sun®
microsystems

Sun System Overview

Credits and Trademarks

Contents

Figures

Preface

Welcome to Sun Microsystems hardware and software products. This textbook is the first edition of a summary and overview of Sun Microsystems' hardware product family, operating system software, application oriented software, and technical documentation offerings.

About This Book

This book is to help you get acquainted with Sun Workstations and File Servers, and to guide you through the available software.

Who Should Read This Book

You should read this book if you are in any or all of these situations:

- You're thinking of buying Sun Workstations and you want to know what's available.
- You've already bought a Sun Workstation, or your boss has dropped a Sun Workstation on your desk, and you don't know where to start.
- You know nothing about the UNIX operating system or any of the Sun software.
- You know about the UNIX operating system but you want to find your way around Sun Microsystems' rendition of it.
- You've run out of bedtime reading material or are consumed by curiosity.

What's In This Book

This book is a guide — a set of signposts if you like — to the general characteristics of Sun Microsystems' hardware, software, and documentation. This book tells you *what's available* and where to look for more information. This book is *not* a tutorial or a 'how to' guide for any of the software.

Contents of the Chapters

Chapter 2 — *Sun Product Family* — supplies an overview of the Workstation systems.

Chapter 3 — *Sun Operating System — User Features* — discusses the major features of the Sun version of the UNIX operating system.

Chapter 4 — *SunView — User Environment* — covers the external (user accessible) features of the Sun Window system — the *User Interface*.

Chapter 5 — *Using the Shells* — discusses the *Shells*. The Shell is still the principal command interface to the UNIX system utilities.

Chapter 6 — *User Access and Commands* — describes the process of gaining access to the system and its applications software.

Chapter 7 — *Working With Files* — covers the facilities available to work with files and make use of the hierarchical file system.

Chapter 8 — *Communications Facilities* — describes the local are network facilities and the remote communication facilities.

Chapter 9 — *Text Editing and Text Processing* — provides a description of the major programs for creating, editing, and manipulating text.

Chapter 10 — *Programming Languages* — describes the programming languages and language tools that are standard in the Sun system.

Chapter 11 — *Software Development Tools* — supplies a picture of the many tools available to assist the programmer in debugging and maintaining software.

Chapter 15 — *Document Production* — covers the application software oriented to producing documents.

Chapter 16 — *System Administration* — describes the tools available to the system administrator.

Existing UNIX users should note that the documentation described here is *Suns'* documentation, which has diverged dramatically from the standard UNIX system manuals.

Chapter 17 — *Sun Technical Documentation* — describes the organization of the Sun suite of technical manuals.

Chapter 18 — *Third Party Software* — contains pointers to the *Catalyst* third-party software program and to the Sun Users' Group.

Chapter 12 — *Graphics Tools* — describes an overview of Suns' graphic libraries.

Chapter 13 — *Sun Operating System — Internal Features* — describes the programming interfaces to the Sun operating system.

1

Introduction

1 Introduction

Welcome to your Sun Workstation. If you haven't ever seen a Sun Workstation before or used Sun Microsystems' version of the UNIX operating system, read on.

The Sun Workstation

Sun Workstations are high-performance bitmapped workstations oriented towards applications in engineering, CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CAE (Computer Aided Engineering), CAP (Computer Aided Publishing), graphics, software development, and many more.

Basic Hardware

Sun Workstations are built around the Motorola MC68020 and MC68010 family of microprocessors. Sun Microsystems have enhanced these computer chips with unique memory management hardware to provide demand-paged virtual memory. Output is to 19-inch bit-mapped monochrome and color displays having 1152 by 900 pixels. The display screen has its own memory — the *frame buffer* to provide high-speed data transfer to the screen. Input is via a modern keyboard and a mouse pointing device.

Basic Software

Sun Workstations run a heavily enhanced version of the 4.2 BSD UNIX system as derived from the University of California at Berkeley. The operating system provides support for interprocess communication and local area networking.

The UNIX system comes equipped with a host of software as a base level package. Supported languages are C, FORTRAN 77, Pascal, and Assembler. Many tools support software development applications. The UNIX system originally evolved in an environment of computer scientists doing research into programming. Utilities exist for performing statistical text processing and document preparation.

To this base of utility packages, Sun Microsystems have added a *User Interface* package based on overlapping windows, and comprehensive libraries of graphics packages to support the more common graphics standards.

As you read through the rest of this book, the various chapters introduce you to the 'flavor' of what's available in the system.

2

Sun Product Family

2

Sun Product Family

Sun Microsystems builds two major families of monochrome and color workstations, plus rack-mountable systems intended to act as file servers. There is a wide choice of peripheral equipment.

