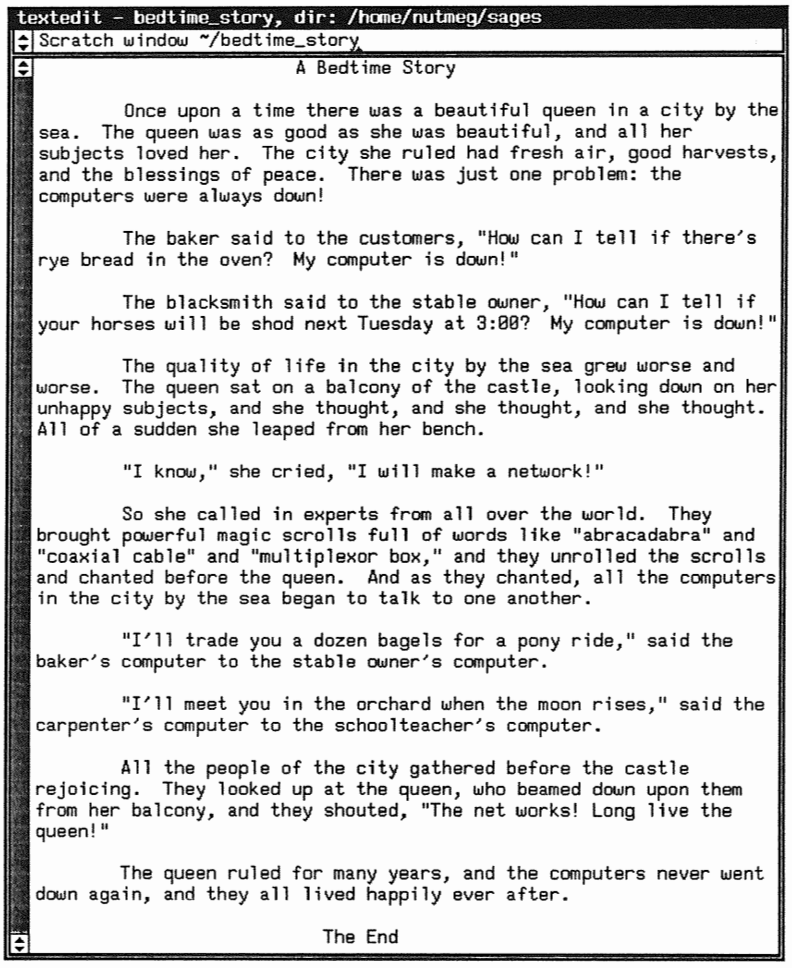
Figure 4-4 *Console Frame*

For more information see the `cmdtool` manual page.

Text Editor

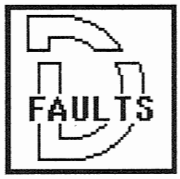
The Text Editor (`textedit`) is a mouse-based text editing tool to create and to review text files. There are mouse functions to select regions of text, and functions to perform cut and paste operations. An example of a Text Edit frame is shown below.

Figure 4-5 *Text Editor Frame*



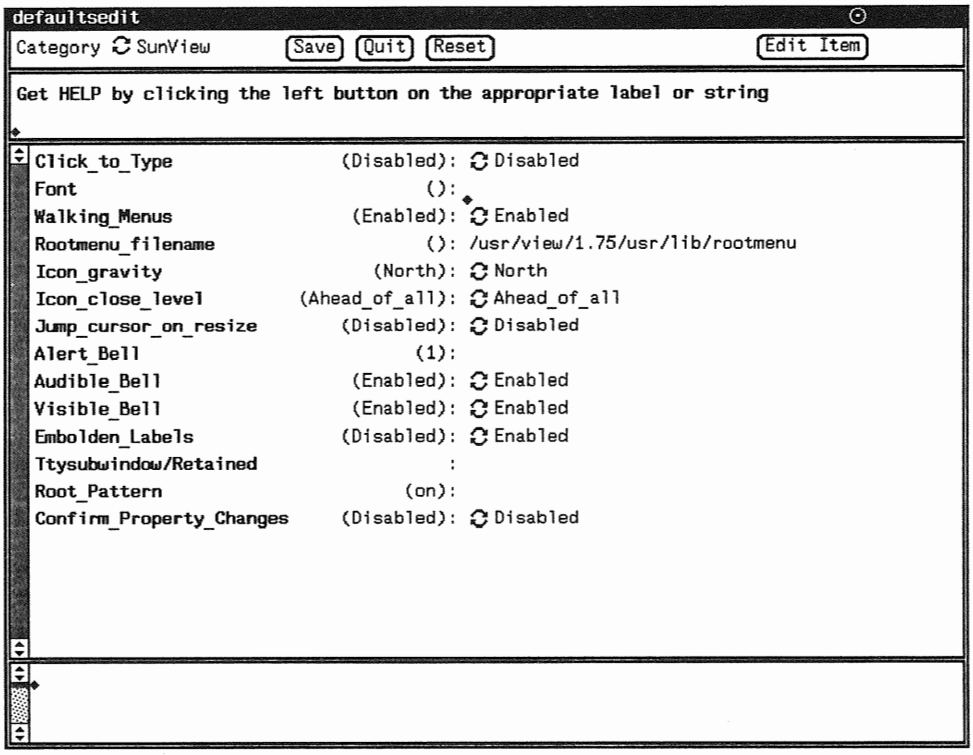
For more information on Text Editor see *SunView 1 Beginner's Guide* and the `textedit` manual page.

Defaults Editor



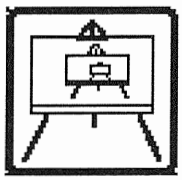
The Defaults Editor is a tool for customizing SunView. You can change default settings of SunView to create an environment comfortable for you. An example of a Default Editors frame is shown below.

Figure 4-6 *Defaults Editor Frame*



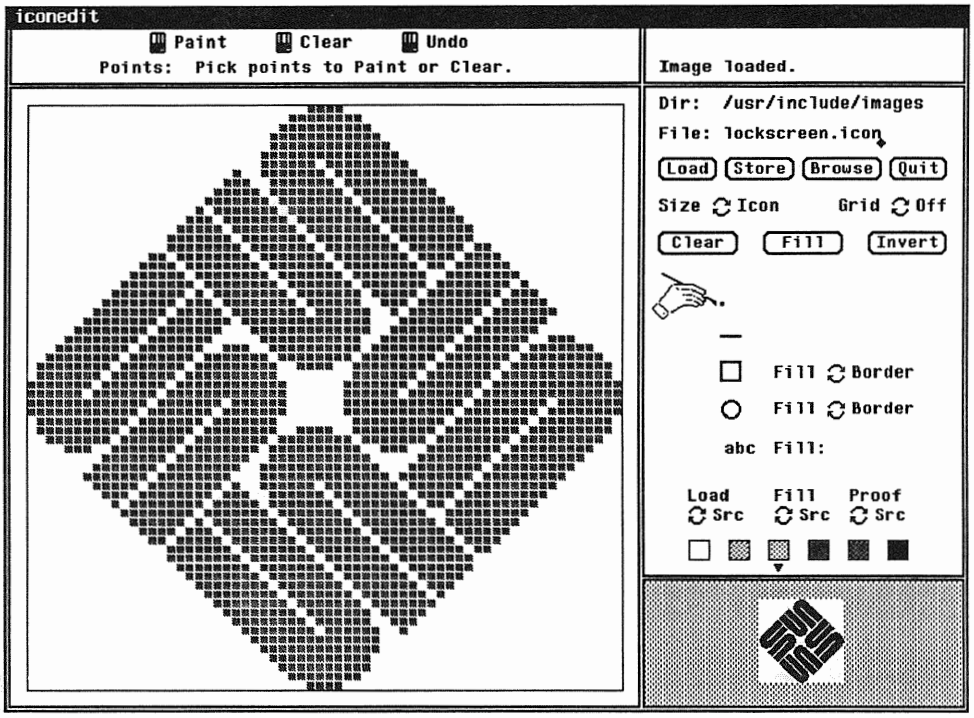
For more information on Defaults Editor see *SunView 1 Beginner's Guide* and the `defaultsedit` manual page.

Icon Editor



The Icon Editor (`iconedit`) is a window-based drawing editor for designing icons and cursor images. An example of an Icon Edit frame used to draw the Sun logo is shown below.

Figure 4-7 *Icon Edit Frame*

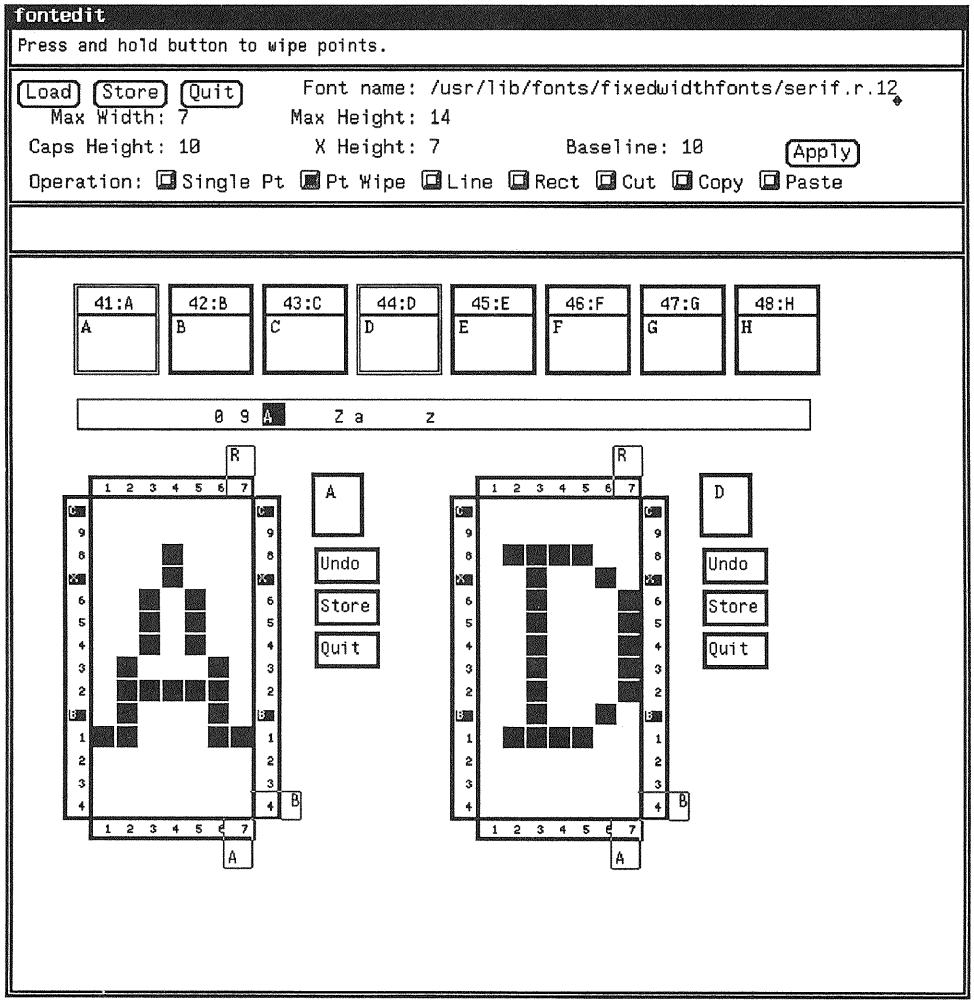


For more information on Icon Edit see *SunView 1 Beginner's Guide* and the `iconedit` manual page.

Font Editor

The Font Editor tool (`fontedit`) creates and edits fonts. You load a font and select a portion of that font to be displayed in the eight small windows. Two characters at a time may be selected for editing. These characters are displayed in large format so you can make pixel-by-pixel modifications to them. A Font Editor frame is shown below.

Figure 4-8 *Font Editor Frame*

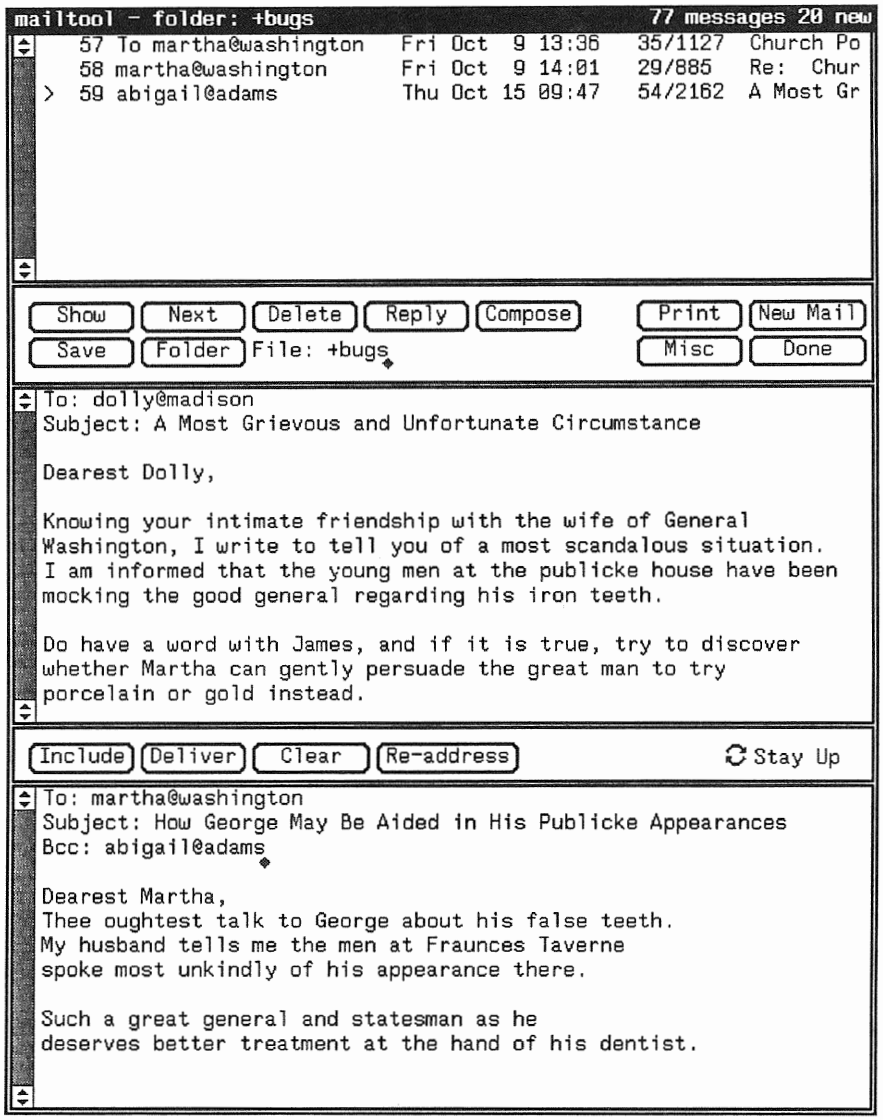


For more information on the Font Editor see *SunView 1 Beginner's Guide* and the `fontedit` manual page.

Mail Tool

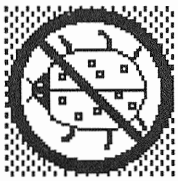
Mail Tool (`mailtool`) provides windows for sending and for receiving mail. You can use the standard text facilities of SunView (`textedit`) to compose messages and to move text between messages. A Mail Tool frame is shown below.

Figure 4-9 *Mail Tool*



For more information on Mail Tool see *Mail and Messages: Beginner's Guide* and the `mailtool` and `mail` manual pages.

Dbx Tool

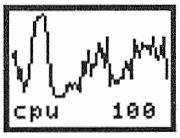


The Dbx Tool (`dbxtool`) is a source-level debugger for C, Pascal and Fortran 77 programs. To use `dbxtool`, you point to the object you want to operate on (say display the value of a variable) and then you 'push a button' to get the operation done. Below is an example of a Dbx Tool Frame.

Figure 4-10 *Dbx Tool Frame*

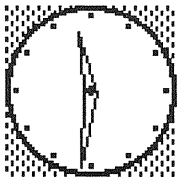
For more information on Dbx Tool see *Debugging Tools* and the `dbxtool` and `dbx` manual pages.

Performance Meter



Performance Meter (`perfmeter`) is a tool for observing system performance. The icon for measuring performance is a speedometer-like gauge with a needle that moves as system conditions change. There are numerous options that you can measure with this tool such as processor performance, network performance, memory performance and I/O performance. For more information on Performance Meter see *SunView 1 Beginner's Guide* and the `perfmeter` manual page.

Clock Tool



Clock (`clocktool`) displays the date and time in the form of a clock on the screen. The display can be customized in various ways:

- round or square clock,
- Arabic or Roman numerals on the clock face,
- display the seconds,
- display the day of the week.

The Clock Frame displays the day, full date with year and the current time. An example of the Clock Frame appears below.

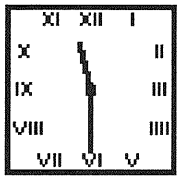
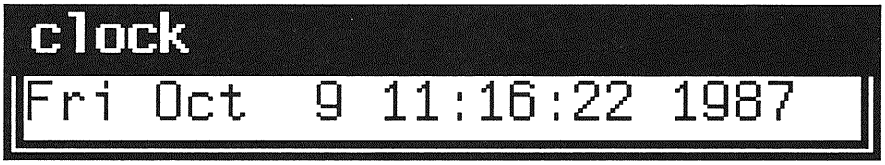


Figure 4-11 *Clock Frame*

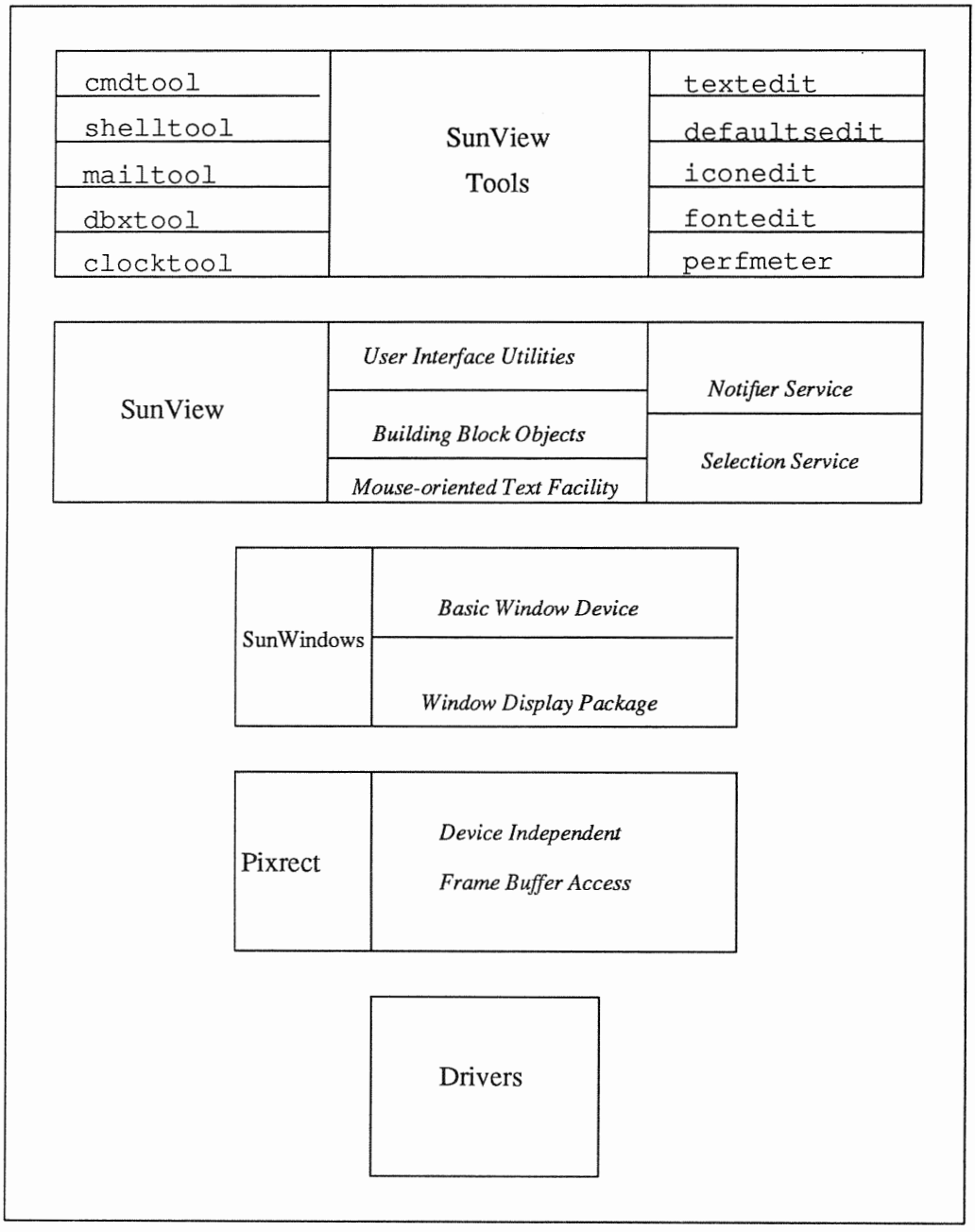


For more information on Clock Frame see *SunView 1 Beginner's Guide* and the `date` manual page.

4.3. Programming Interface

SunView has a programming interface for building window-based applications. The programming interface is a collection of function libraries. Since SunView is based on SunWindows which is based on Pixrects, an application can also make calls to SunWindows and Pixrect functions. This hierarchy is described in the figure below.

Figure 4-12 *Application Software Interfaces*



4.4. SunView Facilities

SunView has user interface utilities and building blocks for creating application software. The user interface utilities include:

- the *Notifier* — the agent that distributes events among multiple applications; and,
- the *Selection Service* — the agent that manages text selections.

The building blocks for creating application software include:

- *text subwindows* for displaying textual data;
- *canvas subwindows* for displaying graphical information;
- *scrollbars* for scrolling backward and forward in a file;
- *control panels* containing 'pushbuttons' for selecting actions;
- *menus* to display choices for the user to select; and,
- *alerts* which display information, instructions, errors, and confirmations.

There is a general-purpose mouse-oriented text facility (`textedit`) with functions to select, extend, cut, and paste.

4.5. SunWindows Facilities

SunWindows facilities include the basic window manager and window display package.