Sun products are built around industry standard chips, bus architectures, and network architectures:

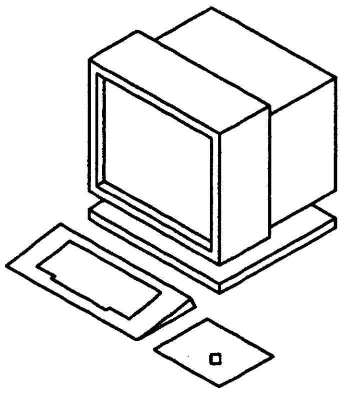
- The Sun-3 product family is a group of 32-bit systems based on the Motorola MC68020 CPU chip and the high-speed 32-bit VMEbus.
- The older Sun-2 product family (Sun-2/120 and Sun-2/170) are 16-bit systems based on the Motorola MC68010 CPU chip and the older Multibus (IEEE P796 bus).
- The newer Sun-2 product family (Sun-2/130 and Sun-2/160) are 16-bit systems based on the Motorola MC68010 CPU chip and VMEbus.
- All Sun Microsystems products communicate *via* the Ethernet — a 10 Megabit-per-second coaxial cable local area network facility.

2.1. Sun-3 Product Family

The Sun-3 product family is a group of 32-bit systems based on the Motorola MC68020 CPU chip and the high-speed 32-bit VMEbus.

- Sun-3 computers come standard with 2 Megabytes of real memory expandable up to 16 Megabytes of real memory.
- Sun-3 computers can address up to 256 Megabytes of virtual address space.

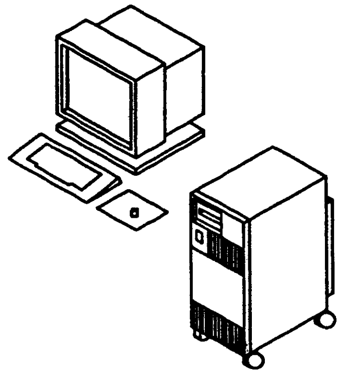
Sun-3/75M



The Sun-3/75M is a standalone monochrome desktop workstation normally supplied without mass storage devices.

- Main memory from 2 to 8 megabytes.
- All input-output traffic is done across the Ethernet. The Sun-3/75M thus requires other Sun products (such as the Sun-3/160S or Sun-3/180S described below) to act as file servers.
- Add on mass storage is available for the Sun-3/75M as an option.
- The Sun-3/75M has no external bus, unlike the other members of the Sun-3 family described below.

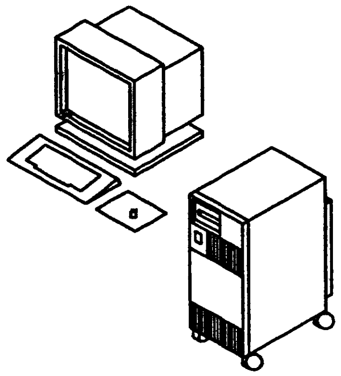
Sun-3/160M



The Sun-3/160M is a deskside *monochrome* workstation based on the VMEbus.

- 128 Kbyte frame buffer.
- Upgradable to a color monitor.
- The deskside pedestal has a 12-slot VME card cage.

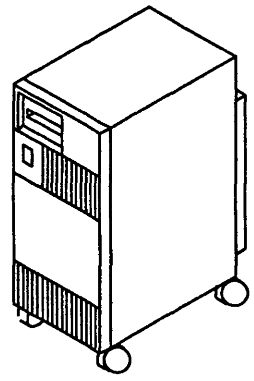
Sun-3/160C



The Sun-3/160C is a deskside *color* workstation based on the VMEbus.

- Eight planes of color.
- 256 simultaneously displayed colors.
- Palette of 2^{24} colors.
- Main memory available from 2 to 16 megabytes.
- Optional hardware includes a graphics processor (GP) and graphics buffer (GB).
- The deskside pedestal has a 12-slot VME card cage.

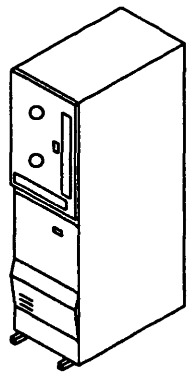
Sun-3/160S



The Sun-3/160S is a deskside file server based on the VMEbus.

- Standard system contains 71 (formatted) megabytes of mass storage.
- Main memory available from 2 to 16 megabytes.
- Mass storage can be expanded to 1.5 gigabytes.
- Can handle asynchronous terminals.
- The deskside pedestal has a 12-slot VME card cage.

Sun-3/180S



The Sun-3/180S is a rack-mountable file server based on the VMEbus.

- Main memory available from 2 to 16 megabytes.
- Standard system contains 130 (formatted) megabytes of mass storage.
- Mass storage can be expanded to 1.5 gigabytes.
- Can handle asynchronous terminals.
- The rack has a 12-slot VME card cage.