Pixrects

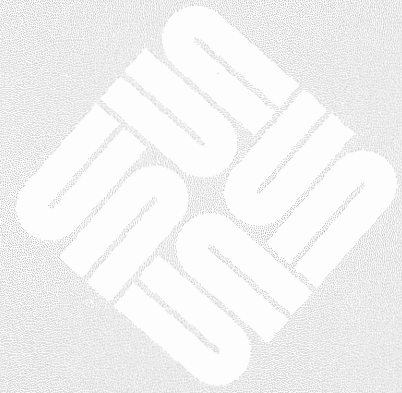
The Pixrect graphics library is a set of RasterOp routines common among all Sun workstations. With these routines, application programs can be written that access the display on all Sun products. In the Sun graphics software world, the Pixrect library is a low-level package, sitting on top of the device drivers.

Further Reading

For more information on programming the Sun Window System see *SunView 1 Programmer's Guide*, *SunView 1 System Programmer's Guide*, and *Pixrect Reference Manual.*

5

Using the Shells



5

Using the Shells

The SunOS operating system uses the SunView window system as an interface for user input. A program called the *shell* is used as a command interface for doing work on the system. As you type commands to the shell it reads and interprets the commands, passes the interpreted commands on to be performed, and waits for each command to finish before proceeding to the next.

Currently, there are two shells used on the SunOS operating system:

- the *C shell*, developed by William Joy at the University of California at Berkeley; and,
- the *Bourne* shell, named after its developer S.R. Bourne. The Bourne shell is the standard UNIX system shell.

The shell is a command interpreter and a programming language, and both shells can serve either purpose. The C shell, however, has more useful features for interactive use, such as command aliasing, job control, and a history mechanism. The Bourne shell, while providing fewer interactive features, runs faster and has a simpler syntax for writing shell programs.

Many similarities exist in the user interface for both shells. The major differences between the shells lie in the internal programming interface and the aliasing, job control, and history features offered only by the C shell.

5.1. User Interface Features of the Shells

The following user interface features are provided by both shells:

- *Command line analysis.* When a command is typed at the command line, the shell accepts and interprets the command you typed. The shell then runs the command, passing the remainder of the command line as arguments to the command. Each command is run as a separate *process.* Most of the time commands run in the *foreground,* or "while you wait". But, commands also can be run in the *background.* During a background process the shell returns immediately so you can continue typing more commands while it processes the command running in the background.
- *Redirection.* By default commands use input from the keyboard and output data and error messages to the terminal. Redirection enables you to manipulate standard input, standard output, and standard error to or from files, rather than from default locations. The process of directing standard input, output and error from commands to files is called 'redirection of standard

input and output'.

- *Pipelines.* Pipelines connect the standard output from one command to the standard input of the next command. New commands can be built by connecting (or piping) UNIX commands together.
- *Filename expansion.* Both shells use *metacharacters* to identify file and directory names. The metacharacters include:

  `?` matches any single character in a filename.

  `*` metacharacter matches any string of characters (including the empty string) in a filename. For example, `*` on its own matches all files in a directory.

  `[]` metacharacters form *character classes* that indicate ranges of letters or digits in filenames. For example, `[a-m]*` matches all files starting with the letters `a` through `m`.

- *Search path.* A user-settable path to locate commands is available in both shells. This path is generally defined in a ''startup'' file and executed during log in. Along with defining a search path, other features can be included in this profile file to tailor the environment to your special needs.

For further details on the user interface features of the shells, see *Getting Started with SunOS: Beginner's Guide*, *Doing More with SunOS: Beginner's Guide*, and the *csh* or *sh* manual pages.

5.2. Programming Features of the Shells

Besides acting as a command interface, both shells also can be used as programming languages. Programs created using the shell programming language are called *shell scripts*.

A shell script is a file containing a sequence of commands. When the filename is typed as a command to the system, the commands in the file are read and then sequentially executed. Shell scripts are quickly created, debugged, and executed because there is no need to run a compiler and loader and recompile to change things. Shell scripts frequently provide a good first cut for an application when using the shell programming features and connecting existing UNIX system commands together. Shell scripts provide for what is often called *rapid prototyping* for applications.

The following are programming features common to both shells:

- setting and accessing variables;
- substituting arguments from the command line;
- `if ... then ... else` constructs for conditional execution of commands;
- `case` switches for selecting groups of commands;
- `while` loops for executing groups of commands iteratively;
- `for` loops for executing groups of commands over lists of files or variables;
- `break`, `continue` and `exit` to get out of looping constructs; and,

- features for taking action on traps and interrupts. These features are used to do 'clean up' action if a shell process is interrupted.

For further details on programming features of the shells, see *Doing More with SunOS: Beginner's Guide* and the *csh* and *sh* manual pages.

5.3. The Shells

The SunOS operating system includes the C shell and the Bourne shell.

C Shell

The C shell command name is `csh`. On Sun systems users normally use the C shell for interactive use. The C shell was developed at the University of California, Berkeley as part of the 4.2BSD operating system.

Another type of shell available on the Sun system is the remote shell, `rsh`. This shell is used to execute commands on other systems. The `rsh` can be either a C shell or a Bourne shell. For more information, see the `rsh` manual page.

The C shell provides a flexible user interface. Its major external features include:

- A *history* facility to reissue previous commands. The shell maintains a history buffer of the last *n* commands, where *n* is specified by the user. The history list is displayed or commands in the history list are referred to by their event number in the list or by searching for substrings of the actual name of the command.
- A *history substitution mechanism* used with the history mechanism described above. Parts of previous commands can be substituted either singlely, or globally.
- A *job control* feature suspends jobs and switches a job back and forth between the foreground and the background. Foreground jobs can be sent to the background and background jobs can be brought to the foreground. You can determine which jobs are running and kill them if required.
- An *alias* mechanism creates and renames commands.
- A feature to optionally announce the presence of mail as it arrives in the system.

The major programming features of the C shell include:

- a programming syntax resembling the C programming language; and,
- the ability to compose compound commands using the shell's internal programming constructs.

For further information, see the manual *Doing More with SunOS: Beginner's Guide* and the `csh` manual page.

Bourne Shell

The Bourne shell command name is `sh`. The Bourne shell is the standard UNIX system version 7 command language interpreter, but is used generally on Sun systems for creating shell scripts. The Bourne shell is named after its originator, S. R. Bourne from Bell Laboratories. The Bourne shell has most of the same external features as the C shell, but lacks a history facility, a job-control facility, and a history substitution facility.

The major programming features of the Bourne shell include:

- internal programming features modeled after the Algol-68 programming language;

- support for 8-bit non-ASCII characters for commands and their arguments; and,
- a school of thought that maintains that writing shell scripts is easier and faster with the Bourne shell.

For more information, see the manual *Doing More with SunOS: Beginner's Guide* and the `sh` manual page.

5.4. Related Shell Utilities

There are numerous tools or utilities used to make shell programming easier. These tools generally are divided into two basic categories: text manipulating utilities and tools which are extensions of the control structure of the shell programming language.

Text Manipulating Utilities

The following list describes some of the basic utilities used for manipulating text when creating shell scripts.

Displaying Command Line Arguments

The `echo` command displays (or echoes) the remainder of its command line. It is used for diagnostics or prompts in shell programs, or for inserting data into a pipeline. For further details see *Doing More with SunOS: Beginner's Guide* and the `echo` manual page.

Making Shell Scripts Executable

The `chmod` command changes permissions on files. This command is used to make script files executable programs. For further details see *Doing More with SunOS: Beginner's Guide* and the `chmod` manual page.

Breaking Lines Apart

The `cut` command extracts selected fields of data from lines in a file. The `cut` command is useful for cutting out columns from tables or fields from a line in a file or from the output of a command. For further details see the `cut` manual page.

Putting Lines Together

The `paste` command merges corresponding lines from files to form a single line. For further details see the `paste` manual page.

Complex Editing

The `sed` utility is a stream editor used for creating complex global substitutions in pipelines. `sed` can not be used interactively. For further details see the `sed` manual page and the `ed` manual page for information on regular expressions.

Pattern Searching

The `grep` command is part of a family of pattern searching commands, used to locate particular patterns of characters in files. For further details see *Getting Started with SunOS: Beginner's Guide*, *Doing More with SunOS: Beginner's Guide*, and the `grep` manual page.

Translating Characters

The `tr` command changes one character into another. For further details see the `tr` manual page.

Control Structure Extensions

The following is a list of utilities used to extend the shell's control structures.

Conditional Testing

The `test` command tests data for use in shell conditionals. Features of the `test` utility include:

- string comparison;
- determining the nature of a file (is it a directory, file, link, symbolic link, and so on), and the file's accessibility (is it readable, writeable, executable, and so on); and,
- Boolean combinations of the above.

For further details see *Doing More with SunOS: Beginner's Guide* and the `test` manual page.

Evaluating Expressions

The `expr` command evaluates expressions that appear on its command line as arguments. Features of the `expr` utility include:

- string computations;
- integer arithmetic; and,
- pattern matching.

For further details see *Doing More with SunOS: Beginner's Guide* and the `expr` manual page.

Waiting on Process

The `wait` command waits for termination of asynchronously running processes. For further details see the `wait` manual page.

Suspending Execution

The `sleep` command suspends a command execution for a specified time. For further details see the `sleep` manual page.

Blocking Hangups

The `nohup` command is used mainly for dialup lines. It runs a command which ignores the hang up of the telephone. For further details see the `nohup` manual page.

Changing Priority

The `nice` command runs a command in low or high priority. For further details see the `nice` manual page.

Killing Processes

The `kill` command terminates specified processes or jobs. For further details see *Doing More with SunOS: Beginner's Guide* and the `kill` manual page.

Scheduling Actions

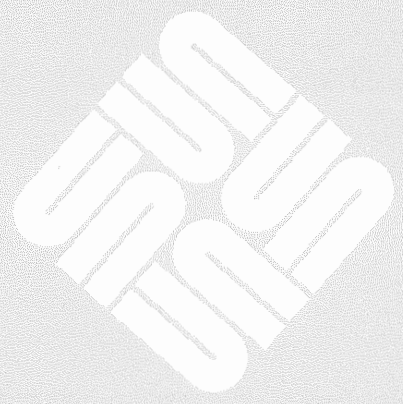
The `at` command schedules execution of a command at an arbitrary time. For further details see *Doing More with SunOS: Beginner's Guide*, `at`, `batch`, and `crontab` manual pages.

Diverting Output

The `tee` command splits the output from commands into one or more files. For further details see *Doing More with SunOS: Beginner's Guide* and the `tee` manual page.

6

User Access and Commands



6

User Access and Commands

User accounts and optional passwords are used to gain access to the SunOS operating system. The `login` and `passwd` programs are used to log on to your local workstation. Once logged on to a workstation you can remotely log into other systems using the `rlogin` program. You also can access geographically remote hosts with the `tip` program described in chapter 10, *Communication Facilities*.

6.1. Gaining Access to the System

The `login` command is used to gain initial access to the system. Features of the `login` command include:

- ability to sign on as a new user;
- ability to verify password and to establish user's individual and group (project) identity;
- define terminal characteristics;
- establish a working directory;
- announce the presence of mail;
- publish a message of the day;
- execute user-specified startup files; and,
- execute command interpreter or other initial program.

For more information see *Using the Network: Beginner's Guide* and the `login` manual page.

6.2. Changing your Password

The `passwd` command establishes your initial password or changes an existing password. Passwords can be changed and are encrypted for security. For further details see *Getting Started with SunOS: Beginner's Guide* and the `passwd` manual page.

6.3. Logging Out

The `logout` command logs you out of the system. An optional `.logout` file can be created containing good bye messages, jokes, etc. For further details see *Getting Started with SunOS: Beginner's Guide* and the `logout` manual page.

6.4. Useful Commands

The following is a brief description of some useful commands. A pointer for where to find more information for each command also is included. Files, directories, and the commands used to manipulate them are described in chapter 7, *Working with Files*.

Displaying Date and Time

The `date` command displays the current date and time. As superuser you can use `date` to set the system date and time. For further details see *Getting Started with SunOS: Beginner's Guide* and the `date` manual page.

Finding Out Who is Logged on

The `who` command displays a list of presently logged on users, ports and login times. The `w` command displays a list of who is logged in and which command they are running. For further details see *Doing More with SunOS: Beginner's Guide* and the `who` and `w` manual pages.

Displaying What is Going on

The `ps` command displays information for active processes. The display can contain the following process information:

- a list your own or everybody's processes; and,
- can provide optional process status information such as, state and scheduling information, priority, attached terminal, what a process is waiting for, and size.

For further details see *Doing More with SunOS: Beginner's Guide* and the `ps` manual page.

Logging into Another Machine

The `rlogin` command logs you onto another machine in the local network. For further details see *Using the Network: Beginner's Guide* and the `rlogin` manual page.

Running Commands on Other Machines

The `rsh` command executes a shell on a remote host in the local network. For further details see *Using the Network: Beginner's Guide* and the `rsh` manual page.

Reminder Service

The `calendar` command provides an automatic reminder service of events for today and tomorrow. For further details see *Getting Started with SunOS: Beginner's Guide* and the `calendar` manual page.

Calculator — `bc`

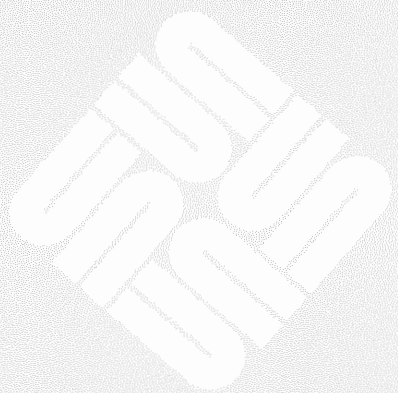
The `bc` command is a C-like interactive interface to the `dc` desk calculator. Its features include:

- all the capabilities of `dc` with a high-level syntax;
- arrays and recursive functions;
- when called returns immediate evaluation of expressions and evaluation of functions;
- arbitrary precision elementary functions exp, sin, cos, atan; and,
- go-to-less programming.

For further details see *Getting Started with SunOS: Beginner's Guide* and the `bc` and `dc` manual pages.

7

Working with Files



7

Working with Files

The SunOS operating system uses *files* to organize and to store information. The system file organization is called a *hierarchical file system*, and it is one of the SunOS operating system's major strengths.

7.1. What is a File?

A *file* is a collection of data such as words, characters, numbers, etc., that is stored together. Files have several *attributes* — the major and most important attribute being the *name* of the file. Other attributes include the access permissions, size, date, and time the file was created, and the date and time the file was last changed.

7.2. What is a Directory?

Files reside in a directory. A *directory* is a special kind of file that contains other files and directories. Since a directory can contain other directories, the hierarchy, in principle, can extend to unlimited depth.

The SunOS operating system treats just about every object in the system as a file. For example, the physical memory of the computer can be accessed and manipulated like a file.

Home Directory

When new accounts are created an initial working directory also is created. This initial directory is called a *home* directory and you are positioned in this home directory when you log on to the system.

Working Directory

Changing directories positions you at a specified place in the file system hierarchy. This specified place is called the *current directory* or the *working directory*.

For more information on files and directories see *Getting Started with SunOS: Beginner's Guide.*

7.3. Access Permissions

When files or directories are created, access permissions are also created as part of the file's attributes. Access permissions establish who can access the directory or file, and what operations they can perform. The access permissions are called the *mode* of the file. Three types of users can access a directory or file:

- the owner — the person who originally created the directory or file;
- the group — the group of users to which the owner belongs; and,
- the public — any one other than the owner and the members of the owner's group.

For each type of user there are three kinds of access to a directory or file:

read view the contents of the file or view the inside the directory;

write change the contents of the file or create other directories or files inside a directory; and,

execute type the name of the file as a command or perform certain operations to traverse through the directory.

For more information see *Doing More with SunOS: Beginner's Guide.*

7.4. Manipulating Files and Directories

There are two types of utilities that manipulate files and directories:

- the first type creates, removes, and renames files and directories; and,
- the second type copies and manipulates the contents of files and directories.

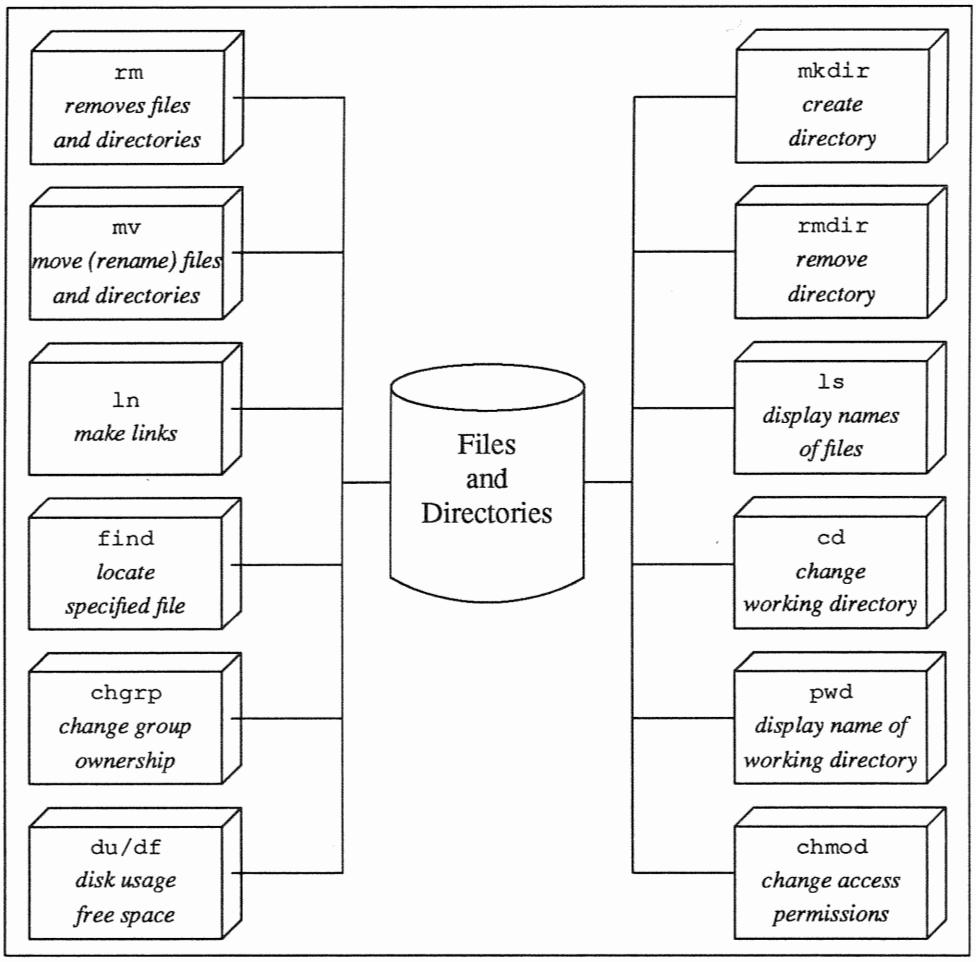
In general, the first type of programs manipulate the attributes of files and directories, while the second type of programs manipulates the data stored in the files.

7.5. Managing Files and Directories

There are numerous utilities for managing files and directories. Simple files are created with such utilities as `cat` (described later), or with one of the system's text editors described in chapter 9, *Editing Text Files*. Directories are created with a special command, described below.

The figure, below shows commands used to manage files and directories. Each of these commands is described below the figure, and a reference is included for further information.

Figure 7-1 *Commands to Manage Files and Directories*



Displaying a List of File Names

The `ls` command displays an alphabetically, sorted list of file and directory names. The `ls` command can display filenames in many alternate ways, such as:

- displaying filenames in reverse alphabetical order;
- displaying filenames by time of last access;
- mark the display showing directories distinct from files and mark executable (program) files;
- display optional information — size, owner, group, date last modified, date last accessed, permissions, and i-node number.

For further details see *Getting Started with SunOS: Beginner's Guide* and the `ls` manual page.

Changing the Working Directory

The `cd` command changes the working directory to a specified place in the directory hierarchy. A `cd` command without any specified directory name, always returns to the home directory. For further details see *Doing More with SunOS:*

Beginner's Guide and the `cd` manual page.

Displaying Current Directory Name

The `pwd` command displays the name of the current working directory. For further details see *Getting Started with SunOS: Beginner's Guide* and the `pwd` manual page.

Creating a New Directory

The `mkdir` command creates a new directory. For further details see *Getting Started with SunOS: Beginner's Guide* and the `mkdir` manual page.

Removing a Directory

The `rmdir` removes a directory from the file system. The directory must be empty (contain no files or other directories) before it can be removed. For further details see *Doing More with SunOS: Beginner's Guide* and the `rmdir` and `rm` manual pages.

Changing Access Permissions

The `chmod` command changes the the access permissions (or mode) for one or more files. Only the owner of the file, or the superuser, can change a file's access permissions. For further details see *Doing More with SunOS: Beginner's Guide* and the `chmod` manual page.

Changing Group Ownership

Groups can be created on the system for users who are associated with a project. When files are created, they are associated not only with an owner, but also with a group. The `chgrp` command changes the group a file belongs to. For further details see *Doing More with SunOS: Beginner's Guide* and the `chgrp` manual page.

Renaming or Moving Files

The `mv` command moves a file or files from one place to another in the directory hierarchy. The `mv` command actually renames files or directories. When a file is moved from one place to another, the destination name can be different from the source name. Moving a file with a different source and a different destination names within the same directory effectively renames the file without physically moving it.

The `mv` command also moves entire directory hierarchies from place to place in the file system. For further details see *Getting Started with SunOS: Beginner's Guide* and the `mv` manual page.

Creating File Links

The `ln` command creates a link or an alias to an existing file. There are two types of links:

- *hard* links can be created only within a file system and make a real physical link in the structure; and,
- *symbolic* links are similar to a macro or string substitution and can span file systems. The difference between the two types of links becomes apparent when removing files as described in the `rm` command, below. For further details see *Doing More with SunOS: Beginner's Guide* and the `ln` and `rm` manual pages.

Removing Files and Directories

The `rm` command removes files and, when used with special options, also removes entire directory hierarchies. The `rm` command can do the following:

- remove only the *name* of the file if any other names are linked to the file. This is an important effect of symbolic links as discussed in the `ln` command. When removing symbolically linked files you can end up with a symbolic link that does not point to anything.
- interactively delete files; an option of the `rm` command steps through a directory asking if you want to delete a file.
- used with the special recursive option, `-r` deletes entire directory hierarchies.

For further details see *Getting Started with SunOS: Beginner's Guide*, *Doing More with SunOS: Beginner's Guide*, and the `rm`, `rmdir`, and `ln` manual pages.