2.2. Sun-3 Optional Hardware

Optional hardware for the Sun-3 product family includes:

- Floating Point Accelerator
- Graphics Processor Board
- Graphics Buffer

Floating Point Accelerator

The Floating Point Accelerator board is intended for those applications requiring higher floating-point performance than the MC68881 offers. The floating-point accelerator runs up to four times faster than the MC68881 and gives two times the performance of a VAX 780 with floating-point accelerator. The floating-point accelerator complies with the IEEE 754 version 10 standard for floating point. Both 32-bit and 64-bit floating-point operations are supported.

Graphics Board

The graphics board is an optional board available for Sun-3 products and also for the Sun-2/130 and Sun-2/160 workstations. The graphics board provides dramatic improvements in graphics response time. Graphics applications using the graphics board can render 40,000 2D vectors per second or 25,000 3D vectors per second.

Graphics Buffer

The graphics buffer is an optional board available for Sun-3 products and also for the Sun-2/130 and Sun-2/160 workstations. The graphics buffer augments the graphics board and is intended for applications where 3D transformations and hidden surface remove is time critical. Using the graphics board/graphics buffer combination, graphics applications can shade and remove hidden surfaces of 3D polygons faster than 1 million pixels per second. Arbitrary polygons can be filled at up to 35 million pixels per second.

2.3. Sun-2 Product Family

The Sun-2 product family is a group of 16-bit systems based on the Motorola MC68010 CPU chip. One model in the Sun-2 family is a standalone desktop workstation with no external bus. Two models of the Sun-2 family are based on the Multibus. Two models of the Sun-2 family are based on the high-speed 16-bit VMEbus. Sun-2 computers some standard with 1 Megabyte of real memory expandable up to 4 Megabytes of real memory. Sun-2 computers can address up to 16 Megabytes of virtual address space. The Sun-2 product family currently consists of these standard offerings:

Sun-2/50

The Sun-2/50 is a standalone monochrome desktop workstation that is normally supplied without mass storage devices. All input-output traffic is done across the Ethernet. The Sun-2/50 thus requires other Sun products (such as the Sun-2/170 or Sun-2/120 described below) to act as file servers. Add on mass storage is available for the Sun-2/50 as an option. The Sun-2/50 has no external bus, unlike the other members of the Sun-2 family described below.

Sun-2/120

The Sun-2/120 is a deskside monochrome workstation based on the Multibus. The deskside pedestal has a 9-slot Multibus card cage.

Sun-2/170

The Sun-2/170 is a rack-mountable file server based on the Multibus. The rack has a 15-slot Multibus card cage. The Sun-2/170 can be expanded to four Megabytes of real memory.

Sun-2/160

The Sun-2/160 is a deskside color workstation based on the VMEbus. The deskside pedestal has a 12-slot VME card cage. A Sun-2/160 is field-upgradeable to a Sun-3/160C.

Sun-2/130

The Sun-2/130 is a deskside monochrome workstation based on the VMEbus. The deskside pedestal has a 12-slot VME card cage. A Sun-2/130 is field-upgradeable to a Sun-3/160M.

2.4. Peripherals

Peripherals that may be attached to Sun systems include SMD disks of various sizes ranging from 84 Megabytes to the 474 Megabyte 'Eagle' disk, nine-track tape, ¼-inch cartridge tape, SCSI disks and tapes.