Finding Specified Files and Directories

The `find` command prowls the directory hierarchy locating every file that meets a specified criteria. Any directory can be the root or starting place for the search. The `find` command can do specified operating system commands on each file that matches the specified criteria. The criteria includes:

- filename matches a given pattern;
- creation date in given range;
- date of last use in given range;
- given permissions;
- given owner;
- given special file characteristics; and,
- Boolean combinations of above.

For further details see *Doing More with SunOS: Beginner's Guide* and the `find` manual page.

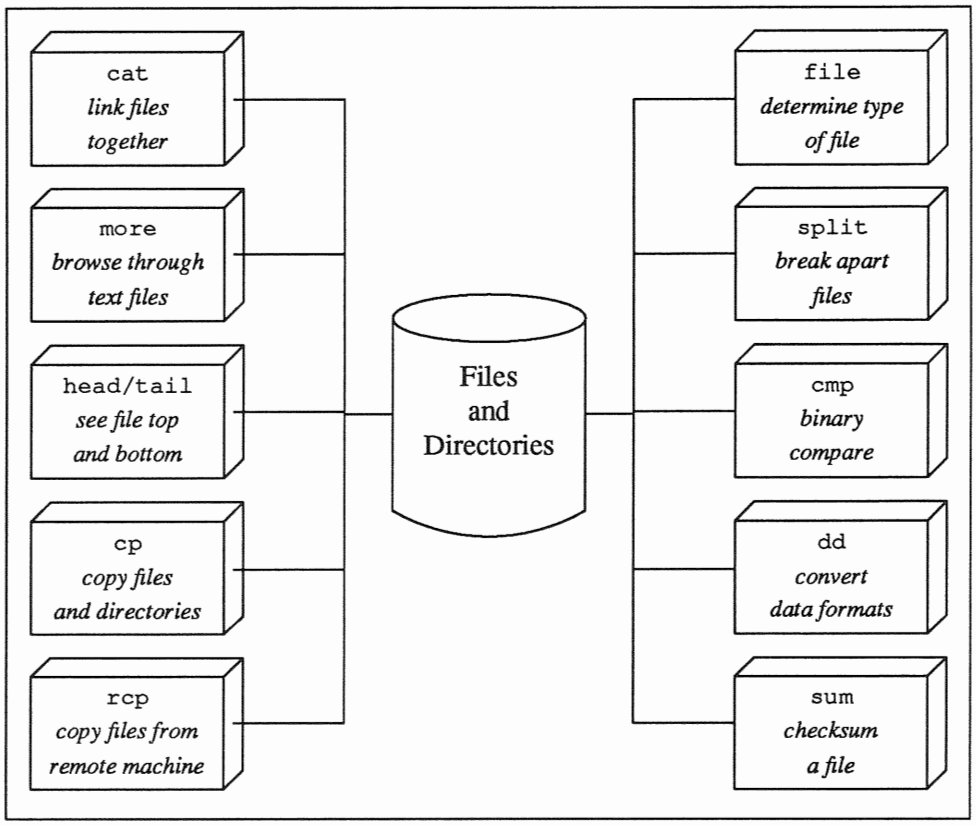
Displaying Statistics for the File System

The `df` command displays the amount of free space on file systems, and the `du` command displays a summary of total space occupied by all files in a hierarchy. For further details see *Doing More with SunOS: Beginner's Guide* and the `df` and `du` manual pages.

7.6. File Manipulation Facilities

Commands described so far are used for creating files and directories and manipulating their attributes. The figure below shows commands used to manipulate the contents of files and directories. Each of these commands is described below the figure, and a reference is included for further information.

Figure 7-2 *Commands to Manipulate the Contents of Files and Directories*



Determining the Type of a File

The `file` command determines what kind of information is in a file. By consulting the file system index and by reading the file itself, the `file` command determines, among many others, if a file is one of the following:

- a plain text file — ASCII;
- a repository of files — directory;
- input for one of the text formatting packages troff, nroff, or eqn input text;
- source code for the C programming language — C program text. The `file` command also can determine if it is FORTRAN source code; and,
- an executable file.

For further details see *Doing More with SunOS: Beginner's Guide* and the `file` manual page.

Combining Files

The `cat` command joins together files from end to end (concatenates) and displays the result to standard output (the screen). The `cat` command has a variety of uses, including:

- inserting data into a pipeline;

- optionally displaying non-printing characters;
- optional number lines; and,
- buffer output that appears in pieces.

The `cat` command works on any file regardless of its contents. For further details see *Getting Started with SunOS: Beginner's Guide* and the `cat` manual page.

Copying Files and Directories

The `cp` command copies files (and optionally whole directory hierarchies) from one place to another. The `cp` command can:

- copy a set of files to a directory;
- copy any file regardless of its contents; and,
- used with the `-r` (recursive) option copy entire directory hierarchies.

For further details see *Getting Started with SunOS: Beginner's Guide* and the `cp` manual page.

Remote Copying

The remote copy command, `rcp`, copies files and directories to and from other machines on the local network. For further details see *Using the Network: Beginner's Guide* and the `rcp` manual page.

Browsing through a File

The `more` command scrolls forward through the contents of a file. Features of the `more` command include:

- account for the size characteristics of a terminal and displays a page of a file at a time;
- option to scroll forward a page or a line at a time by typing keys; and,
- option to skip forward to selected patterns in the file.

For further details see *Getting Started with SunOS: Beginner's Guide, Doing More with SunOS: Beginner's Guide* and the `more` manual page.

Doing Binary Compares

The `cmp` command performs binary comparison on a pair of files and reports the first place where it finds a difference in the data. For further details see the `cmp` manual page.

Displaying Top or Bottom of File

The `head` command displays the top ten lines of a file, and the `tail` command displays the last ten lines of a file. For further details see *Doing More with SunOS: Beginner's Guide* and the `head` and `tail` manual pages.

Splitting a File

The `split` command divides a file into a specified number of pieces. This command is useful for dealing with some older utilities that have limitations on the number of lines they can digest at one time. For further details see the `split` manual page.

Converting Data Formats

The `dd` command translates physical file formats to exchange data with other operating systems. For further details see the `dd` manual page.

Doing a File Checksum

The `sum` command sums the words of a file, providing a convenient checksum. For further details see the `sum` manual page.

7.7. Summary of File and Directory Commands

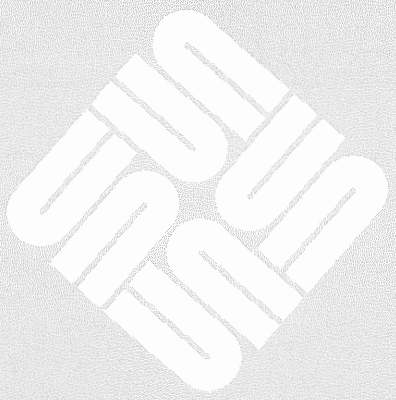
The following table provides an alphabetical list of the file and directory commands presented in this chapter.

Table 7-1 *Summary of File and Directory Commands*

Program Name	***Function***
`cat`	join files
`cd`	change directory
`chgrp`	change group
`chmod`	change access permissions
`cmp`	do a binary compare of files
`cp`	copy files
`dd`	convert file formats
`df`	display free space
`du`	display disk usage
`file`	find file type
`find`	find files
`head`	display top of file
`ln`	create links to file
`ls`	display file and directory names
`mkdir`	create directory
`more`	page through file
`mv`	move or rename file
`pwd`	display name of current working directory
`rcp`	remote copy files
`rm`	remove file
`rmdir`	remove directory
`split`	split file
`sum`	checksum file
`tail`	display end of file

8

Editing Text Files

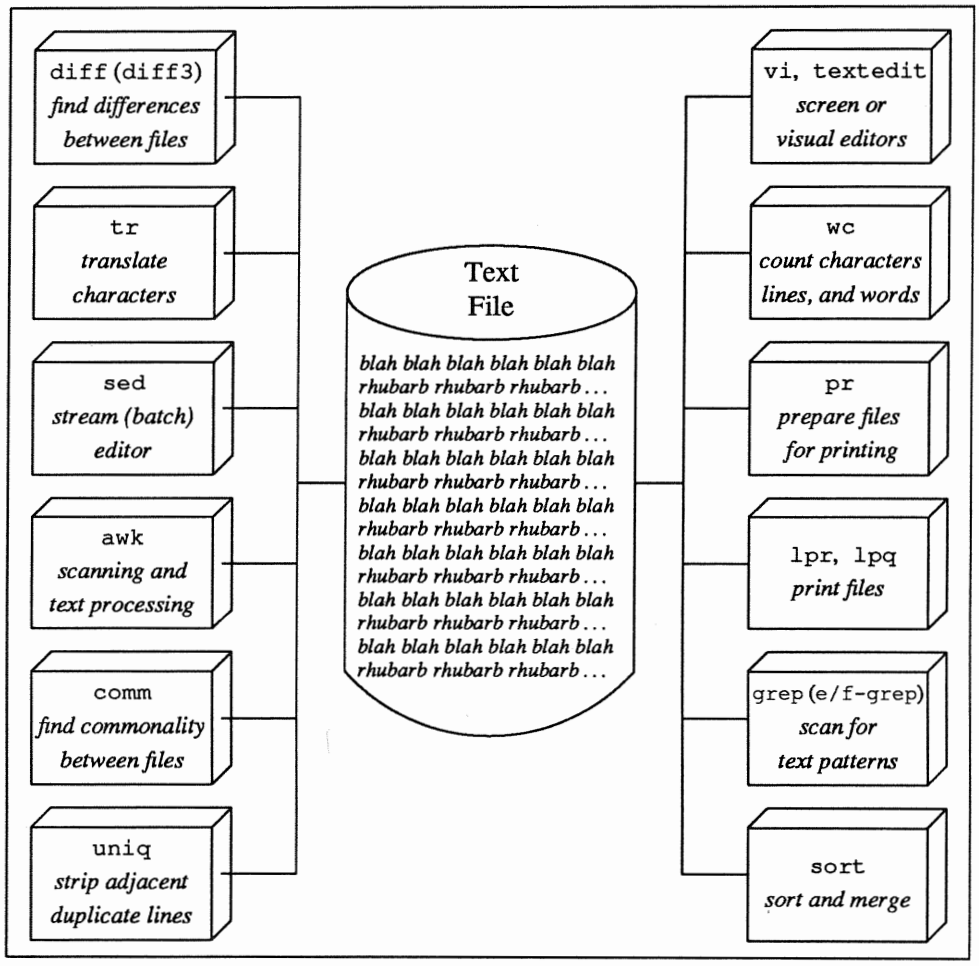


8

Editing Text Files

Since files are used to store and to organize data, it is reasonable that there are numerous utilities to manipulate a file's contents. The diagram below summarizes common text editing utilities. The sections which follow describe the functions and abilities of these text editing utilities in more detail and provide references for further information.

Figure 8-1 *Commonly Used Text Editing Utilities*



8.1. Printing Text Files

While the `lpr` command sends files to the printer, it does no transformations on the text whatsoever and places no interpretations on the data in the text file. No 'carriage-control' characters are embedded as in other systems. The `pr` command (described below) can be used to *pr*epare a file for printing with `lpr`.

Preparing Files for Printing

The `pr` command prepares files for printing with a printer program — usually `lpr`. `pr` is a simple but flexible utility, and can do the following:

- place a title, date, and page number on every page;
- arrange text into multiple columns;
- merge several files into multiple columns in parallel; and,
- specify the number of columns on a page and the number of lines on a page.

For further details see *Editing Text Files* and the `pr` manual page.

Printing Files Offline

The `lpr` utility is the line printer control program for spooling arbitrary files to printers for off-line printing. One option to `lpr` can use the `pr` program described above to paginate text before it is printed.

In addition to printing files, the line printer spooler system has facilities to examine the print queue (`lpq`) and to remove jobs from the print queue (`lprm`). For further details see *Editing Text Files* and the `lpr`, `lpq`, and `lprm` manual pages.

8.2. Interactive and Non-interactive Text Editors

Text editors and other text manipulation software comprise a large part of the facilities offered in the SunOS operating system. Most text editing utilities rely on *regular expressions* to specify *text patterns* for searching. The major capabilities of searching using regular expressions include:

- match single characters or strings of characters;
- match any arbitrary character;
- match classes of characters; such as, match any lower-case letter, or match upper-case letters in the range I thru M.
- match closures — zero to many of the previously mentioned patterns;
- match specified patterns only at the start or end of a line — such patterns are said to be anchored; and,
- match alternative patterns — known as *alternation*. Only some of the pattern scanning utilities, most notably `egrep` and `awk`, handle alternation.

Editing Text Files

In addition to the `textedit` mouse-driven editor described in chapter 4, *SunView — A Window Environment* there is a family of three editors; one a screen editor, and the other two, line editors that are the principal tools for creating and editing text.

Visual Editor

The `vi` editor is a screen-oriented display editor. It provides 'what you see is what you get' editing for either line-oriented or full screen terminals. `vi` is a powerful and flexible editor with regular expression searching and user-specific settings.

For further details see *Getting Started with SunOS: Beginner's Guide, Editing Text Files*, and the `vi` manual page.

Line Oriented Editor

The `ex` editor is the line-oriented parent of `vi`, based on the original `ed` editor. The `ex` editor subsumes all functions of the `ed` editor. For further details see *Editing Text Files* and the `ex` manual page.

Original Line Oriented Editor

The `ed` editor was the original line-editor for the UNIX system. Although it is superseded by more powerful display editors such as `vi`, `ed` still has its place among some utilities which generate `ed` commands for automatic editing. An interactive context editor, `ed` provides random access to all lines of a file. Its features include:

- find lines by number or pattern;
- add, delete, change, copy, move or join lines;
- permute or split contents of a line;
- replace one or all instances of a pattern within a line;
- combine or split files;
- escape to the shell command language during editing;
- do any of above operations on every pattern-selected line in a given range; and
- optional encryption for extra security.

For further details see *Editing Text Files* and the `ed` manual page.

Batch or Stream Editing

The `sed` command is a non-interactive stream text editor. It is a version of the `ed` editor, used for processing large files. Features of `sed` include:

- performs a sequence of editing operations on each line of an input stream of unbounded length;
- lines are selected by address or range of addresses; and,
- has control flow and conditional testing, multiple output streams, and multi-line capability.

For further details see *Editing Text Files* and the `sed` manual page.

8.3. Information Processing and Text Manipulation

Information processing includes utilities intended originally for statistical text processing, such as:

- counting lines, words, and characters in a file;
- rearching for specific patterns in a file;
- transliterating characters; and,
- sorting the contents of a file.

Utilities used for information processing are described below.

Counting Things in Files

The `wc` command counts the lines, words (blank-separated strings) and characters in a file. For further details see *Editing Text Files* and the `wc` manual page.

Translating Characters

The `tr` command changes one character into another by:

- translating one-to-one characters according to an arbitrary code;
- coalescing selected repeated characters; and.
- deleting selected characters.

For further details see *Editing Text Files* and the `tr` manual page.

Scanning for Text Patterns

The `grep` command is one command in a family of programs that search for patterns in a file. The name `grep` stands for 'Global Regular Expression Print' and is derived from typing the `ed` line editor global command:

```
g/RE/p
```

This command line prints every line that contains the specified regular expression (*RE*). Features of `grep` include:

- display all lines in a file that satisfy a regular expression;
- display all lines that fail to match;
- display count of matches; and,
- display first match in each file.

Three flavors of a pattern matching program

There are actually three programs in the `grep` family:

`grep` is the original pattern scanning program. `grep` handles regular expressions containing *any* character, *character classes*, *anchored matches*, and *closures*.

`egrep` is the extended version of `grep`. `egrep` handles all the regular expressions that `grep` can handle, plus *alternation* — look for an occurrence of pattern A *or* B *or* C, and so on.

`fgrep` searches for fixed strings. The only metacharacters supported are those that anchor the pattern to the beginning or end of a line. The fixed strings may be in a file.

For further details see *Getting Started with SunOS: Beginner's Guide*, *Doing More with SunOS: Beginner's Guide*, *Editing Text Files*, and the `grep`, `egrep`, and `fgrep`, manual pages.

Sorting Files

The `sort` command is the main general-purpose sort utility. `sort` merges or sorts ASCII files line-by-line. This program is radically different from the traditional sort-merge utilities found on other computer systems because `sort` does not need input in fixed width fields starting in specific columns. Instead, `sort` breaks lines of a file into *fields* separated by white space (you can specify the field delimiter). Features of `sort` include:

- no limit on input size;
- sort up or down;
- sort lexicographically or on numeric key;
- sort by multiple keys located by delimiters or by character position;
- sort upper case together with lower into dictionary order; and,
- optionally suppress duplicate data.

Removing Successive Duplicate Lines

The `uniq` command collapses successive duplicate lines in a file into one line. The `sort` command also performs this function, but it is useful, at times, to have the facility available as a separate function. `uniq` reports on lines that were originally unique, duplicated, or both, and displays a redundancy count for each line.

Sorting Topologically

The `tsort` command is a topological sort that converts a partial order into a total order. For further details see *Editing Text Files* and the `sort`, `uniq`, and `tsort` manual pages.

Scanning Patterns and Processing Text

The `awk` command is a pattern scanning and processing language. With this language it is easy to specify many data transformations and selection operations. `awk` can be viewed as a 'programmable report generator'. It contains all the capabilities of the `grep` family of pattern scanners, but can manipulate data from text files and also can perform computations.

The `awk` programming language is named after its authors Aho, Weinberger, and Kernighan. Features of `awk` include:

- Searches input file for specified patterns and does actions on each line of input that satisfies the selection criteria. Patterns include regular expressions in the style of `grep` and `egrep`, arithmetic and lexicographic conditions, and Boolean combinations and ranges of these.
- Treats data as string or numeric as appropriate.
- Breaks input into records and fields. Fields are referenced as variables, and records can span multiple lines.
- Expression and string manipulation works on variables and arrays (with non-numeric subscripts). There is a full set of arithmetic operators and control flow in the style of the C programming language.
- Output is formatted as desired, and is directed to multiple output streams.

For further details see *Editing Text Files* and the `awk` manual page.

Displaying Differences Between Files

The `diff` command compares two files and reports differences. `diff` is so named because it is a *differential* file comparator — it does more than just report that there is a mismatch. Features of `diff` include:

- reports line changes, additions, and deletions necessary to bring two files into agreement; and,

- produces an editor script to convert one file into another — the generated editor script is intended for the `ed` text editor described above.

A variant of `diff` called `diff3` compares *two* new versions of a file against one old one. For further details see *Editing Text Files* and the `diff` and `diff3` manual pages.

Finding Commonality Between Files

The `comm` command identifies common lines in two sorted files. The output is displayed in three columns which show lines present in first file only, present in both, and/or present only in second. For further details see *Editing Text Files* and the `comm` manual page.

Checking Spelling

The `spell` command finds spelling errors by comparing each word in a document against a dictionary list that includes proper names. Features of `spell` include:

- handles common prefixes and suffixes; and
- collects words to help tailor local spelling lists.

For further details see *Editing Text Files* and the `spell` manual page.

Searching for Words in a Sorted File

The `look` command searches for words in a sorted file (usually a dictionary) that begin with a specified prefix. For further details see *Editing Text Files* and the `look` manual page.

Encrypting and Decrypting Files

The `crypt` commands encrypts and decrypts files for greater security. For further details see *Editing Text Files* and the `crypt` manual page.

Joining Records in File

The `join` commands combines two files by joining records with identical keys. For further details see *Editing Text Files* and the `join` manual page.

8.4. Summary of Text Processing Utilities

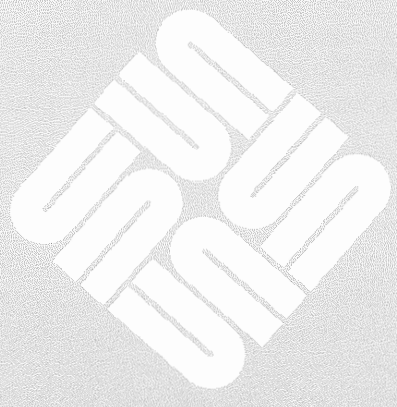
The following table is an alphabetical list of the text utilities described in this chapter.

Table 8-1 *Summary of Editing and Text Processing Programs*

Program Name	*Function*
`awk`	scan patterns and process text
`comm`	find commonality between files
`crypt`	encrypt
`diff`	display differences between files
`ed`	(very primitive) line editor for text files
`egrep`	scan for text patterns
`ex`	line oriented editor for text files
`fgrep`	scan for text patterns
`grep`	scan for text patterns
`join`	join records in file
`look`	search for words in a sorted file
`lpq`	display print queue
`lpr`	print files offline
`lprm`	remove jobs from print queue
`pr`	prepare files for printing
`sed`	batch or stream editing
`sort`	sort files
`spell`	check spelling
`tr`	translate characters
`uniq`	remove adjacent duplicate lines from sorted file
`vi`	visual editor for text files
`wc`	count characters, words, and lines in files

9

Formatting Documents



9

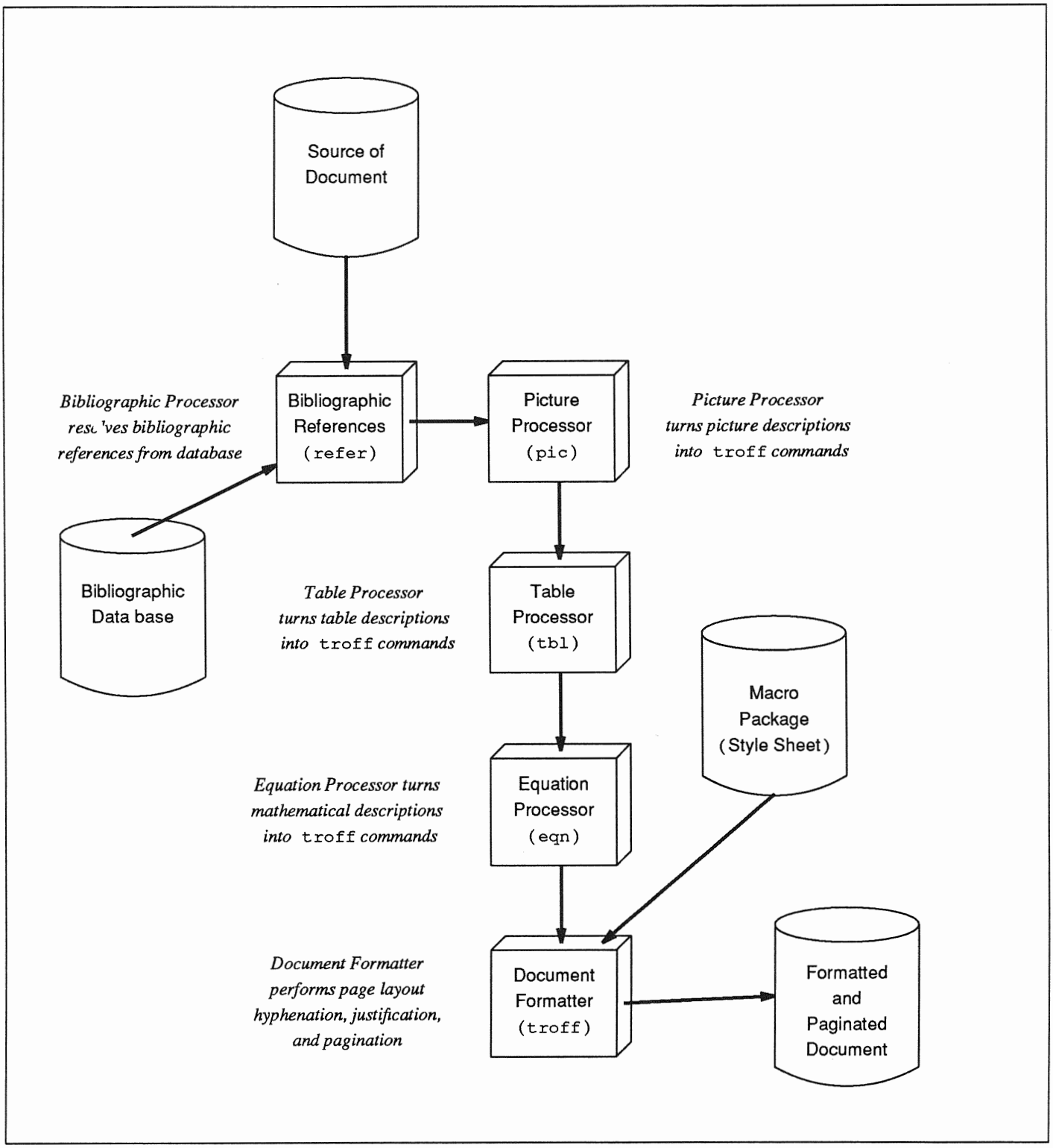
Formatting Documents

Numerous tools for producing technical documents are available with the SunOS operating system. The `textedit` mouse-based system of SunView is a basic text editing utility. Information on `textedit` is described in chapter 4, *Sunview — A Window Environment.* The major tool for producing complex documents is `troff` — a *text formatter* originally created for a second-generation phototypesetter called a C/A/T.