3

Sun Operating System — User Features

3

Sun Operating System — User Features

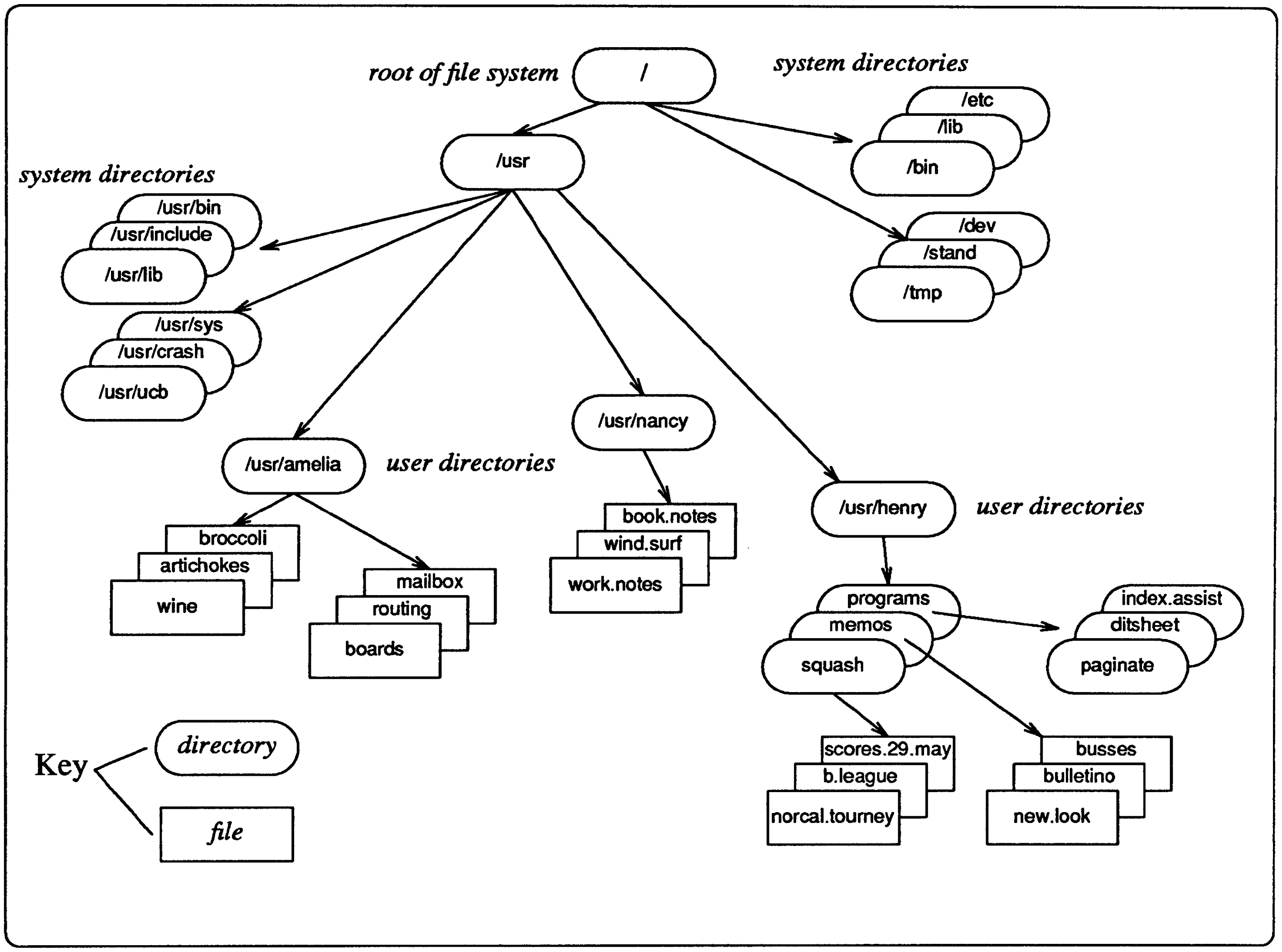
Sun Microsystems provides its own enhanced version of the UNIX operating system as the basic operating software to exploit the resources of Sun hardware products. Sun's operating system is derived from the 4.2BSD system developed for the DEC VAX at the Uniersity of California at Berkeley. 4.2BSD was in turn derived from UNIX system version 32V developed at Bell Laboratories to use the DEC VAX hardware. The 4.2BSD system runs at approximately 5000 VAX sites. Although the Sun hardware is microchip based instead of being a VAX, the Sun hardware has many features in common with the VAX architecture and so many of the same strategies can be employed in the operating system.

3.1. Hierarchical File System

The most visible aspect of the operating system is the file system — the facility that you use to store and retrieve data. At the user (command) level, the basic object on which you can operate is a *file*. A file is a collection of data with several attributes, the most important being its *name*.

The hierarchical *file system* provides the means to group related files within a *directory*. A directory is itself simply a special case of a file, which means that a directory can contain other directories as well as files. Directories can be grouped within directories to arbitrary limits in the Sun Microsystems operating system. Here is a representative picture of a file system containing directories and files for both the system and several users:

Figure 3-1 *Hierarchical File System*



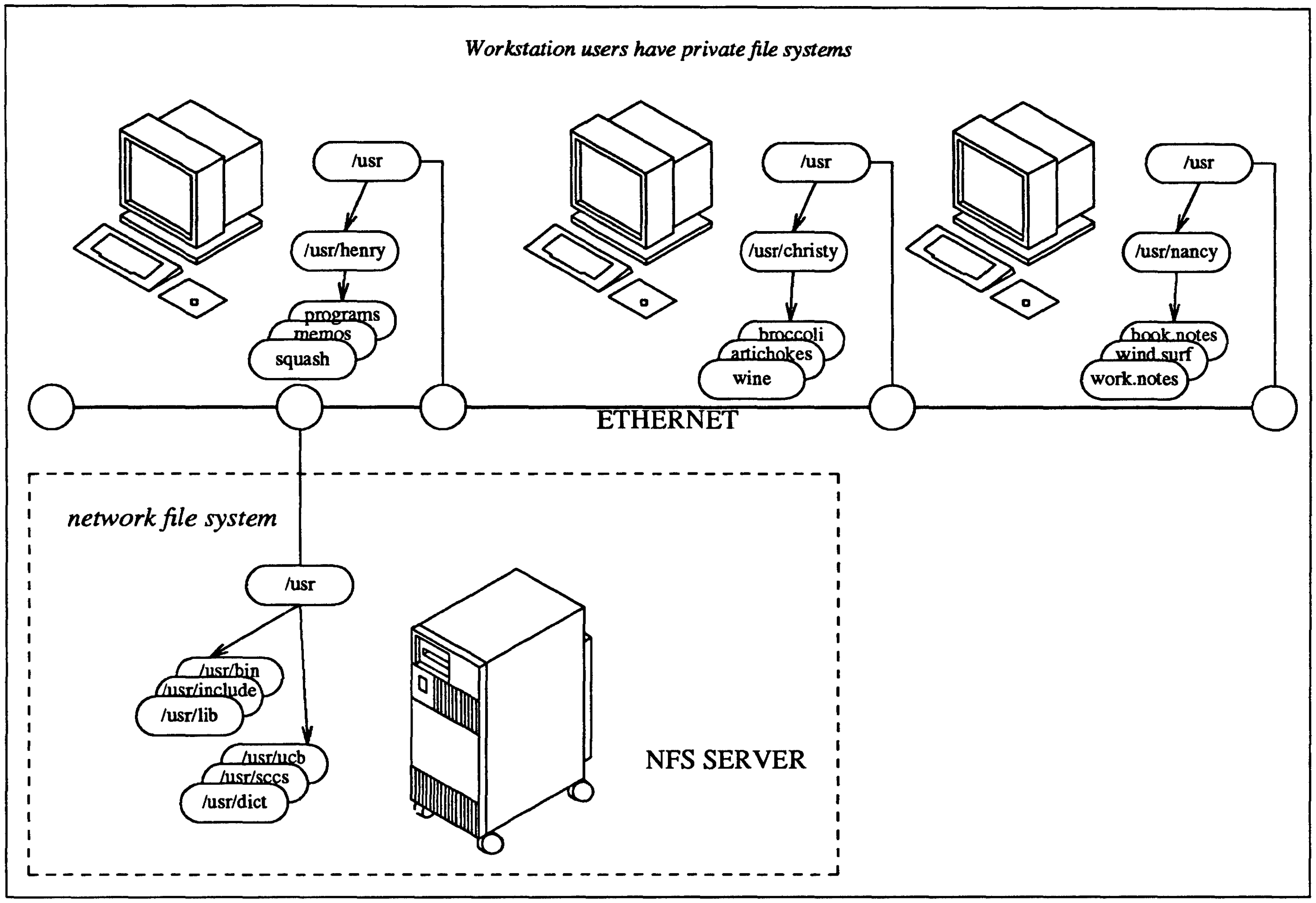
3.2. The Hierarchical File System in the Network

Sun Microsystems have taken the idea of file systems a major step further with the *Network File System*. The Network File System (NFS) is a facility for sharing files in a heterogeneous environment of machines, operating systems, and networks. Sharing is accomplished by mounting a remote filesystem, then reading or writing files in place. The network file system avoids the problem of each node in a network maintaining its own file systems and thereby duplicating resources.

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

Figure 3-2 below shows three workstations with their public `/usr` file systems mounted from a NFS server. All file trasnfers take place over the Ethernet.

Figure 3-2 *Network File System*



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

The network file system is Sun Microsystems' first and very major step towards addressing the issues of heterogeneous networks of resources.

Yellow Pages Network Service

The Yellow Pages (YP) is a network service to ease the job of administering networked machines. The YP is a centralized read-only database. For a client on the network file system, this means that an application's access to data served by the YP is independent of the relative locations of the client and the server. The YP database on the server provides password, group, network, and host information to client machines.

3.3. User Interface — Window System

The *user interface* based on the Sun window system (SunView) is described in more detail later on. Briefly, the user interface is based on multiple overlapping windows on the screen to provide the 'desktop' metaphor. Instead of typing commands, you manipulate objects in the user environment via a pointing device called a mouse.