Since `troff` was created for older typesetting technology, it requires detailed instructions for laying out a document. Not only must users be familiar with the details of each `troff` request, but they must also be knowledgeable of typography terms and concepts. Numerous utilities and tools are now available to ease the use of `troff`.

Macro packages are perhaps the most important tool to help users in generating documents. A macro package is a type of *style guide* — users type in 'high-level' commands reminiscent of the structure of the document instead of hundreds of detailed `troff` requests. In addition to macro packages, various *preprocessors* produce mathematical equations, tables, and line drawings. The picture below conveys some of the extensive tools available for producing documents.

Figure 9-1 *Document Formatting Model with Macro Package*



9.1. Formatting Documents

The major document formatting programs available are `troff` and `nroff`. Although there are 'What You See Is What You Get' (WYSIWYG) document preparation packages available for the Sun Workstation, `troff` and `nroff` still are valuable for very difficult formatting applications applications where WYSIWYG systems don't, as yet, fit the bill.

Although `troff` originally was written for a specific second-generation photo-typesetter it can be used with appropriate conversion utilities to drive other types of devices. Both `troff` and `nroff` use the same input language and are capable of elaborate formatting feats when programmed appropriately. Features of these utilities are:

- completely definable page format keyed to dynamically planted 'interrupts' at specified lines;
- maintain several separately definable typesetting environments (for example, one for body text, one for footnotes, and one for unusually elaborate headings);
- arbitrary number of output pools can be combined at will;
- macros with substitutable arguments, and ability to use mid-line macros;
- computation and printing of numerical quantities;
- conditional execution of macros;
- tabular layout facility;
- positions expressible in inches, centimeters, ems, points, machine units or arithmetic combinations thereof;
- access to character-width computation for unusually difficult layout problems;
- overstrikes, built-up brackets, horizontal and vertical line drawing;
- dynamic relative or absolute positioning and size selection, globally or at the character level;
- can exploit the characteristics of the terminal being used, for approximating special characters, reverse motions, proportional spacing, etc.;
- typesetter has a vocabulary of several 102-character fonts (4 simultaneously) in 15 sizes; and,
- provides terminal output to view sample of final output;

For further details see *Formatting Documents* and *Using* `nroff` *and* `troff` and the `troff` and `nroff` manual pages.

9.2. Macro Packages

A macro facility is available to use with `troff`. Macros are a sequence of frequently-used `troff` requests collected together into named chunks. These named chunks (macros) are called by name to achieve standardized formatting actions.

A *macro package* can be considered the *style sheet* for a document. The user types in 'high-level' instructions to indicate the start of text constructs such as paragraphs, tables, and such, and the macro packages translates these instructions into the detailed `troff` requests needed to achieve the desired layout.

`-ms` Macro Package

Sun supports the `-ms` macro package — a standardized manuscript layout package of requests for use with `nroff` and `troff`. Features of the `-ms` macro package include:

- standard indented paragraphs, block-paragraphs, itemized paragraphs, quoted paragraphs;
- various forms of indented and non-indented displays, 'keep' displays, floating displays, tables, and displayed and numbered equations;
- automatically numbered headings;
- footnotes — automatically numbered or with user-defined symbols;
- multiple-column layout;
- standardized placing of page numbers;
- standardized running headers and footers — users can specify in detail the form of odd and even headers and footers; and,
- draft dates.

For further details see *Formatting Documents.*

`-man` Macro Package

The `-man` macro package is the second major macro package designed for formatting the on-line manual pages. The `-man` macro package is less complex, but based upon the `-ms` macro package. These macros are primarily used to format the 'man pages'. For further details see *Formatting Documents.*

9.3. Preprocessors

The major preprocessors for `troff` are `tbl`, used for describing tabular layouts, and `eqn` used for describing mathematical equations. Both tabular layout and mathematics are known as 'penalty copy' in the typesetting trade because of the need for a large number of finely detailed formatting requests.

Mathematical Typography

The mathematical typesetting preprocessor, `eqn`, translates easily readable formulas, either in-line or displayed, into detailed `troff` or `nroff` instructions.

Example of `eqn`

Equation formulas are written as if you were saying them aloud to someone. For example:

```
sigma sup 2 ~=~ 1 over N sum from i=1 to N ( x sub i - x bar ) sup 2
```

produces:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

There is a version of `eqn` for `nroff` named `neqn`. The `neqn` preprocessor accepts the same input language as `eqn` and prepares formulas for workstation or terminal display. `neqn` has the same facilities as `eqn` within the graphical capability of the workstation.

Features of `eqn` include:

- automatic calculation of size changes for subscripts, sub-subscripts, and for equations such as e^{x^2};
- full vocabulary of Greek letters and special symbols, such as 'gamma' for γ, 'GAMMA' for Γ, 'integral' for $\int$, and so on;
- automatic calculation of large bracket sizes;
- vertical 'piling' for matrices, conditional alternatives, etc.;
- integrals, sums, etc., with arbitrarily complex limits;
- diacriticals: dots, double dots, hats, bars, etc.; and,
- easily learned by nonprogrammers and mathematical typists.

For further details see *Formatting Documents* and the `eqn` manual page.

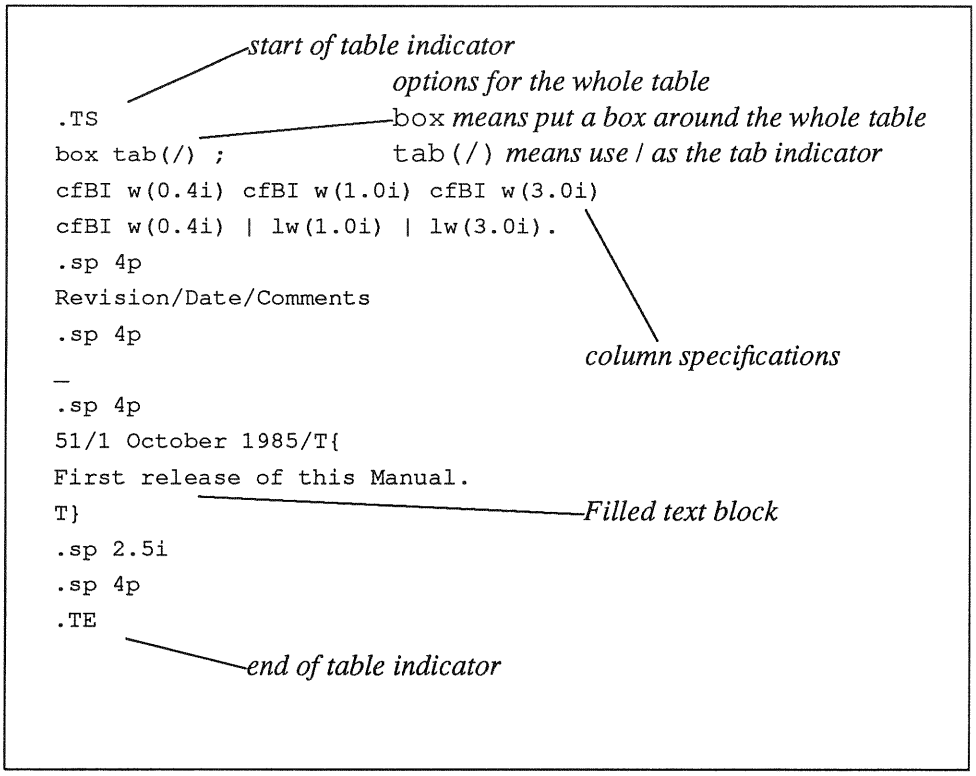
Laying Out Tables

The `tbl` preprocessor translates simple descriptions of table layouts and contents into detailed typesetting instructions for `nroff` and `troff`. Features of `tbl` include:

- computes column widths;
- handles left- and right-justified columns, centered columns and decimal-point alignment;
- places column titles;
- table entries can be text, which is adjusted to fit column width; and,
- can box all or parts of table.

Example of a Table

Here is a small example of the input for a table:




```
.TS
box tab(/) ;
cfBI w(0.4i) cfBI w(1.0i) cfBI w(3.0i)
cfBI w(0.4i) | lw(1.0i) | lw(3.0i).
.sp 4p
Revision/Date/Comments
.sp 4p
_
.sp 4p
51/1 October 1985/T{
First release of this Manual.
T}
.sp 2.5i
.sp 4p
.TE
```

The table below is the result from formatting the above table source.

Revision	*Date*	*Comments*
51	1 October 1985	First release of this Manual.

For further details see *Formatting Documents* and the `tbl` manual page.

Bibliographic References

The `refer` preprocessor is a bibliography system to support citations in documents. The `refer` system comprises a set of facilities for data entry, indexing, retrieval, sorting of a bibliographic database.

Supporting utilities make `refer` easier to use. They include:

- `addbib` which creates and extends the bibliographic database;
- `sortbib` which sorts the bibliographic database by author, date, or other criteria; and,
- `roffbib` which formats the entire bibliographic database as a bibliography or annotated bibliography.

For further details see *Formatting Documents* and the `addbib`, `sortbib`, `roffbib`, `indxbib`, and `lookbib` manual pages.

9.4. Other Document Formatting Tools

In addition to the major document formatting tools described above, there are a number of other minor supporting tools.

Checking Spelling Errors

The `spell` utility checks the spelling of a document against an on-line dictionary. For further details see *Formatting Documents* and the `spell` manual page.

Handling Reverse Paper Motions

The `col` utility converts files with reverse line feeds into canonical form for one-pass printing. `col` is used mainly with `nroff` for printers that can not do reverse paper motions. For further details see *Formatting Documents* and the `col` manual page.

Stripping `troff` Constructs

The `deroff` utility removes `troff` requests from a source file. The spelling checker facility, among others, uses `deroff` to remove all `troff` requests which would otherwise be displayed as misspelled words. For further details see *Formatting Documents* and the `deroff` manual page.

Checking Syntax

The `checknr` program checks a document for possible mismatched opening and closing delimiters and unknown `troff` requests. The complexity of `troff` requests and macro calls can sometimes lead to strange effects such as entire parts of a document disappearing into a black hole. Such problems are often caused by starting a display and forgeting to end it.

The `checkeq` program checks a document for correctly specified equations. For further details see *Formatting Documents* and the `checknr` and `checkeq` manual pages.

Generating a Permuted Index

The `ptx` command generates a permuted, or keyword-in-context, index from text files. For further details see *Formatting Documents* and the `ptx` manual page.

9.5. Summary of Document Formatting Utilities

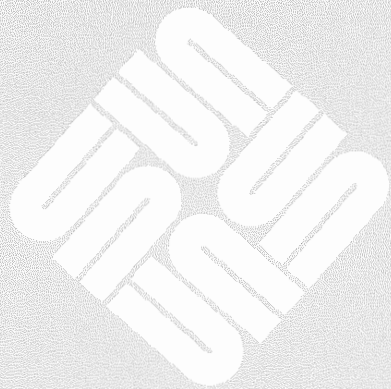
Below is an alphabetical list of the document formatting commands described in this chapter.

Table 9-1 *Summary of Document Formatting Programs*

Program Name	*Function*
`checkeq`	check that specified equations are correct
`checknr`	check `nroff` and `troff` constructs
`col`	filter out reverse paper motions
`deroff`	remove `nroff` and `troff` constructs
`eqn`	language for specifying mathematical equations
`-man`	macro package to format the on-line manual pages
`-ms`	popular macro package for producing documents
`nroff`	document formatter for typewriter-like devices
`ptx`	generate permuted (keyword in context) index
`refer`	bibliographic database processor
`spell`	check spelling
`tbl`	language for describing columnar layouts
`troff`	document formatter for typesetters or laser writers

10

Communication Facilities



10

Communication Facilities

Electronic mail and facilities for transferring files to and from remote machines are available on the SunOS operating system for:

- communicating between users on a single time-sharing machine;
- communicating between users on different machines in the local network;
- communicating between users on different host machines in geographically distributed locations; and,
- facilities for accessing remote machines.

The first two categories are considered roughly equivalent since the local network facilities, in many cases, makes such access transparent.

10.1. Local Communications Facilities

There are numerous ways to communicate with other users on the same host or on other host machines on the local network. Local communications utilities are described below.

Talking Directly with Another User

The `talk` command establishes direct workstation or terminal communication with other users on a different machine on the local network. The `mesg` command inhibits receipt of messages from the `talk` command. For further details see *Mail and Messages: Beginner's Guide* and the `talk`, and `mesg`, manual pages.

Sending and Receiving Mail

The `mail` command is an electronic mail system that sends messages to users on the same machine, or to users on other machines on the local network, and to remote machines (using the capabilities of `uucp` described below). Features of the `mail` command, include:

- sending messages to one or more users;
- reading and disposing of individual messages;
- announcing the presence of mail through the `login` program, and optionally by the `csh` program;
- saving messages in files or forwarding them; and,
- supporting items such as 'Subject:' and 'Cc:' fields.

Mail Tool is a window interface for the `mail` program. For further details see *Mail and Messages: Beginner's Guide*, *SunView 1 Beginner's Guide*, and the `mail` and `mailtool` manual pages.

10.2. Remote Communication Facilities

There are ways to access remote machines (machines outside the local network) for transferring files. Facilities for remote communication are described below.

Directly Accessing Remote Machines

The `tip` command establishes a full-duplex connection for logging onto remote computers. Features of `tip` include:

- a transparent interface to the remote machine;
- transmitting files; and,
- taking remote input from local file or putting remote output into local file.

For further details see *Using the Network: Beginner's Guide* and the `tip` manual page.

Transferring Data Between UNIX Systems

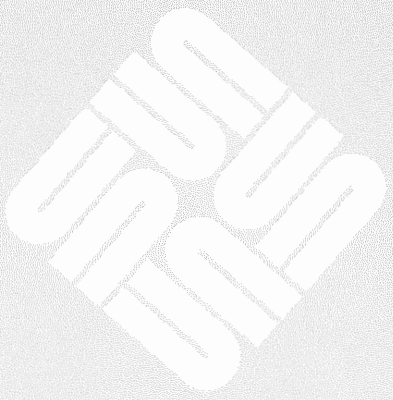
The `uucp` command allows spooled file transfers between machines. Features of `uucp` include:

- automatic queuing until line is available and remote machine is responding; and,
- copying between remote machines.

In general, the `uucp` facilities are not used directly by users but service programs such as `mail`. For further details see *System and Network Administration* and the `uucp` manual page.

11

SunOS — Internal Features



11

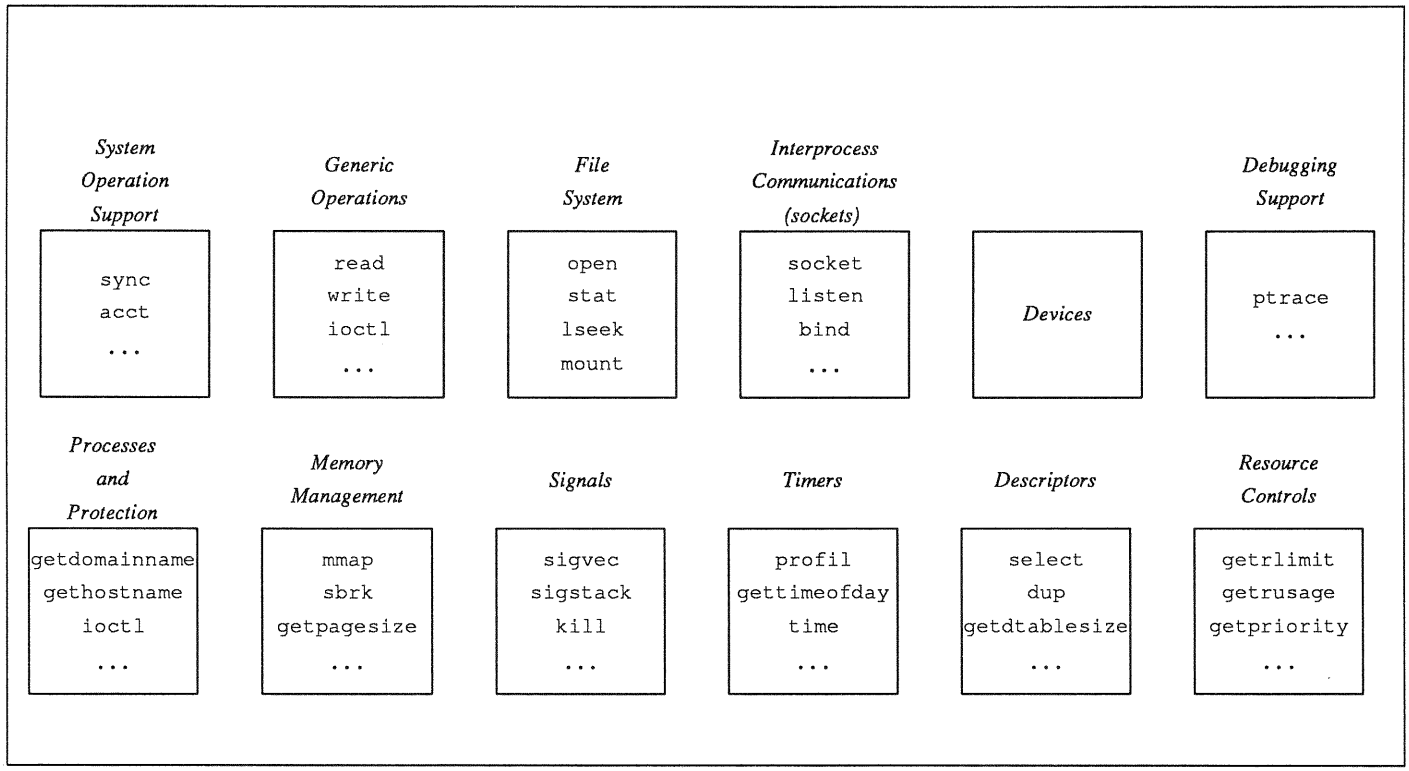
SunOS — Internal Features

The SunOS operating system is based on the converged Berkeley 4.2BSD and AT&T System V. It is enhanced to provide high-performance facilities for all software packages and to maximize system throughput for workstations on heterogeneous networks. In this chapter the major features of the SunOS operating system are presented and reference pointers are included for more information.

At the bottom layer of the operating system is the *kernel* — a collection of system services that manage the resources of the system on behalf of application programs. Applications can either access the kernel's primitive functions directly, or make use of library functions that in turn use the kernel's primitive functions. The SunOS kernel supports a virtual memory management scheme that promotes system resource sharing and portability across diverse hardware platforms.