Figure 3-3 *Workstation Screen with Overlapping Windows*

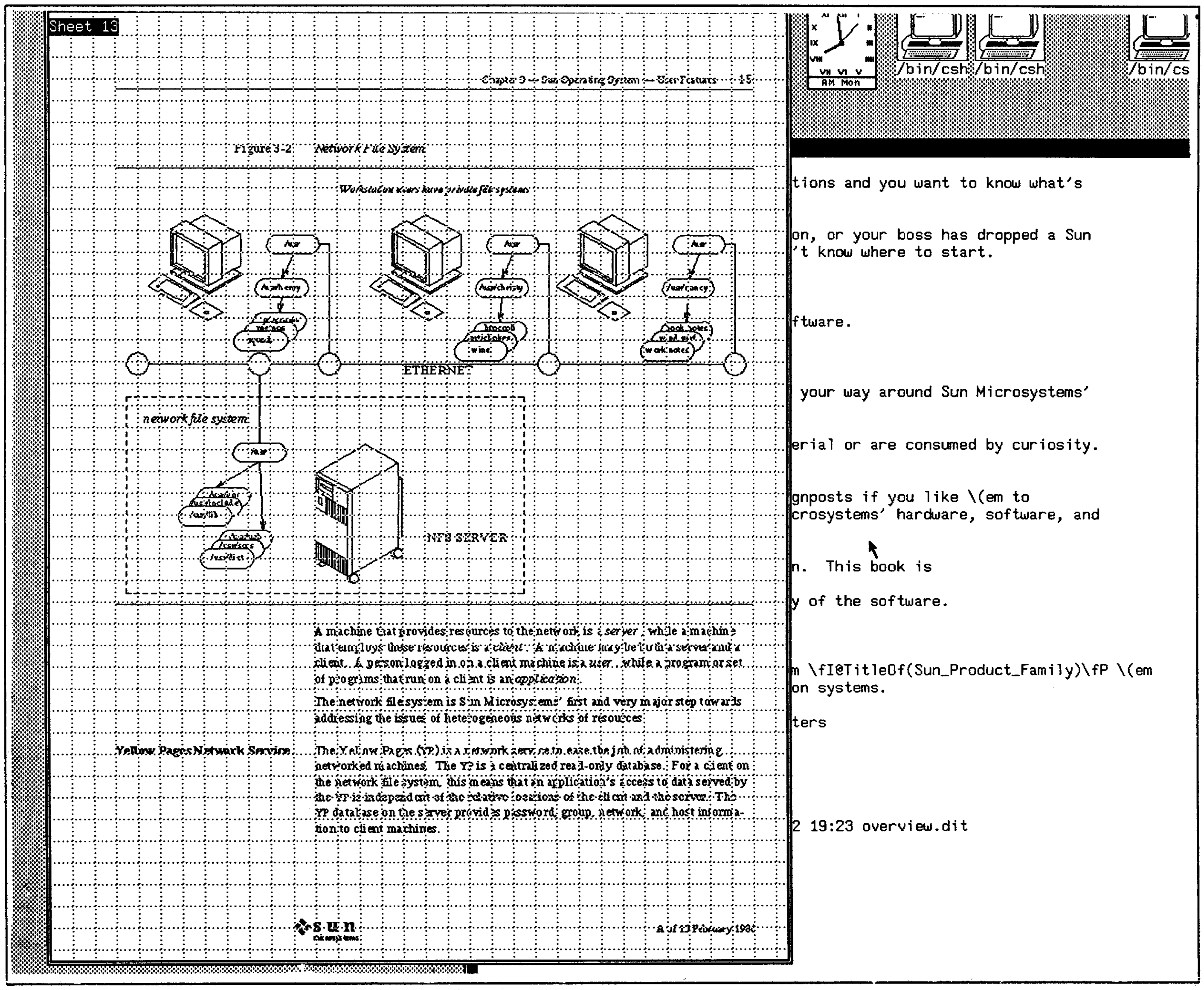


Figure 3-2 *Network File System*

Workstation users have private file systems

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

The network filesystem is Sun Microsystems' first and very major step towards addressing the issues of heterogeneous networks of resources.

Yellow Pages Network Service The Yellow Pages (YP) is a network service to ease the job of administering networked machines. The YP is a centralized read-only database. For a client on the network file system, this means that an application's access to data served by the YP is independent of the relative locations of the client and the server. The YP database on the server provides password, group, network, and host information to client machines.

```
tions and you want to know what's

on, or your boss has dropped a Sun
't know where to start.

ftware.

 your way around Sun Microsystems'

erial or are consumed by curiosity.

gnposts if you like \(em to
crosystems' hardware, software, and

n.  This book is

y of the software.

m \fI@TitleOf(Sun_Product_Family)\fP \(em
on systems.

ters

2 19:23 overview.dit
```

3.4. C-Shell — Command Interpeter

The *Shell* is the command interpeter for UNIX systems. 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 responds to user commands and arranges to run the appropriate utility software on the user's behalf.

On Version 7 UNIX systems, the 'standard' Shell is the *Bourne Shell* named for S. R. Bourne, its creator.