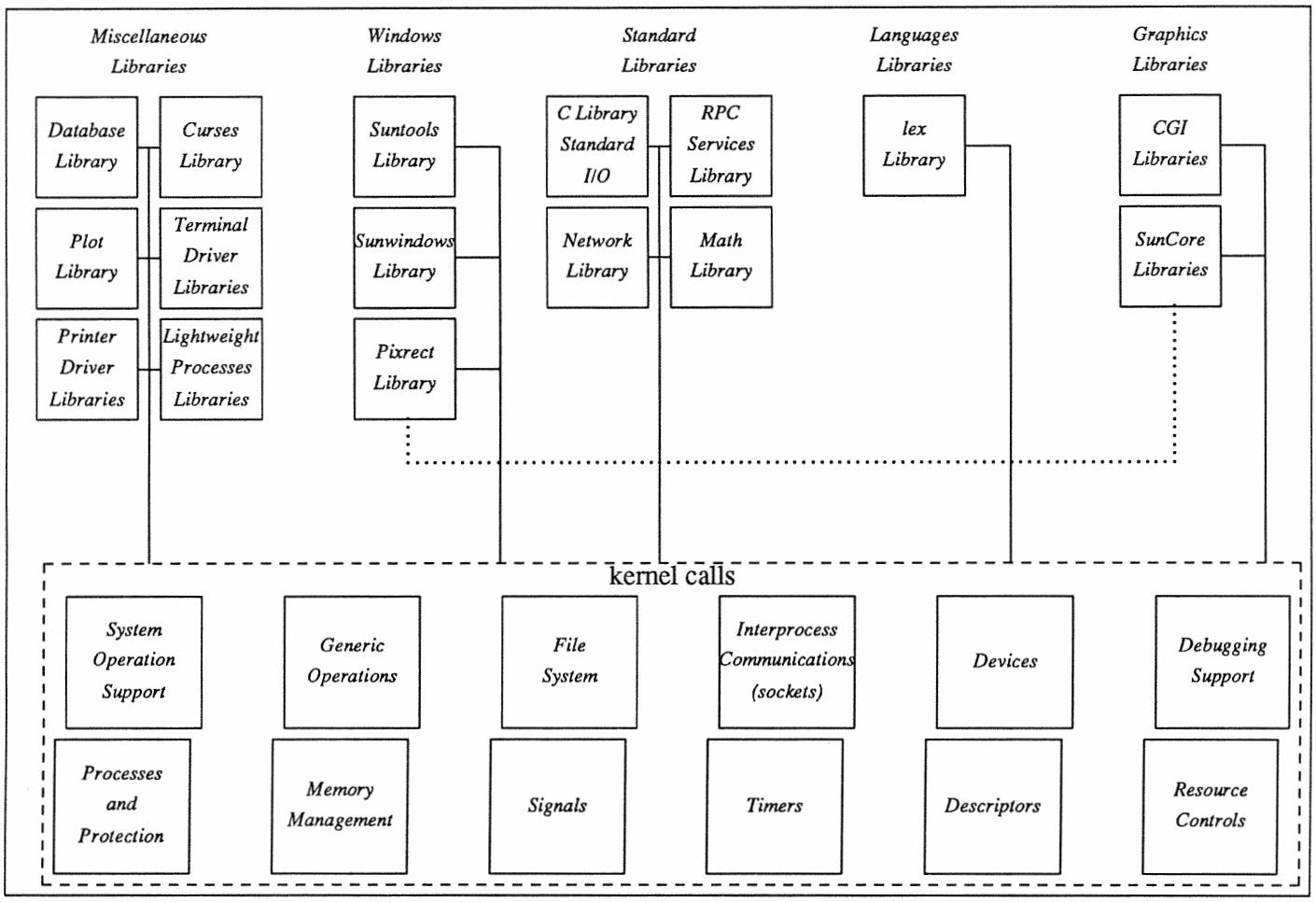
The illustration below shows the kernel primitive functions divided into functional groups. For more information on kernel functions see *System Services Overview* and *SunOS Reference Manual.*

Figure 11-1 *Kernel Primitive Functions*



As previously stated, most applications use one or more standard *library packages* that in turn call the kernel. The main groups of library packages are illustrated in the figure below.

Figure 11-2 *Standard Library Packages*



To find out more about these libraries consult the following manuals:

Library	*Described in Manual(s)*
C Library	*SunOS Reference Manual*
Standard I/O Library	*Programming Utilities and Libraries* *SunOS Reference Manual*
Network Library	*Network Programming* *SunOS Reference Manual*
Remote Procedure Call Library (RPC)	*Network Programming* *SunOS Reference Manual*
Math Library	*SunOS Reference Manual*
`lex` Library	*Programming Utilities and Libraries*
CGI Libraries	*SunCGI Reference Manual*
SunCore Libraries	*SunCore Reference Manual*
Suntools Library	*SunView 1 System Programmer's Guide*
Sunwindows Library	*SunView 1 System Programmer's Guide*
Pixrect Library	*Pixrect Reference Manual*
Database Library	*SunOS Reference Manual*
Curses Library	*SunOS Reference Manual*
Terminal Driver Libraries	*SunOS Reference Manual*
Plot Library	*SunOS Reference Manual*
Lightweight Processes Library	*Programming Utilities and Libraries* *System Services Overview*

11.1. Major Features of SunOS

Other than the kernel primitives and standard libraries, major features of the SunOS operating system include:

- shared libraries;
- lightweight process library;
- device-independent I/O and redirection;
- virtual memory;
- job control facilities; and,
- standards.

Each of these features is described below and reference pointers are included for more information.

Shared Libraries and Dynamic Linking

The shared libraries system enables processes running different programs to share libraries common to each other. The shared libraries system is simple and flexible and improves overall system performance. Features of shared libraries include:

- minimizes kernel support;
- does not require that shared libraries be used;
- designed to minimize burden placed on users of exiting code; and
- improves the development environment.

Dynamic linking allows you to develop your own shared libraries. For more information on shared libraries see *Programming Utilities and Libraries.*

Lightweight Process Library

The *lightweight process library* provides primitives for manipulating threads of control, as well as, for managing events such as interrupts and traps. Lightweight processes are good for implementing service processes which must maintain state for multiple connections, and for programs which manage asynchrony. At present, there is no kernel support for lightweight processes, so concurrent system calls must be implemented by forked UNIX processes.

Features of lightweight processes include:

- thread creation, destruction, status gathering, scheduling manipulation, suspend, and resume;
- multiplexing the clock (any number of threads can sleep concurrently);
- individualized context switching;
- monitors and conditions variables to synchronize threads;
- extends rendezvous (message send-receive-reply) for communication between threads;
- an exception handling facility that provides both notify and escape exceptions;
- a way to map interrupts into extended rendezvous;
- a way to map traps into exceptions; and,
- utilities to allocate red-zone-protected stacks, and to provide some stack integrity checking for environments which lack sophisticated memory management.

For more information on the lightweight process library see *System Services Overview.*

Device-independent I/O and Redirection

Highly efficient buffered stream I/O is integrated with formatted input and output.

Virtual Memory

The virtual memory management scheme promotes system resource sharing and portability across diverse hardware platforms. The VM system accommodates page-by-page sharing and employs a copy-on-write mechanism to create

individual copies of pages when necessary. For more information see *System Services Overview.*

Job Control Facilities

Job control provides support for multiplexing terminals between jobs, running several jobs at once, some in the background and others in the foreground and moving running jobs from background to foreground and vice-versa. These facilities are provided with the C shell command interpreter described in chapter 5, *Using the Shells* and the `csh` manual page.

Standards

The SunOS operating system incorporates the System V Release 3 Base Level interface, reflecting System V and BSD convergence. Features from this convergence include:

- all base level system calls including and library routines;
- complete System V STREAMS interface. STREAMS supports portable communication protocol modules and simplifies writing device drivers;
- full System V and BSD compatible tty interface using STREAMS;
- System V compatible archive utility;
- System V batch utility and job scheduler;
- access to Sun libraries (such as SunView) from System V programs; and,
- easy access to System V library routines.

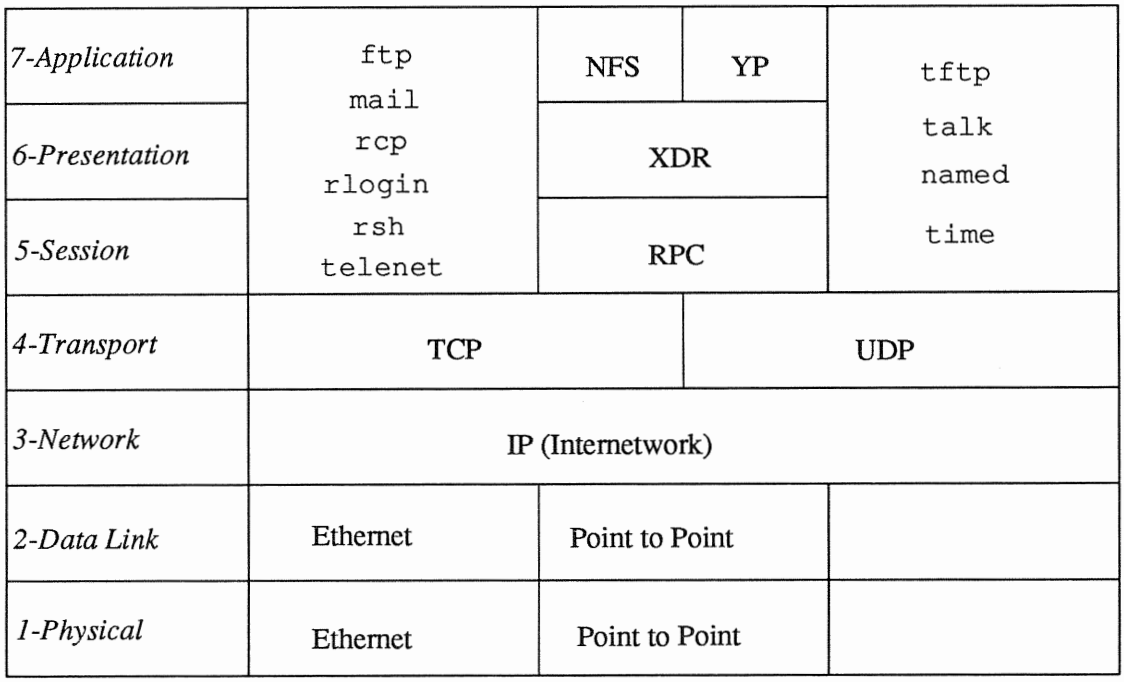
For more information see *System Services Overview.*

11.2. Networking Facilities

The SunOS operating system includes an ISO-OSI model local networking subsystem. Fully supported is the DARPA internet family of protocols and associated addressing. The datagram (UDP) and stream (TCP) protocols are supported, as well as the error message protocol (ICMP) and packet forwarding at the internet layer (IP). A routing information protocol allows hosts to determine the shortest route to a destination within the local network.

The OSI model is a *layered* mode, as shown in the diagram below.

Figure 11-3 *Network Architecture*



Remote Procedure Call

The Remote Procedure Call (RPC) facility provides a mechanism whereby one process (the *caller* process) can have another process (the *server* process) execute a procedure call, as if the caller process had executed the procedure call in its own address space (as in the local model of a procedure call). Because the caller and the server are now two separate processes, they no longer have to live on the same physical machine.

The RPC mechanism is implemented as a library of procedures, plus a specification for portable data transmission, known as the eXternal Data Representation (XDR). Together RPC and XDR provide a kind of standard I/O library for interprocess communication. Thus programmers now have a standardized access to sockets without having to be concerned with low-level details of socket-based IPC.

The details of programming applications to use Remote Procedure Calls can be overwhelming. To aid in this task is `rpcgen`, a compiler, which helps you write RPC applications simply and directly. The `rpcgen` compiler accepts a remote program interface definition written in a language, called RPC language, which is similar to C. For further information see *Network Programming* and the `rpcgen` manual page.

Interprocess Communication

The network (interprocess) communications facilities derived from 4.2BSD are based on the *socket*. Network communication uses standard protocols and features of network communication include:

- interprocess communications integrated into UNIX;
- user access to interprocess and network communication through sockets;
- arbitrary processes in the system may communicate in either a message or stream oriented fashion; and,
- communications through the socket mechanisms provide remote logins, copies, and shells over the local network.

The Remote Procedure Call (RPC) services provide an easier to use layer of abstraction than does the socket mechanisms. For more information on interprocess communication, see *Network Programming* and *SunOS Reference Manual.*

STREAMS

STREAMS is a general, flexible facility and a set of tools for developing system communication services. It is intended to augment the existing character I/O mechanism, and thus support the evolution of system communications facilities — ranging from complete networking protocol suites to individual device drivers. STREAMS defines standard interfaces for character I/O within the kernel, and between the kernel and the rest of the system. For further information see *System Services Overview.*

Reference Documentation

To use the networking facilities refer to one or more of these documents that are a part of the general manual named *Network Programming*. This manual is divided into three parts.

PART ONE, focuses on Sun's network programming mechanisms. It includes:

- The *Network Services* overview, which introduces the fundamental network services without dealing with any protocol or implementation related issues.
- The `rpcgen` *Programming Guide*, which introduces the `rpcgen` protocol compiler and the C-like language that it uses to specify RPC applications and define network data. In almost all cases, `rpcgen` will allow network applications developers to avoid the use of lower-level RPC mechanisms.
- The *Remote Procedure Call Programming Guide*, is intended for programmers who want to understand the lower-level RPC mechanisms. Readers are assumed to be familiar with the C language and to have a working knowledge of network theory.
- The *External Data Representation: Sun Technical Notes*, introduces XDR and explains the justification for its "canonical" approach to network data interchange. This section also gives Sun implementation information and a few examples of advanced XDR usage.

PART TWO includes a number of protocol specifications. One of these, the *External Data Representation Protocol Specification*, has been accepted (as of the date of this printing) as an ARPA RFC (Request for Comments). These protocol specifications include:

- The *External Data Representation Protocol Specification*, which includes a complete specification of XDR data types, a discussion of the XDR approach and a number of examples of XDR usage.

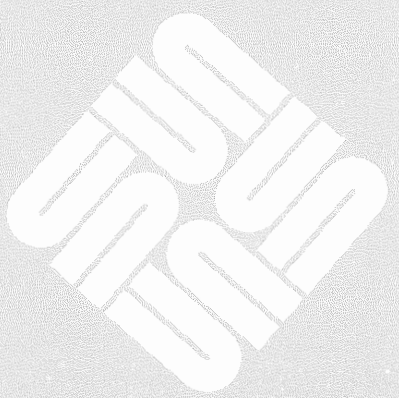
- The *Remote Procedure Call Protocol Specification*, which includes a discussion of the RPC model, a detailed treatment of the RPC authentication facilities and a complete specification of the Portmapper Protocol.
- The *Network File System: Version 2 Protocol Specification*, which includes a complete specification of the Mount Protocol, as well as the NFS specification itself.

PART THREE documents Berkeley style, socket-Based Inter-Process Communications. It includes:

- *A Socket-Based Interprocess Communications Tutorial*, which assumes little more that basic networking concepts and introduces socket-based IP and includes many examples.
- *An Advanced Socket-Based Interprocess Communications Tutorial*, which takes up where the *Tutorial* leaves off.
- *Berkeley-Style IPC Implementation Notes*, which describes the low-level networking primitives (e.g. `accept()`, `bind()` and `select()`) which originated with the 4.2BSD UNIX system.

12

System Administration



12

System Administration

System administration involves a collection of tasks for installing new versions of the SunOS operating system, maintaining hardware and software, keeping files and machines secure from unauthorized access, troubleshooting and upgrading the system. This chapter presents an overview of the utilities used for doing these tasks.

System administration is divided into the following areas:

- initially setting up the hardware and installing the software;
- managing the system on a daily basis — adding new user accounts;
- fixing hardware and software when they break;
- rebooting the system;
- adding printers and terminals and making the system aware of their existence;
- setting up system security; and,
- setting up and administering the NFS file service and, if applicable, the Yellow Pages maps.

In general, you must be the superuser to manage the system and to perform any of the tasks listed above. The superuser (also called *root*) has powers to override other users' permissions and has access to all files and directories on the system. The superuser, if out of control, can wreak much havoc on the system.

Manuals to Read

When you receive your system you should read the installation manual that came with your system to find out how to unpack and set up the hardware.

To install the SunOS operating system and to configure the system using the `suninstall` utility make sure you read *Installing the SunOS*.

Finally, the *System and Network Administration* manual contains information and folklore that you need to manage the system.

The manual pages for System Administration tasks can be found in the *SunOS Reference Manual.*

12.1. Setup and Installation

Initial setup and installation of a Sun system is aided by a special program called `suninstall`. The `suninstall` utility is a menu and mouse based configuration program to help you create the correct configuration for your system.

Configuring the System

Your system is delivered with a *generic* kernel — that is, the kernel is configured to drive every kind of device that Sun supports. The code for the device drivers and their data structures occupies real memory. When you install your system, you can reconfigure the kernel to include only those drivers for your particular system. The `config` utility creates the necessary files and commands to configure the kernel. The procedure for reconfiguring the kernel is well described in *Installing the SunOS*, and for more information see the `config` and `suninstall` manual pages.

12.2. Startup and Shutdown Procedures

The startup and shutdown procedures automate the boot procedures as much as possible. For example, there are automatic boot procedures to bring up SunOS and automatic reboot and file consistency checks and repairs in the event of a system crash. The system normally reboots itself when the power is turned on. However, the superuser can shut down the system and reboot it at will.

Halting and Rebooting Utilities

The following utilities are used for halting and rebooting the system.

The `halt` utility halts the system and returns control to the PROM monitor.

The `shutdown` utility informs users that the system is coming down, and then proceeds to shut the system down at the time you specify.

The `reboot` utility performs an automatic reboot.

The `sync` utility forces all outstanding input/output on the system to complete — used to shut down gracefully.

For further information see *System and Network Administration*, and the `halt`, `shutdown`, `reboot`, and `sync` manual pages.

Checking File Systems

When a system has crashed, or when you are rebooting the `fsck` utility checks and repairs file system. It is an interactive repair program usually called automatically during the boot procedure. Features of `fsck` include:

- displays gross statistics: number of files, number of directories, number of special files, space used, and free space;
- reports duplicate use of space;
- retrieves lost space;
- reports inaccessible files;
- checks consistency of directories; and,
- lists names of all files.

For further information see *System and Network Administration* and the `fsck` manual page.

12.3. Day to Day Administration Tasks

Your Sun system must be administered on an daily basis. Some of the tasks you may need to do are described below.

Becoming Superuser

The `su` command temporarily changes your user ID to superuser empowering you with all system rights and privileges. Superuser access is the same as `root` access. You can log in as `root` from scratch, or log in as yourself and then become superuser using `su`. However you become superuser, you need to know the `root` password. For more information see *System and Network Administration* and the `su` manual page.

Changing Ownership of a File

The `chown` command changes the ownership of one or more files. At times you may need to give a file to someone else other than the original owner. This is usually true when you create a new account on the system. For more information see the *Doing More with SunOS: Beginner's Guide*, *System and Network Administration*, and the `chown` manual page.

Scheduling Events

The `cron` utility schedules regular actions at specified times. Features of `cron` include:

- actions are arbitrary programs; and,
- times are conjunctions of month, day of month, day of week, hour and minute — ranges may be specified for each.

For more information see *System and Network Administration* and the `cron` manual page.

Mounting and Unmounting File Systems

The `mount` command attaches a file system obtained from a local device or from a network to a machine's tree of directories.

The `umount` command removes the file system accessed from a device or from the network from a machine's tree of directories. The `umount` command also protects against removing a busy device. For more information see the *System and Network Administration* and the `mount` and `umount` manual pages.

Creating New File Systems

The `newfs` command creates a new file system on a device. Actually, `newfs` is a front end to the `mkfs` command, but `newfs` does much more of the hard work for you.

The `mkfs` command makes a new file system on a device. For more information see the *System and Network Administration* and the `newfs` and `mkfs` manual pages.

The Automounter

The `automount` command lessens the need for manual mounting remote file systems. The mounting takes place invisibly when you refer to the remote filesystem with `automount`. Remote machines and filesystems are located by `automount` either by using a file on your machine or by using a Yellow Pages map. For more information see *Using the Network: Beginner's Guide* and the `automount` manual page.

Creating Special Files

The `mknod` command makes a node (file system entry) for a special file. Special files are physical devices, virtual devices, physical memory, etc. Special files generally reside in the `/dev` directory. For more information see the *System and Network Administration* and the `mknod` manual page.

Dumping and Restoring Files

The `dump` command backs up the file system stored on a specified device, selectively by date, or indiscriminately.

The `restore` command restores a backed up file system, or selectively retrieves parts thereof. The `restore` command also has an option for restoring backed up file system interactively. For more information see the *System and Network Administration* and the `dump` and `restore` manual pages.

The `tar` command manages file archives on magnetic tape. This command is useful for creating tapes of files to transfer between different locations. For more information see the `tar` manual page.

12.4. Network Administration

Along with daily system administration tasks there are also network administration tasks which you will need to do. Basic network administration tasks include:

- On a server machine you must set up the hardware and the files so that clients can mount files from the server.
- Set up the server so it broadcasts to the network which files are available for clients to access.
- And finally, set up files on client machines that enable users to mount files on the server.

For more information on network administration see *System and Network Administration.*

12.5. System Security

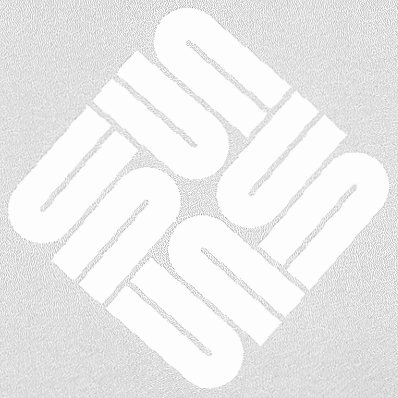
System security has become an important issue and is essential for preserving data privacy and integrity. The SunOS operating system provides the following security features:

- an auditing facility which checks valid user ID and valid passwords, file permissions, and administrative actions;
- file encryption;
- network security — an option to mount secure filesystems requiring DES authentication of user and host; and,
- an install-time option to run systems at a moderately high level of security, patterned after the widely accepted C2 classification.

For more information see *Security Features Guide.*

13

Software Development



13

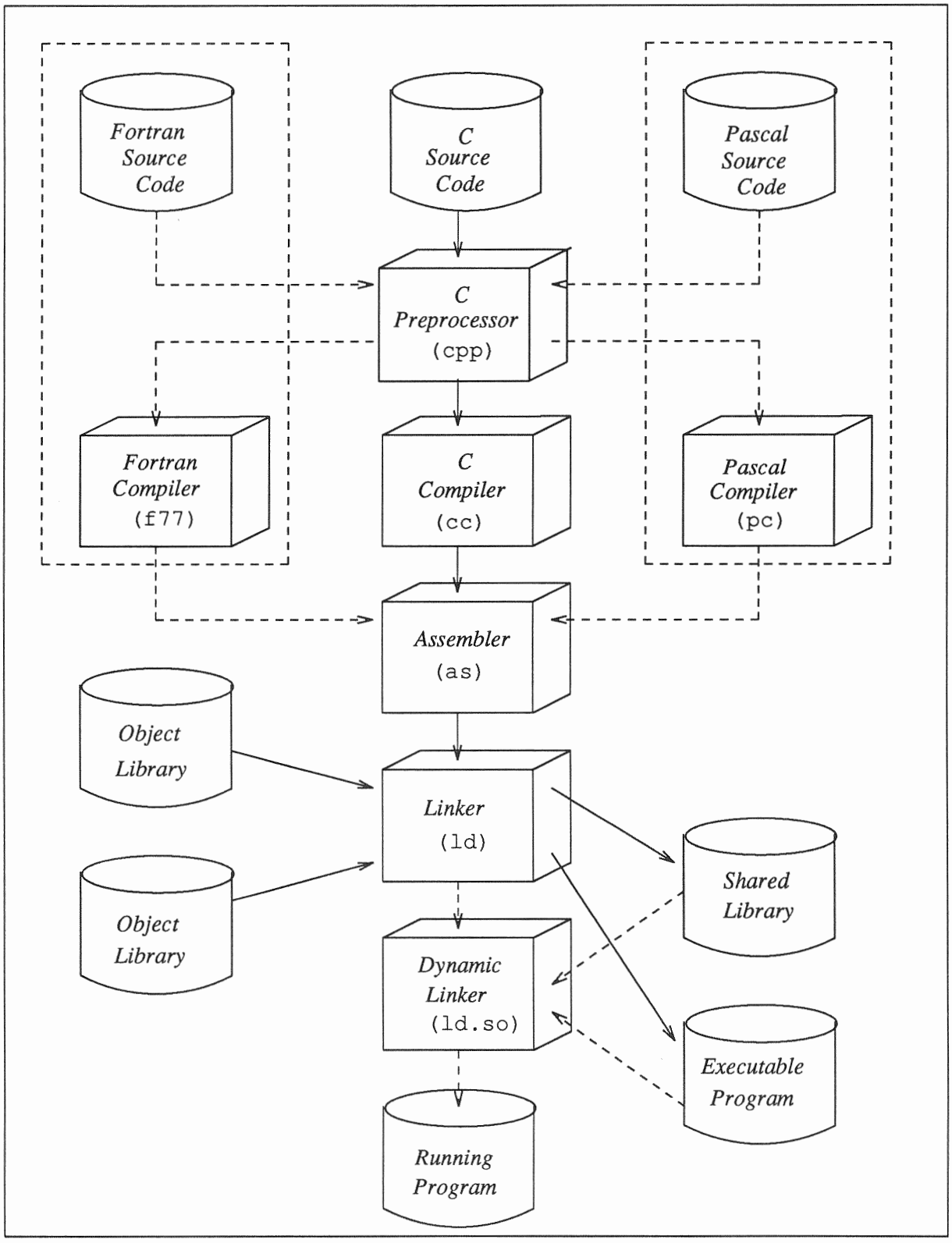
Software Development

The SunOS operating system supports the C and assembler languages. The C compiler includes Sun-4 code generators which take full advantage of Sun's RISC architecture SPARC . A number of software development tools are also support and this chapter presents an overview of the C language and tools for the trade.

Other languages, such as Fortran, Modula-2, Lisp, and Pascal are available as unbundled products. See chapter 15, *Unbundled and Third Party Software* for more information.

The diagram that follow shows the flow for source code to a compiled program. Since Pascal and Fortran are optional languages, but their routines can be used in C programs, they appear as optional features (enclosed in a dotted box).

Figure 13-1 *Flow from Source Code to Compiled Program*



13.1. C Programming Language

The SunOS operating system and most of the system utilities are written in C. For a description of C, read *The C Programming Language*, by Brian W. Kernighan and Dennis M. Ritchie, Prentice-Hall, 1978.

C is a general purpose language designed for structured programming. Features of the language include:

- generalized initialization, block structure, long integers, unions, and explicit type conversions;
- supports arbitrary length variable names;
- supports definable data types, which include character, integer, float, double, enumeration types, pointers to all types, functions returning above types, arrays of all types, structures and unions of all types;
- operations to give machine-independent control of full machine facility, including to-memory operations and pointer arithmetic;
- macro preprocessor for parameterized code and inclusion of standard files
- all procedures recursive, with parameters by value;
- machine-independent pointer manipulation;
- object code uses full addressing capability of the Sun workstation; and,
- runtime library gives access to all system facilities.

Compiling C Programs

The `cc` command is the C compiler which compiles and/or link edits programs in the C language. The C compiler has been enhanced to generate the position independent code (pic) used to build shared libraries. For more information see *C Programmer's Guide* and the `cc` manual page.

Symbolic Definitions and Conditional Compilation

The `cpp` program is the C preprocessor, which is a component of the C compiler. Features of `cpp` include:

- defining symbolic names;
- defining macros;
- conditional compilation; and,
- can also be used with FORTRAN.

For more information see the `cpp` manual page.

Checking Validity of C Programs

The `lint` command is a verifier for C programs. By itself the C compiler tends to be somewhat forgiving of programming styles which would give rise to compiler error messages in (say) Pascal. `lint` reports questionable or nonportable usage such as mismatched data declarations and procedure interfaces, nonportable type conversions, unused variables, unreachable code, no-effect operations, mistyped pointers, and obsolete syntax. It can also do full cross-module checking of separately compiled programs, and can check for the correct use of library functions. For more information see *Programming Utilities and Libraries* and the `lint` manual page.

Formatting C Programs

The `indent` utility is a formatter for arranging C source code into standard styles. Features of `indent` include:

- several different styles are available for placement of comments, arrangement of declarations, compound-statement braces, and case labels;
- can format a program suitable for processing by `troff`; and,
- can use a *profile* with your own options set.

For more information see the `indent` manual page.

13.2. Assembler

The `as` utility is the machine-level assembler for the Sun hardware family. Basic features of `as` include:

- creates object program normally consisting of read-only and sharable code, initialized data or read-write code, uninitialized data;
- relocatable object code is directly executable without further transformation;
- object code normally includes a symbol table; and,
- 'conditional jump' instructions become branches or branches plus jumps depending on distance.

For more information see *Assembly Language Reference* and the `as` manual page.

13.3. Linker

The `ld` utility is the link editor. Features of `ld` include:

- combine relocatable object files;
- insert required routines from specified libraries; and,
- resulting code is sharable by default.

For more information see the `ld` manual page.

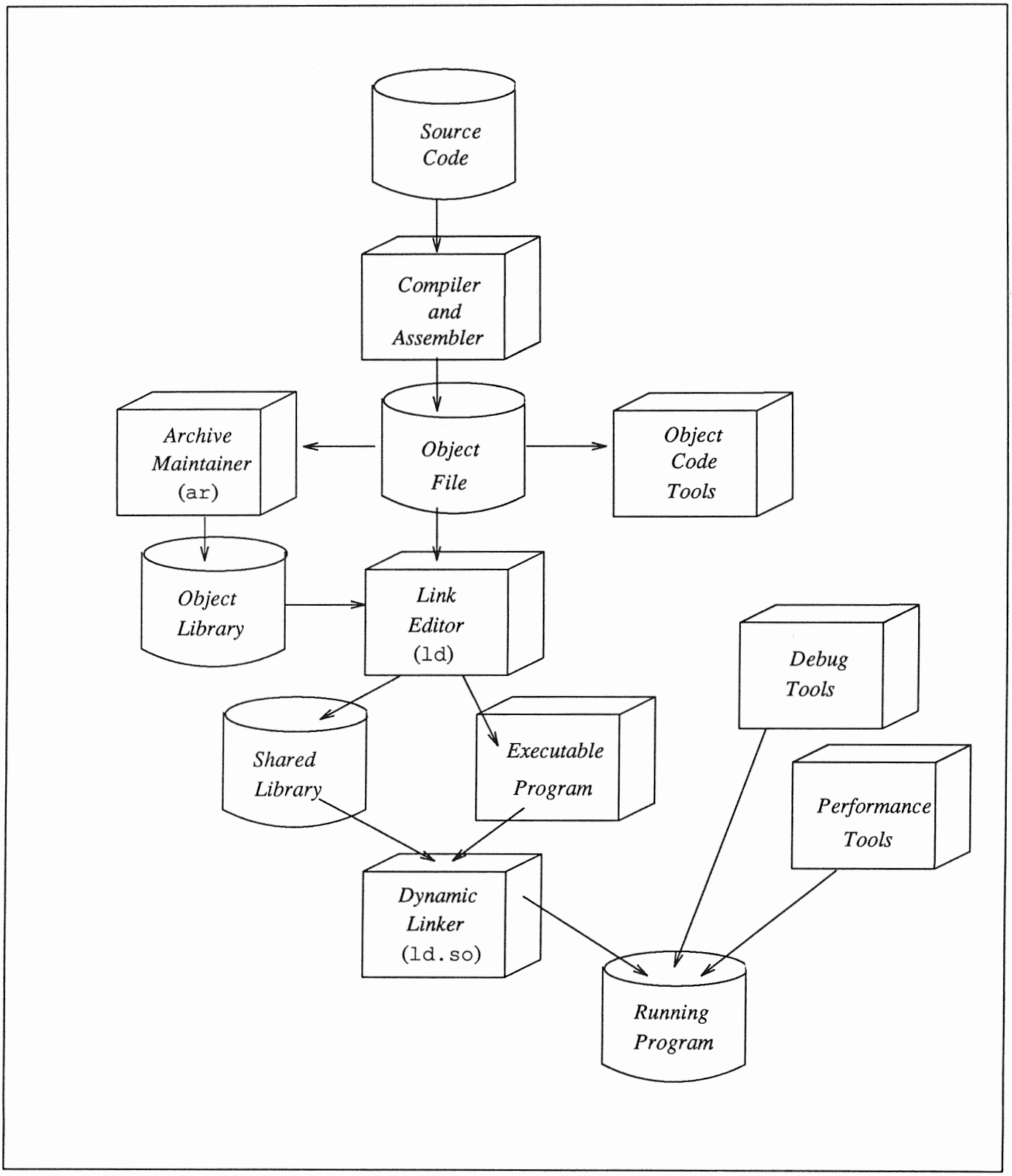
From its inception, the UNIX system has been extremely strong in supporting software development. The system grew within a group of computer scientists pursuing research in computer science. Sun Microsystems continues to add to the quality of the programming and programming language support tools.

13.4. Programming Tools to Work With Object Code

The SunOS operating system has very strong software development support and Sun Miscrosystems continues to add to the quality of the programming and programming language support tools.

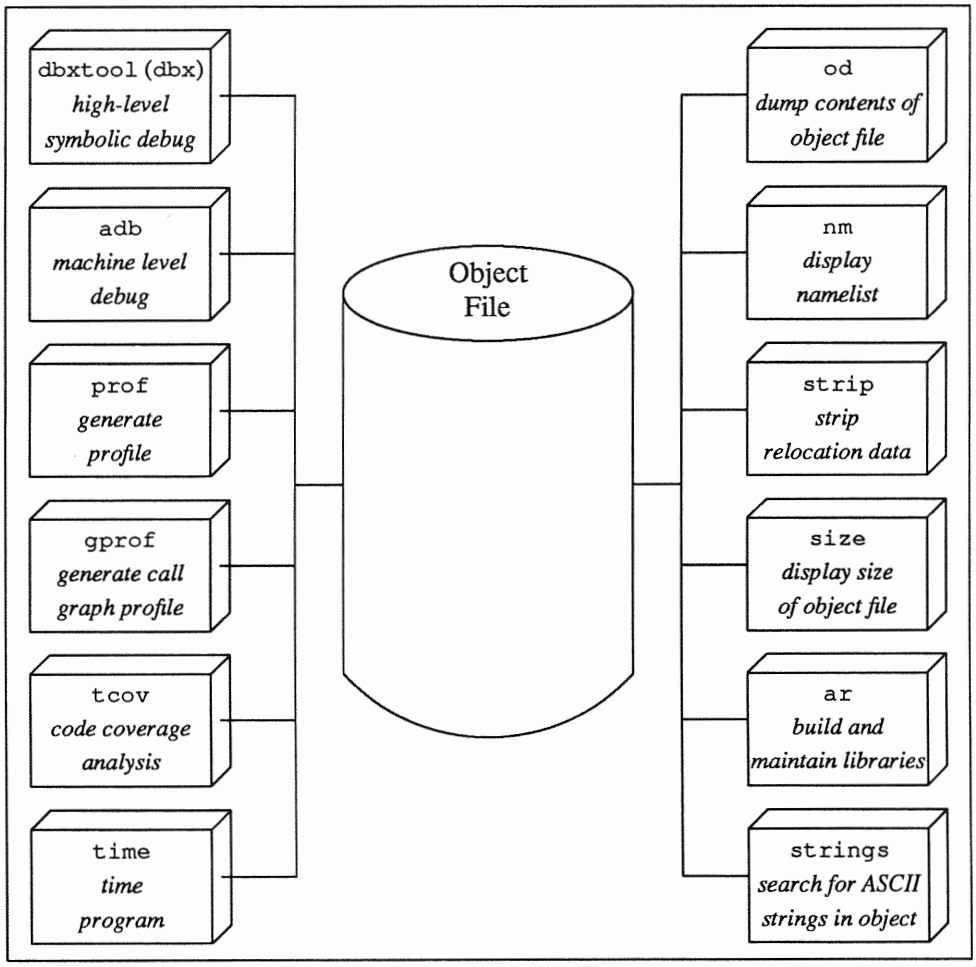
The diagram below shows the flow from source code to object code, plus the major groups of tools that can be used with the object code. A later diagram shows more detail.

Figure 13-2 *Major Object Code Programming Tools*



The following diagram shows in more detail commonly used tools to work with object code. Each tool is then briefly described and a reference pointer is included for more information.

Figure 13-3 *Commonly Used Tools to Work with Object Code*



Debugging Programs at the Source Level

The `dbxtool` and `dbx` utilities are part and parcel of the same debugging capability. `dbxtool` is a window-based source level debugger for `dbx`. The `dbx` utility is a source level debugger for programs written in C, FORTRAN 77, or Pascal, or any combination of them.

Main features and most commonly used commands of `dbx` include:

- multiple source language debugging;
- can display a stack back trace to show where a program stopped;
- can stop *at* specific lines in the source file, stop *in* specific functions, or can stop *when* specific events (such as a variable becoming equal to a designated value) occur;

- display the value of variables by name — displaying can be indirect through pointers;
- execute designated commands when specific conditions become true;
- tracing facility available;
- can debug arbitrary processes;
- can debug multiple processes; and,
- can be used to debug the kernel.

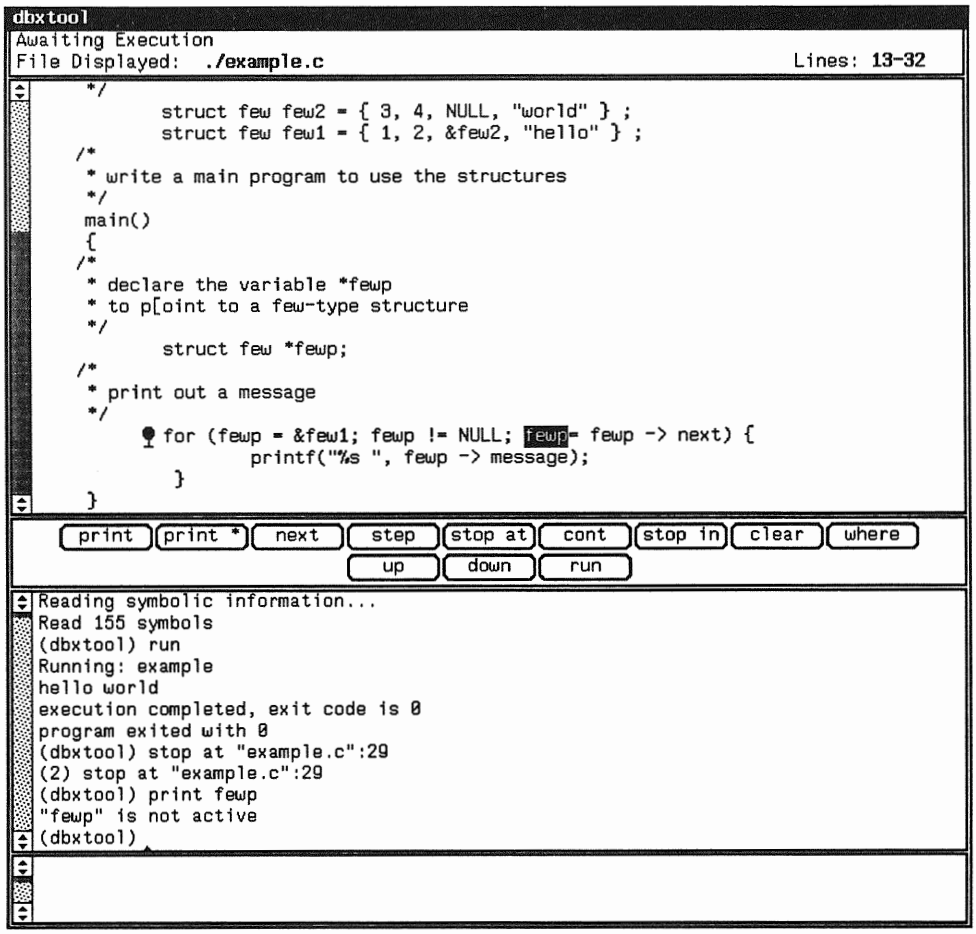
The `dbxtool` window and mouse-based interface to `dbx` includes all the features of `dbx`. However, you can do the most common operations (print, next, step, stop at, stop in, cont, and redo) by 'pushing a button' in the control panel.

The `dbxtool` window has five areas:

- *status window* displays the file and line number range of the code in the source window and information about the current state of the debugging process;
- *source window* usually displays the current focus of execution, though you can move it to any part of a source file (or to any other file);
- *menu of command buttons* contains the commands that can be constructed with the mouse;
- *command dialogue window* provides an area where you can type commands and where the commands obtained from the buttons window are echoed; and,
- *variable values display window* (generally called the "display window") displays the values of selected variables and expressions whenever execution halts.

In addition to the standard 'buttons' in the control panel, you can construct your own buttons, either as you go, or in a `dbxtool` profile. The picture below shows a `dbxtool` window with a program being operated on.

Figure 13-4 *Example* `dbxtool` *Window*



For more information see *Debugging Tools* and the `dbxtool` and `dbx` manual pages.

Debugging at the Machine Level

The `adb` utility is a very low-level symbolic debugger. It is now largely superseded by `dbxtool` and `dbx`. Some of `adb`'s features include:

- examine arbitrary files with no limit on size;
- interactive breakpoint debugging with the debugger as a separate process;
- symbolic reference to global variables;
- patching;
- stack trace for C programs;
- Output formats of: 1-, 2-, or 4-byte integers in octal, decimal, or hexadecimal, single and double floating point, character and string; and
- disassembled machine instructions.

For more information see *Debugging Tools* and the `adb` manual page.

Building and Maintaining Libraries

The `ar` utility is a library maintenance utility. The principal use of `ar` is to build and to maintain object code libraries used by `ld`, the link editor. `ar` can be used as a general utility for collecting groups of files into a single unit. Major features of `ar` include:

- maintain archives and libraries;
- combine several files into one for housekeeping efficiency;
- create new archive;
- update archive by date;
- replace or delete files from the archive;
- display table of contents (what files are in the archive); and,
- retrieve files from archive.

For more information see the `ar` manual page.

Dumping File Contents

The `od` command dumps the contents of any file. Output options include; any combination of octal or decimal by words, octal by bytes, ASCII, opcodes, and hexadecimal. The range of dumping also is controllable. For more information the `od` manual page.

Displaying the Namelist

The `nm` utility displays the namelist (symbol table) of an object program, and provides control over the style and order of names that are printed. For more information see the `nm` manual page.

Displaying Size of a Program

The `size` utility displays the memory requirements of one or more object files. For more information see the `size` manual page.

Stripping Relocation and Symbol Table

The `strip` utility removes the relocation and symbol table information from an object file to save space. For more information see the `strip` manual page.

Search for ASCII Strings in Binary File

The `strings` utility is a useful tool to locate ASCII strings in a binary file. For more information see the `strings` manual page.

13.5. Performance Analysis Tools

The SunOS operating system supports several facilities for monitoring performance of software, ranging from a simple command to reporting the time a program takes to execute, to a code-coverage tool providing detailed statement-by-statement analysis of a program.

Timing a Program

The `time` utility is a simple system command that produces a report on how much time a given program takes to execute. For more information see *Programming Utilities and Libraries* and the `time` and `/usr/bin/time` manual pages.

Profiling a Program

The `prof` utility constructs a profile of time spent per routine from statistics gathered by time-sampling the execution of a program. `prof` also displays subroutine call frequency and average times for C programs. For more information

see *Programming Utilities and Libraries* and the `prof` manual page.

Generating a Call Graph Profile

The `gprof` constructs a *call-graph profile* for a program. The call-graph profile not only includes the 'flat' profile in the same style as `prof`, but it also displays the callers of a specific routine, and the number of times a routine was called or called another routine. For more information see *Programming Utilities and Libraries* and the `gprof` manual page.

Analyzing Code Coverage

The `tcov` utility is a code coverage tool that satisfies two widely divergent needs:

- it increases the resolution of `prof` and `gprof` down to the statement level, thereby providing extremely detailed analysis of where a program spends its time; and,
- it provides a report on which parts of a program are actually being executed. Such a report can be used (for instance) to discover how much testing a given set of regression tests are actually doing.

For more information see *Programming Utilities and Libraries* and the `tcov` manual page.

13.6. Program Generation Tools

Frederick Brooks pointed out[1] that there is a world of difference between a *program* (something that a couple of guys can cobble together in their garage over a weekend) and a *programming systems product* (a whole system that must work together and be documented). The SunOS operating system supplies many tools for assisting in the job of generating large systems. Two of the tools that assist *programming in the large* are `make`, building and maintaining consistency, and `sccs` for maintaining history.

Building and Maintaining Programs

The `make` utility is an indispensable tool for making sure that large programs are properly compiled with minimal effort. `make` has several unique features, including:

- ability to specify through a control file (called a *makefile*) what you want to build — called a *target*, things the target depend upon — called *dependencies*, and to go about constructing the target from its dependents — *rules*; and,
- has innate rules that specify the dependencies between object files and the C compiler, `yacc`, `lex`, etc.

For more information see *Programming Utilities and Libraries* and the `make` manual page.