The C-Shell was implemented by William N. Joy at the University of California at Berkeley. The C-Shell is a powerful interactive command interpreter. Here are some of the C-Shell's major features:

- Jobs can be run in the background and brought to the foreground. The C-Shell can display the status of jobs.
- The C-Shell can maintain a *history* of user commands both during a login session and across login sessions. You can display the history list and refer to previous commands either by their event number or by the first few unique characters of the command. You can make substitutions to repeated commands.
- You can tailor your command set by using the C-Shell *alias* facility — a macro-like facility to provide synonyms for commands.
- You can tailor your command environment by using the C-Shell *profile* facilities where you can set standard aspects of the environment such as where the Shell searches for commands.
- The C-Shell can be used as a programming language — you can place sequences of commands in a file and get the command sequences executed just by typing the name of the file. The programming features of the Shell can be used to perform often complex tasks without the need to write software.

4

SunView — User Environment

4

SunView — User Environment

Sun Microsystems window-based user environment exploits the capabilities of the high-resolution bit-mapped screen in *SunView* — the Sun Visual/Integrated Environment for Workstations.

SunView is two things:

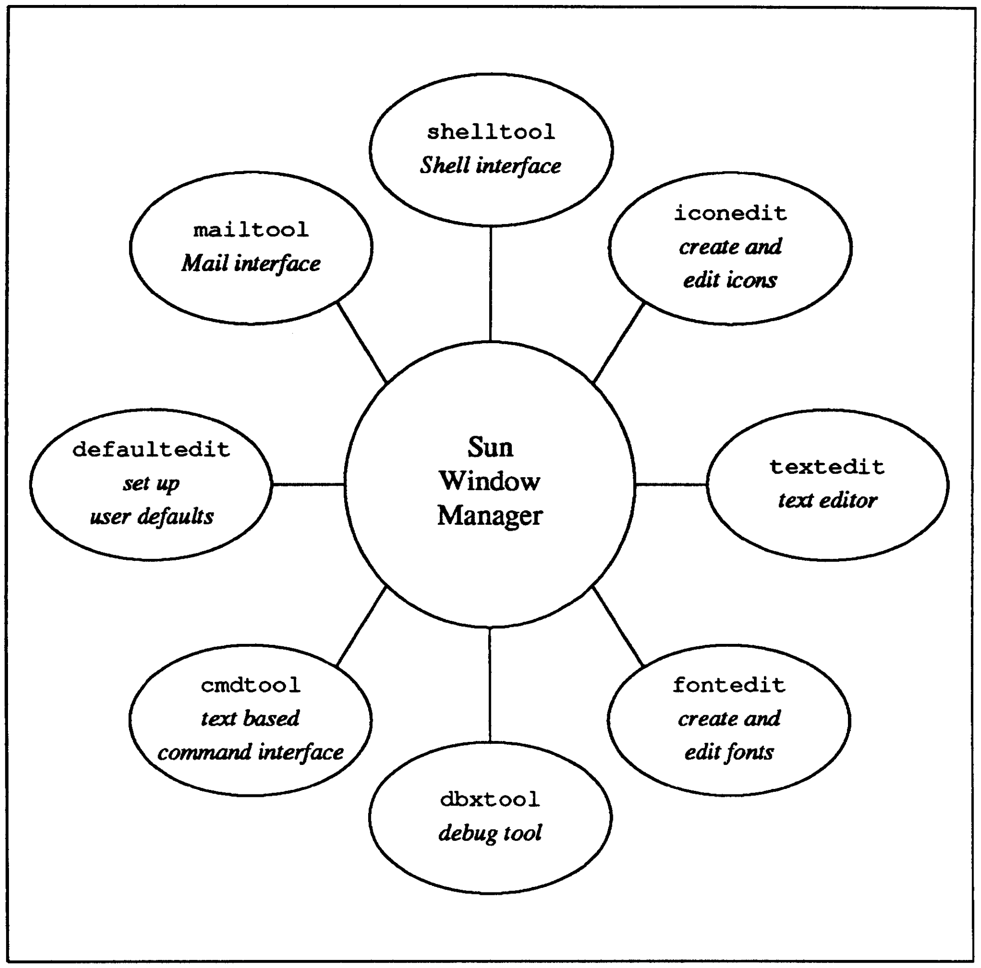
- A *user interface* accessible *via* the `suntools` package which provides multiple overlapping windows on the screen. Each window runs user tasks independent of the other windows on the screen.
- *SunGuide* — Sun General User Interface Design Environment — is a *programming interface* accessible *via* a collection of subroutine libraries. There are three major levels of abstraction in the programming interface to SunView, and a programmer can use any or all of these three layers to write applications.

This chapter concentrates on the external (user accessible) features of SunView.