Maintaining History

The `sccs` utility is the Source Code Control System. `sccs` maintains and controls multiple versions of text files, and also maintains the multiple versions in an SCCS database. Features of `sccs` include the ability to:

[1] *The Mythical Man Month*

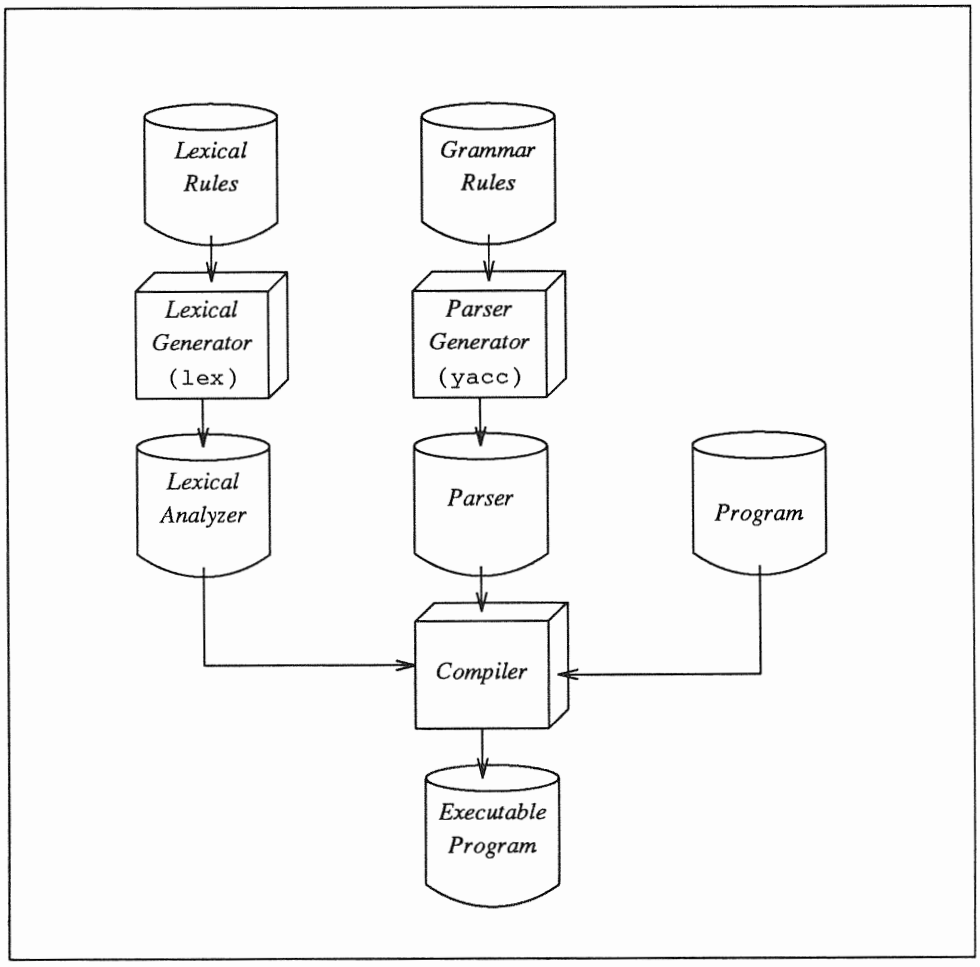
- create an SCCS database for a file to establish the initial version;
- produce a read-only copy of a file from the database for compiling or any other activity that doesn't involve changing the file;
- edit a writable copy of a file from the database for making changes; and,
- `delta` the modified copy of the file back into the database when you are satisfied with the changes you made. The *delta* is a record of the differences between this version and the last version. The current version of the database file is thus a complete list of all changes made during the file's history.

For more information see *Programming Utilities and Libraries* and the `sccs` manual page.

13.7. Compiler Development Tools

The `lex` and `yacc` utilities began life to assist generating lexical analyzers and syntactic parsers for compiler development. Over time, they have been applied to other areas such as a language for describing equations in document production (`eqn`), for the syntax analyzer for the `make` program, and for a language to describe pictures for document production (`pic`). The `lex` and `yacc` utilities are constructed to work together, as shown in the figure below.

Figure 13-5 `lex` *and* `yacc` *in Program Development*



Generating Lexical Analyzers

The `lex` utility generates lexical analyzers. Features of `lex` include:

- converts specification of regular expressions and semantic actions into a recognizing subroutine;
- arbitrary C functions may be called upon isolation of each lexical token;
- full regular expression, plus left and right context dependence; and,
- resulting lexical analyzers interface cleanly with `yacc` parsers.

For more information see *Programming Utilities and Libraries* and the `lex` manual page,

Generating Syntactic Parsers

The `yacc` utility is an LR(1)-based compiler writing system. Features of `yacc` include:

- during execution of resulting parsers, arbitrary C functions may be called to do code generation or semantic actions;

- BNF syntax specifications;
- precedence relations; and,
- accepts formally ambiguous grammars with non-BNF resolution rules.

For more information see *Programming Utilities and Libraries* and the `yacc` manual page.

13.8. Other Programming Tools

In addition to the tools described above, there are some ancillary utilities that find diverse applications.

Macro Processing

The `m4` utility is a general purpose stream-oriented macroprocessor that recognizes macros anywhere in text. Some features of `m4` include:

- syntax fits with functional syntax of most higher-level languages; and,
- can evaluate integer arithmetic expressions.

For more information see *Programming Utilities and Libraries* and the `m4` manual page.

Calculators

The `dc`, an interactive programmable desk calculator and `bc`, C-like interactive interface to the `dc` desk calculator are also good programming tools. For more information see, chapter 3, *SunOS — User Features. User Access and Commands*, and *Games, Demos & Other Pursuits*, and the `dc` and `bc` manual pages.

13.9. Summary of Language Utilities

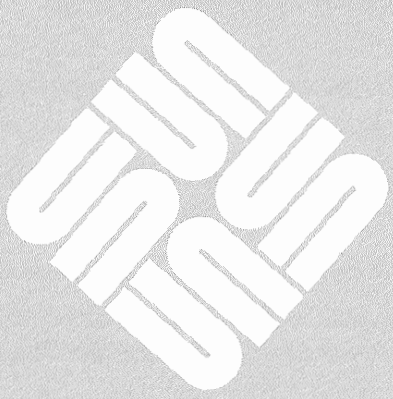
The following table contains an alphabetical list of the software development utilities described in this chapter.

Table 13-1 *Summary of Language Processing Programs*

Program Name	*Function*
`adb`	debug at the machine level
`ar`	build and maintain libraries
`dbx`	debug programs at the source level
`dbxtool`	debug programs at the source level
`gprof`	generate a call graph profile
`ld`	link programs
`lex`	generate lexical analyzers
`lint`	check validity of C programs
`m4`	macro processor
`make`	build and maintain programs
`nm`	display the namelist
`od`	dump file contents
`prof`	profile a program
`sccs`	control revision history
`size`	display size of a program
`strip`	strip relocation and symbol table
`tcov`	analyze code coverage
`time`	time a program
`yacc`	generate syntactic analyzers

14

Graphics Software

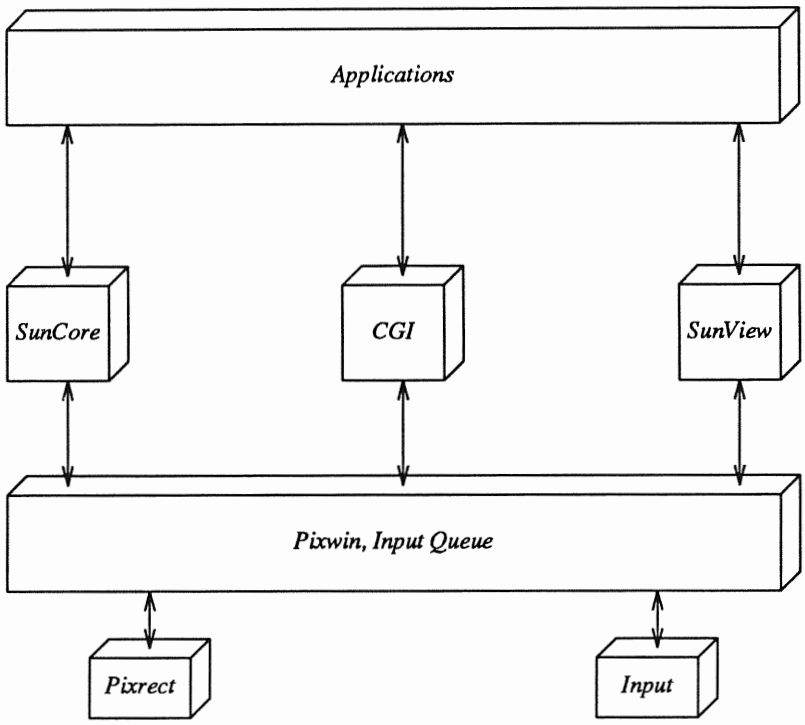


14

Graphics Software

Sun Microsystems graphics software supports several standards. The figure below illustrates the relationships between the various graphics packages in the SunOS 4.0 Release.

Figure 14-1 *Graphics Standards*



Sun Microsystems supplies the following graphics packages in SunOS 4.0:

14.1. SunCore

SunCore is an implementation of the *ACM SIGGRAPH Core System.* SunCore is a comprehensive package of engineering graphics software providing support for interactive 3D graphics application programs. SunCore conforms to level 3C (dynamic output with 3D scaling, rotation and translation) of the Core specification for output primitives, and to level 2 (complete input) for input primitives. For more information see *SunCore Reference Manual.*

14.2. SunCGI

CGI was a proposed standard when SunCGI was written. CGI does not appear to be moving toward approval in the near term, while GKS, PHIGS, and CGM have been approved as standards. The process of transition to new standards means a phase-out of SunCGI by SunOS 5.0. This provides 18 months for transition to new software technology.

SunCGI is an implementation of a draft of the ANSI *Computer Graphics Interface* (CGI). Previously, CGI was known as the *Virtual Device Interface* (VDI) standard. SunCGI provides access to low-level graphics device functions without the restrictions, benefits, or overhead of higher-level graphics packages like SunCore . SunCGI is useful for 2D graphics programs that do not require segmentation or transformations. The absence of segmentation from SunCGI makes drawing diagrams faster and simpler, but does not provide automatic picture regeneration. SunCGI programs are usually smaller and more efficient than SunCore programs with similar functionality. For more information see *SunCGI Reference Manual.*

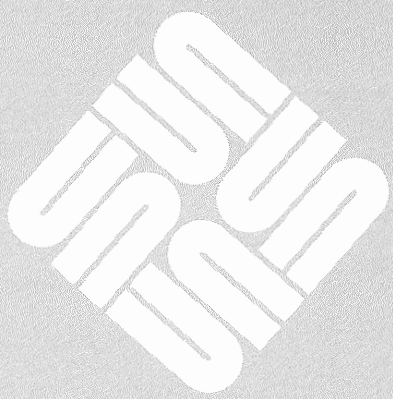
14.3. Pixrect

The Pixrect graphics library is a set of RasterOp routines common among all Sun workstations. With these routines, application programs can be written that access the display on all Sun products.

In the Sun graphics software world, the Pixrect library is a low-level package, sitting on top of the device drivers. For most applications, the higher-level abstractions available in SunView and the Sun graphics libraries are more appropriate. For more information see *Pixrect Reference Manual.*

15

Unbundled and Third Party Software



15

Unbundled and Third Party Software

This chapter introduces software products which are available for purchase, but are not part of the standard SunOS operating system, These software products are termed 'unbundled', since they are not included in the software you receive when you purchase your Sun system. Also included in this chapter is an introduction to the third party software program called Catalyst and the Sun User's Group.

15.1. Unbundled Software

There are numerous software packages available for applications in programming, data communication, artificial intelligence, networking, graphics, windows, and office automation. Software for each of these applications is described below.

Software Development

High-performance compilers are available for the most commonly used languages including: an enhanced ANSI FORTRAN 77 with VAX/VMS FORTRAN 4.0 extensions, Pascal, Ada, Modula-2, and Sun Common Lisp for artificial intelligence.

Sun's core languages feature interlanguage calling (the ability to use FORTRAN routines in a C program) and have full access to systems, graphics, networking, and user interface packages. There is also a family of cross compilers for the core languages that produces executable binaries for multiple Sun architectures from a single development machine. The cross compilers eliminate the need to port individual software packages to each architecture,.

SunPro — Interface for Sun Languages

SunPro is a set of integrated program development facilities that provide a consistent user interface for standard Sun languages. Features of SunPro include:

- `dbxtool` which provides window and mouse control for almost all debugging operations and also provides text editing within the debugger;
- an enhanced `make` command which includes automatic dependency generation and tracking, command consistency, and a fully functional *lib member* notification procedure;
- `filemerge`, based upon the UNIX System V `sdiff` procedure. With `filemerge` you can scroll through versions of different files and recently merged files; and,
- a common compiler driver which provides a consistent compiler interface and a standard mechanism for adding extra compilers.

SunLink — Data Communication

SunLink is Sun's family of data communications software and hardware products. These products provide multivendor compatibility, wide-area networking, and a migration path to emerging communication standards.

Using SunLink products you can link IBM mainframes to DEC minicomputers, word processors, PCs and other computer systems. With SunLink wide-area communication products, geographically distributed networks work together as if they were in one location and benefit from the merged NeWS and X window system.

There are numerous products available in the SunLink family for data communication in:

- wide-area and multivendor networking;
- communication through packet-switched networks; and
- IBM and DEC connectivity.

SunUNIFY

SunUNIFY is an extended relational database system that provides complete development facilities for interactive and networked applications. Features of SunUNIFY include:

- ad hoc queries and updates in SQL, and English-like language;
- concurrent networked database access;
- formatted reports, statistics, and form-letter generation;
- fast data access from programmed applications;
- forms-based data entry and query tools;
- tools to generate custom forms-based applications; logical data integrity checking and data access protection; and,
- database dump, load, logging, and crash recovery.

SunSimplify

SunSimplify is a set of front-end interfaces for SunUNIFY. SunSimplify incorporates graphics, the mouse, and programming interfaces. These features offer advanced user database tools and facilitates development of custom interactive database applications.

The major components of SunSimplify include:

- Schemadesign, a graphical SunView window-based tool for creating and displaying data schemas;
- Databrowse, a unique SunView window-based data display and update tool for accessing database information in an object-oriented fashion;
- local and remote access to the relational query language, report generator, and menu capabilities of the underlying database system;
- ERIC, a programmer's interface to the underlying database system; and,
- local and remote data access on the network.

SunINGRES

SunINGRES is another available full-featured, relational database management system. Features of SunINGRES include:

- concurrent access to databases in a heterogeneous network;
- two high-level data-manipulation and query languages (SQL and QUEL);
- precompilers for SQL and QUEL statements embedded in other languages;
- sophisticated query optimization;
- interactive, forms-based Visual Programming tools;
- transactions for reliability, recoverability, and integrity; and,
- relational data model with integrated data dictionary.

SunTrac — Project Management Software

SunTrac project management system consists of tools to aid project management problems. SunTrac blends sophisticated analysis with networked computing technology. Tools of the SunTrac system include:

- SKETCH, a mouse and menu driven graphical network editor which creates, displays, and enters data to PERT network diagrams;
- TRAC (total risk analysis calculation), a new algorithm which rapidly analyzes risk to cost, schedule, and resources;
- LEVEL (interactive/automatic resource leveling tool), a mouse-driven input to the Gantt chart and Resource profile graphically levels resource usage, and schedules and controls activities;
- ASSIGN (optimal overtime allocation tool), plans the least-cost reduction in project schedule keyed to a user-selected productivity factor;
- PROFILE (graphic summary), displays, analyzes, and prints staffing level and cost profiles for the selected network and data;
- REPORT (activity reports), reviews, sorts, and extracts detailed planning, scheduling, and analytical information; and,
- HELPTRAC, graphically browses and searches online help text.

SPE — Symbolic Programming Environment

The Symbolic Programming Environment (SPE) provides sophisticated program development tools for artificial intelligence and symbolic programming. SPE is a set of specially designed tools for efficient program development in Lisp and other symbolic languages. The tools facilitate rapid prototyping. Features of SPE include:

- editor — a special Emacs-style editor for developing Lisp code, with a set of Lisp libraries for extending the editor's functionality;
- common Lisp window manager — a toolkit of Lisp functions that define a rich set of windowing operations for Lisp programmers.
- Lisp listener — a high-level Lisp interpreter; similar to the UNIX shell;
- window-based debugger — a window-oriented debugging tool that provides information on the state of the program, such as function call tracing, passed

arguments, local variables, and stack information;

- widow-based data inspection — an interactive tool for inspecting instances of Lisp data structures in an application;
- window-based single stepper — a trace mechanism for following the line-by-line execution of Lisp source code;
- window-based trace — a tools that monitors the execution of a particular function(s) during runtime;
- source code analyzer — produces a set of relationships of the functions in the source code where users can follow the control of flow while editing the program;
- source code finder — a tool for finding the specific source code for any given function;
- application manager — an abstract representation of the entire application that keeps track of the dependencies of the source files and functions involved in an application; and,
- UNIX interfaces — a set of library functions that create interfaces between Lisp and UNIX that integrate the Sun's networking tools with the symbolic application programs.

X11/NeWS — Network/extensible Window System

Along with SunView/SunWindows, Sun has expanded its window systems offering to include Network/extensible Window System (X11/NeWS). The X11/NeWS product provides a window system in a networked environment. X11/NeWS merges Version 11 of the X Window System from MIT with NeWS, which provides Sun's unified, native window system and forms the foundation for Sun's future window systems technology. This combined technology is the network-based window platform for the development of portable, flexible, and high-performance window-based applications.

SunGKS — Sun Graphics Kernel System

SunGKS is an implementation of the *Graphical Kernel System* (GKS), an application software interface standard developed by the *International Standards Organization* for interactive computer graphics. It is a library of functions that support floating point-based 2D graphics applications on Sun workstations. The GKS specification uses the notion of a workstation to indicate a virtual display surface. SunGKS uses the windowing facilities provided by SunView to manage multiple GKS workstations on the same physical Sun workstation.

Here is an overview of SunGKS functionality:

- *control* functions control the current state of GKS;
- *primitives* are functions for drawing lines, filled polygons, collections of markers, text and virtual bitmaps;
- *attributes* control the graphical appearance of the primitives. For example, the *polyline* primitive has three attributes: line width, line style and line color. Attributes can be determined individually or in named groups called *representations*;

- *segments* collect primitives into static groups called that can be scaled, translated and rotated;
- *transformations* control the graphical transformations between world coordinates, normalized device coordinates (NDC), and device coordinates (DC);
- *input* functions manage the use of input devices with application programs through an abstract model called a logical input device. This scheme provides portability of applications. Each GKS implementation maps these logical input devices onto available physical input devices like the mouse and keyboard;
- *metafiles* provide a mechanism for passing graphical data between GKS systems (including systems not manufactured by Sun); and,
- *inquiry* functions permit examination of the current state and capabilities of *SunGKS*.

GKS is a functional standard supported in a given language or environment with specified *bindings*. SunGKS is implemented in C with a software interface based on C functions. SunGKS also includes a FORTRAN compatibility library that allows FORTRAN applications to make calls on the C functions of SunGKS .

SunAlis — Office Automation System

The SunAlis office software system lets users share any type of data — text, graphics, spreadsheets or databases. It provides document preparation and management tools for analyzing, communicating and sharing different types of information. Features of SunAlis include:

- a document composer with automatic text formatting;
- universal graphics editor;
- spreadsheet with built-in goal analysis;
- personal and office database management;
- DIF, UNIX file import/export;
- personal and shared document filing;
- electronic mail and messaging;
- personal time management;
- calendar management; and,
- archiving and backup utilities.

NSE — Sun's Network Software Environment

The Sun Network Software Environment (NSE) is a network-based object manager for software development. NSE facilitates parallel development of large software systems, maintains consistency and completeness of objects being developed, and is extensible to different types of objects. Parallel development is supported through an optimistic concurrency control mechanism, where developers do not acquire locks before modifying objects. Instead, developers copy objects, modify the copies, and merge the modified objects with the originals. NSE provides copy operations on units that are complete and consistent sets of

objects. These copy operations produce logical copies; a physical copy of an object is not made until the object has been modified.

Features of NSE include:

- compatibility with today's tools for a clear migration path to more advanced technology;
- complete programmatic interface for a consistent and standard mechanism for software tool integration;
- supports all development objects produced during the software life cycle;
- transparent network-wide access to NSE objects making objects local regardless of physical location;
- NSE environments are complete and isolated creating no interference for object sharing;
- a notification utility for monitoring object changes network-wide;
- a link facility for traceability through the connection of independent objects;
- a version control system which records versions and histories of all development objects; and,
- consistent window and mouse-based user interface.

PC Compatibility Products

Products available for PC's include PC-NFS, a network file system for personal computers, PC-NFS electronic mail and network backup system, and PC-NFS programmer's toolkit.

The PC-NFS network file system integrates IBM PCs (and 100 percent compatible systems) into local area networks (LANs), allowing them to share information and resources with workstations, minicomputers, superminicomputers, and even personal computers.

The PC-NFS electronic mail and backup system allow PC users to access network-wide resources in a multivendor environment. The PC-NFS electronic mail system is a full-featured, window-driven facility for electronic mail across different hardware architectures, networks, and operating systems. The PC-NFS backup is a network resource and management facility that lets you back up local or distributed files directly to a mass-storage device or tape drive on most NFS servers.

The PC-NFS programmer's toolkit provides a set of network development tools that bridge DOS and UNIX systems (SunOS, System V, 4.2BSD, Ultrix). This toolkit is a set of libraries that run on IBM PC, PC/XT, and PC/AT, and other 100 percent compatible machines using DOS 3.0 and PC-NFS 2.0 or higher. The toolkit offers a single programming interface across a host of network boards, and the libraries can be used to develop a distributed database.

Also available is the Sun Integrated Personal Computer (SunIPC), a coprocessor boards and software product for VME-based Sun workstation computers. The SunIPC board lets you run PC-DOS application software in a window on a Sun workstation. The SunIPC gives you access to the full range of software available

to IBM PC, PC/XT, PC/AT, and compatible computers.

For More Information

For more information on any of the unbundled products described above call your Sun sales office.

15.2. Catalyst — the Third Party Program

Sun's Catalyst program is built upon a relationship established with third-party hardware and software vendors. These vendors have implemented their specialized application software or hardware to run on Sun systems. The third party program is an ongoing part of Sun's marketing organization to provide the widest range of applications to serve your computing needs. Currently there are more than 500 third party vendors in the Catalyst program. Listed below are the major application areas as listed from the Catalyst catalog. For detailed information and people to call, please refer to the appropriate section of the Catalyst catalog.

- electronic publishing and office automation;
- artificial intelligence;
- software engineering;
- earth resources engineering;
- finance;
- biological and physical sciences;
- electrical engineering;
- add-on hardware;
- mechanical/architectural/civil engineering;
- manufacturing; and,
- miscellaneous.

15.3. Sun User Group

The *Sun User Group* is an independent organization of individuals and institutions who share a common interest in Sun workstations and related products. The Sun User Group encourages exchange of information between Sun workstation users and Sun Microsystems. The group collects and disseminates techniques, software, documentation, procedures, and related interesting information.