Window User Interface

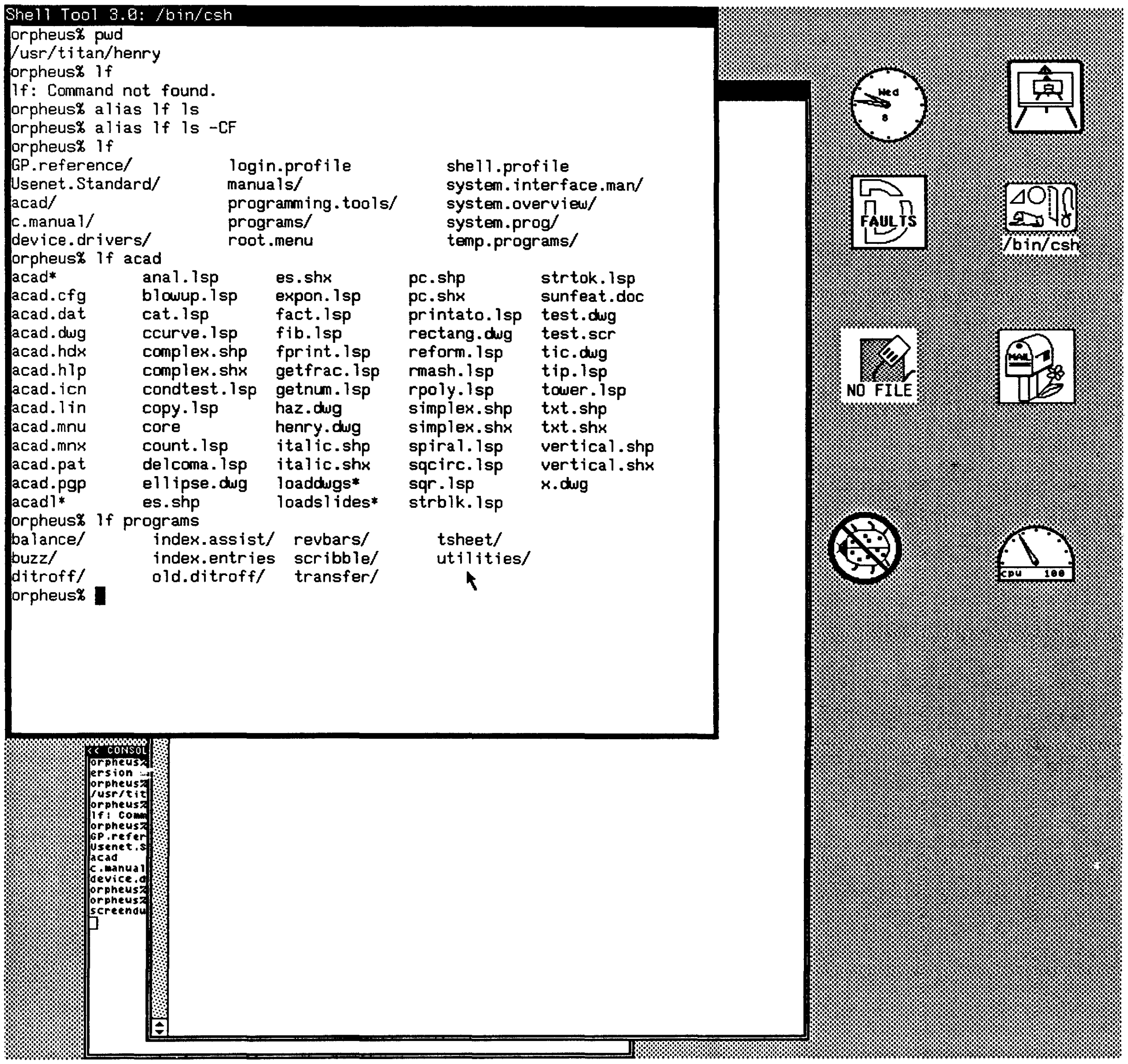
The user interface of SunView is a collection of software utilities that exploit the graphical capabilities of the Sun hardware. The *mouse* (a pointing device) is a primary mechanism for interacting with the tools. The emphasis is on 'point at what you want' rather than on 'type a command and wait for a response'. In many applications the mouse is superseding the keyboard as the communication channel. Application software displays control panels with buttons that you 'push' with the mouse, and sliders (potentiometers) that you adjust with the mouse.

Figure 4-1 *User View of the Sun Window System*



Windows occupy sections of the screen and can overlap each other. You can move windows around on the screen. You can bring a window to the top of the heap by point at it, or send a window to the bottom of the pile. You can adjust the size of a window depending on the 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. Figure 4-2 is a picture of a typical screen with three overlapping windows and some icons.

Figure 4-2 *Typical Screen with Overlapping Windows and Icons*