Donated Software Tape

The Sun User Group offers a distribution tape of donated software at a nominal charge. The software is contributed from the user community and is not supported by Sun Microsystems. Among the many items of software on the distribution tape can be found:

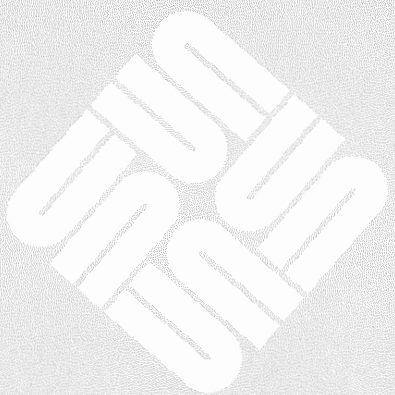
- device drivers for hardware products that are not a directly supported part of the Sun workstation product family;
- utilities that run in the Sun Windows environment;
- communications enhancements;

- compilers for languages other than those that Sun Microsystems supports directly;
- editors and spreadsheets;
- mail handlers;
- games; and,
- system administration tools.

You must be a member of the Sun User Group, and a Sun customer, to get a copy of the Donated Software Tape.

16

Sun Technical Documentation



16

Sun Technical Documentation

Included with your Sun system is a comprehensive package of documentation for the hardware and the SunOS operating system. The documentation is grouped into a *docubox*. The docubox contains four miniboxes, including:

- *System Administration Manuals*;
- *Reference Manuals*;
- *Beginner's Guides*; and,
- *Programming Guides*.

Hardware manuals are packaged separately from the docuboxes, but all are included when you purchase a Sun system.

The figure which follows illustrates the basic documentation miniboxes. The *System Administration* minibox contains a *Read This First* (RTF) for the entire docubox. Each minibox, including *System Administration*, are also accompanied with an RTF.

Figure 16-1 *Miniboxes of Technical Manuals*

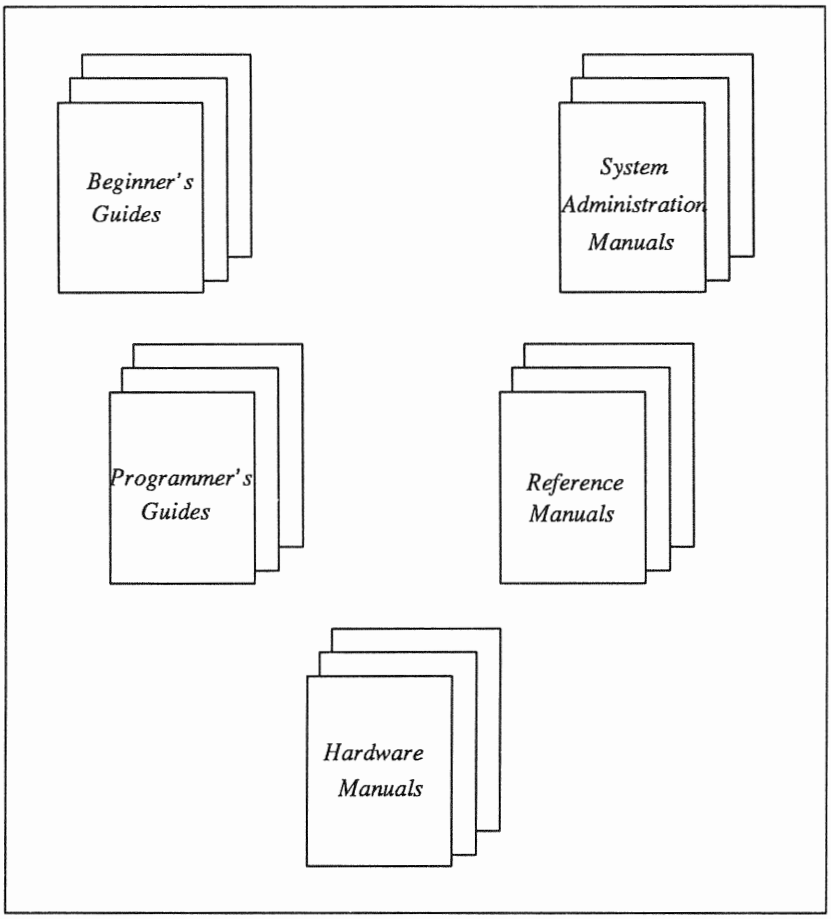
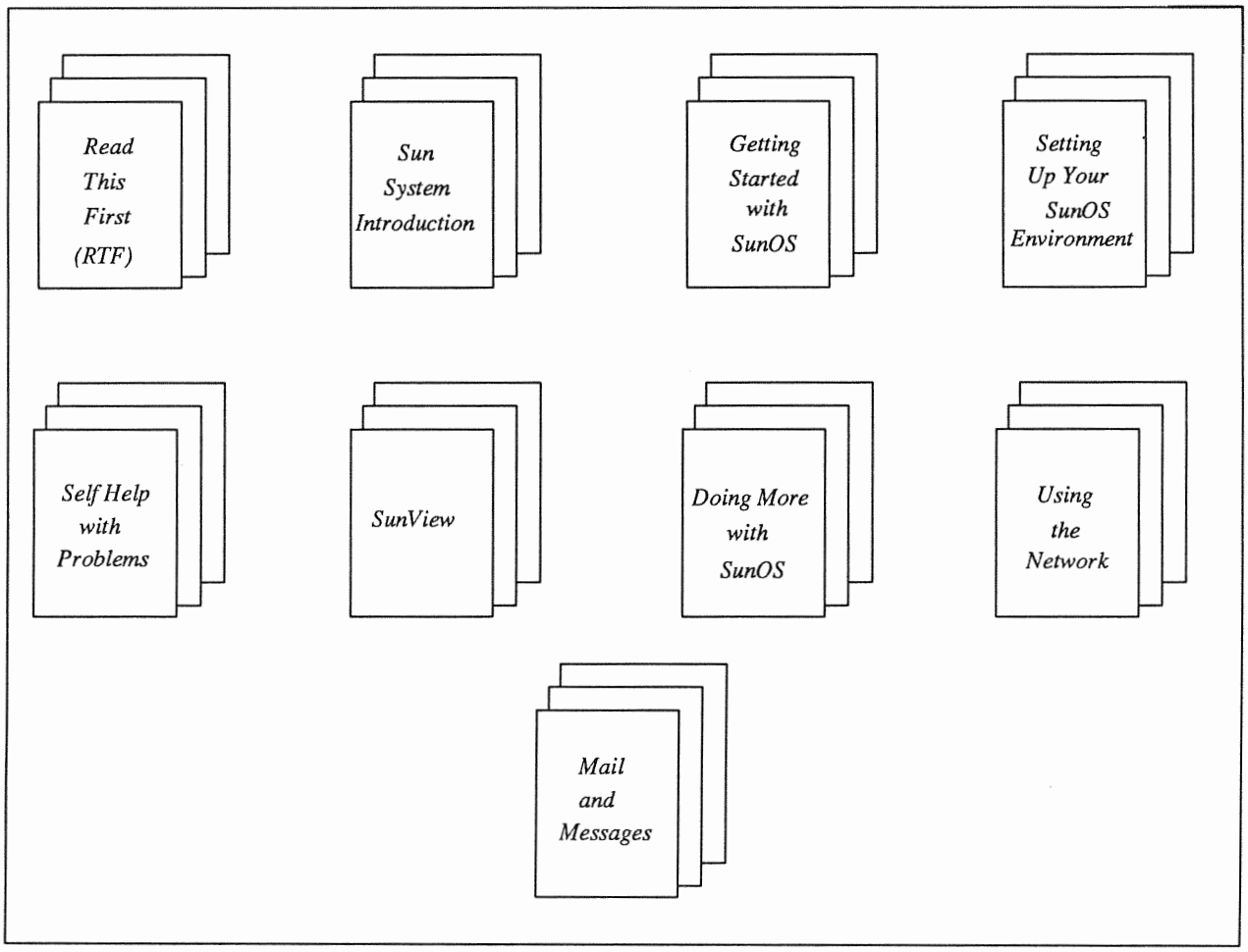


Figure 16-2 *Beginner's Guides Minibox*



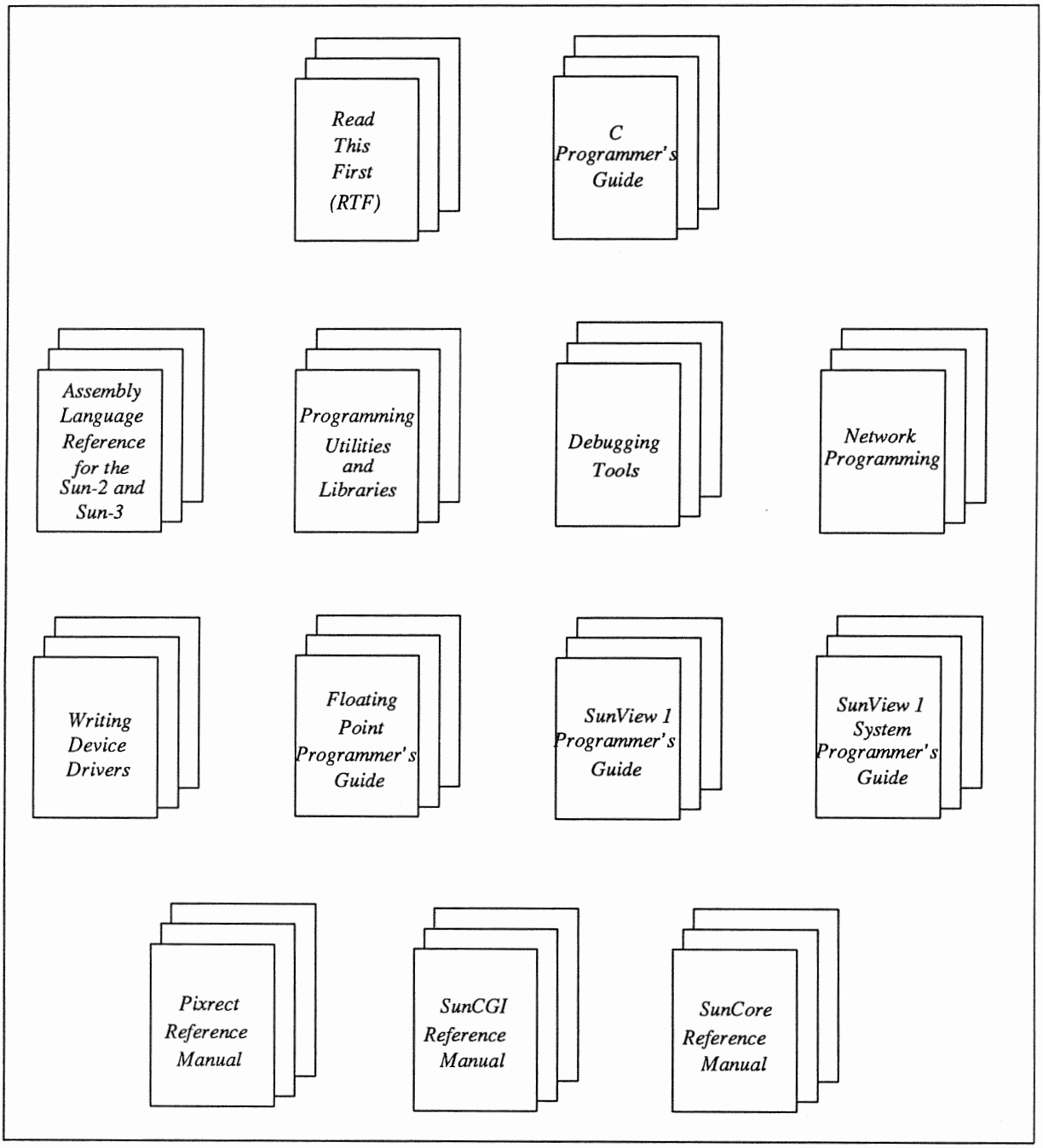
16.1. Beginner's Guides

The *Beginner's Guides* minibox includes the following manuals:

- *Sun System Introduction*— an introduction to the Sun family of workstations and the SunOS operating system. This manual is a roadmap for what is available and where to find more information.
- *Getting Started with SunOS: Beginner's Guide*— is a basic introduction to SunOS user commands.
- *Setting Up Your SunOS Environment: Beginner's Guide* — describes how to customize your shell, editor, etc.
- *Self-Help with Problems: Beginner's Guide*— helps you diagnose and solve problems common to beginning users.
- *SunView 1 Beginner's Guide*— isan introduction to the SunView window environment.
- *Mail and Messages: Beginner's Guide*— isan introduction on how to send and receive mail and messages.

- *Doing More with SunOS: Beginner's Guide*— describes more advanced user commands.
- *Using the Network: Beginner's Guide*— describes how to use the network effectively.

Figure 16-3 *Programmer's Guides Minibox*



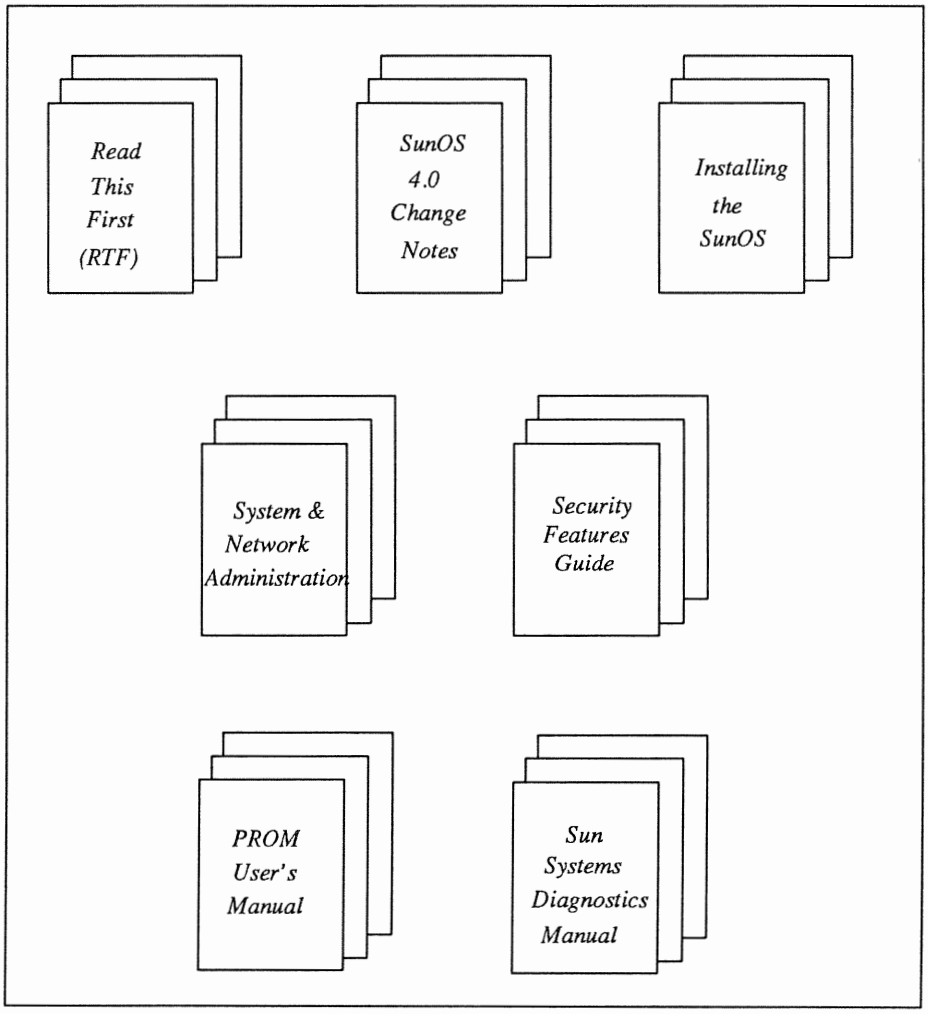
16.2. Programmer's Guides

The *Programmer's Guides* include the following manuals:

- *C Programmer's Guide*— describes using the C language under SunOS .
- *Assembly Language Reference*— describes MC680x0 programming under SunOS.

- *Programming Utilities and Libraries*— describes software development tools.
- *Debugging Tools*— describes debugging under SunOS.
- *Network Programming*— a programmer's reference manual covering `rpcgen`, remote procedure calls (RPC) and socket based interprocess communication (IPC).
- *Writing Device Drivers*— programmer's reference manual covering all of the hardware and kernel interfaces relative to the structure of device drivers. Also includes all STREAMS tutorial and reference material.
- *Floating-Point Programmer's Guide*— describes floating point usage and performance under SunOS.
- *SunView 1 Programmer's Guide*— a guide and reference manual for programmer's using SunView user interface tool kit to develop window-based applications.
- *SunView 1 System Programmer's Guide*— provides additional information for low-level SunView and SunWindows packages and routines.
- *Pixrect Reference Manual*— a reference manual describing a set of RasterOp graphics routines common among all Sun workstations. Describes how applications can access a frame buffer in a device-independent way across the Sun product line.
- *SunCGI Reference Manual*— a reference manual for Sun's 2-D integer-based, device-level graphics interface based upon *Computer Graphics Interface* (CGI), proposed by the American Standards Institute (ANSI).
- *SunCore Reference Manual*— a reference manual for Sun's 3-D graphics software interface based upon the ACM/SIGGRAPH core proposal.

Figure 16-4 *System Administration Minibox*



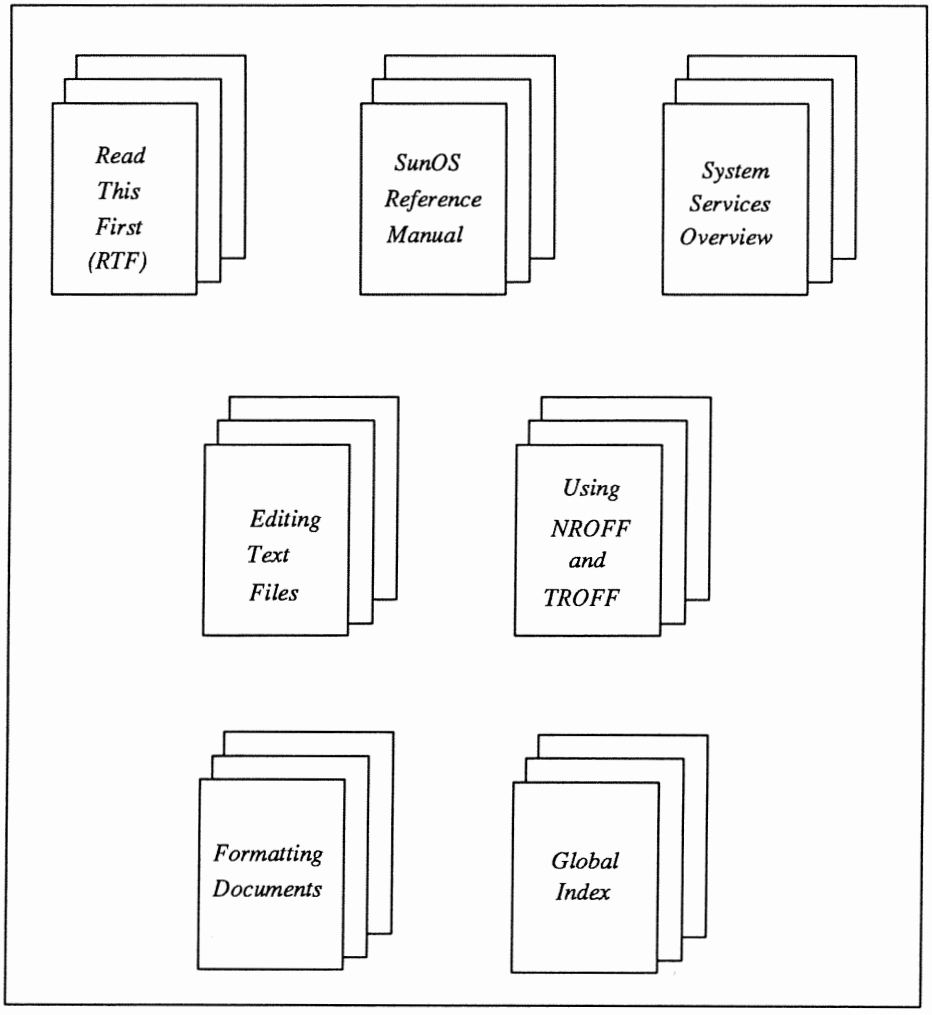
16.3. System Administration Manuals

The *System Administration Manuals* include the following:

- *SunOS 4.0 Change Notes*— describes the new features for SunOS release 4.0.
- *Installing the SunOS*— installation instructions for loading the SunOS operating system on to your Sun hardware.
- *System and Network Administration*— describes how to administer an already running system, set up with `suninstall`. Also describes administration procedures for the system itself, and if applicable, procedures for the system's network environment.
- *Security Features Guide*— describes SunOS security features for the programmer, user, and administrator. Also includes information on network security.

- *PROM User's Manual*— describes power-up sequences, tests, and PROM monitor commands, including EEPROM and NVRAM layouts for all Sun systems.
- *Sun System Diagnostics*— provides a user guide description for the Sun System Diagnostics Program (`sysdiag`). The `sysdiag` program is a system exerciser found on the SunOS distribution tape.

Figure 16-5 *Reference Manuals Minibox*



16.4. Reference Manuals

The *Reference Manuals* include the following:

- *SunOS Reference Manual*— the manual pages for all Sun system commands and system calls. The manual pages are divided into eight sections:
  1. user commands
  2. system calls for the C language
  3. library routines for the C language
  4. devices
  5. file formats
  6. games and demos
  7. miscellaneous tables

8. maintenance commands

- *System Services Overview*— an introduction and overview for all system services and interfaces.
- *Editing Text Files*— describes the text editors, `vi`, `ex`, `ed`, and `sed`. Also describes how to scan files using `grep` and `awk`, and commands for comparing files and directories.
- *Using* `nroff` *and* `troff`— a reference guide for the `nroff/troff` formatting programs.
- *Formatting Documents*— a tutorial for macro packages and `nroff/troff` preprocessors for creating tables, equations, and bibliographies.
- *Global Index*— and index for all SunOS release documentation.

16.5. Hardware Manuals

Accompanying the docubox manuals is a set of hardware manuals. What kind and how many hardware manuals you receive depends upon your system configuration. Every system includes an *enclosure install* type of manual and install manuals for each CPU board you receive. These manuals provide instructions on how to set up your system hardware. There is also a manual(s) for the cardcage slot assignment and backplane configuration procedures.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Notes

Notes

Notes

Notes

Notes

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 37-29639

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor
Green Lane
Bagshot
Surrey, GU19 5NL
England
0276 51440
TLX 859017

Australia: (02) 436 4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
Nordic Countries: (08) 764 78 10
Switzerland: (1) 82 89 555
The Netherlands: 02155 24888
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276-51440

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales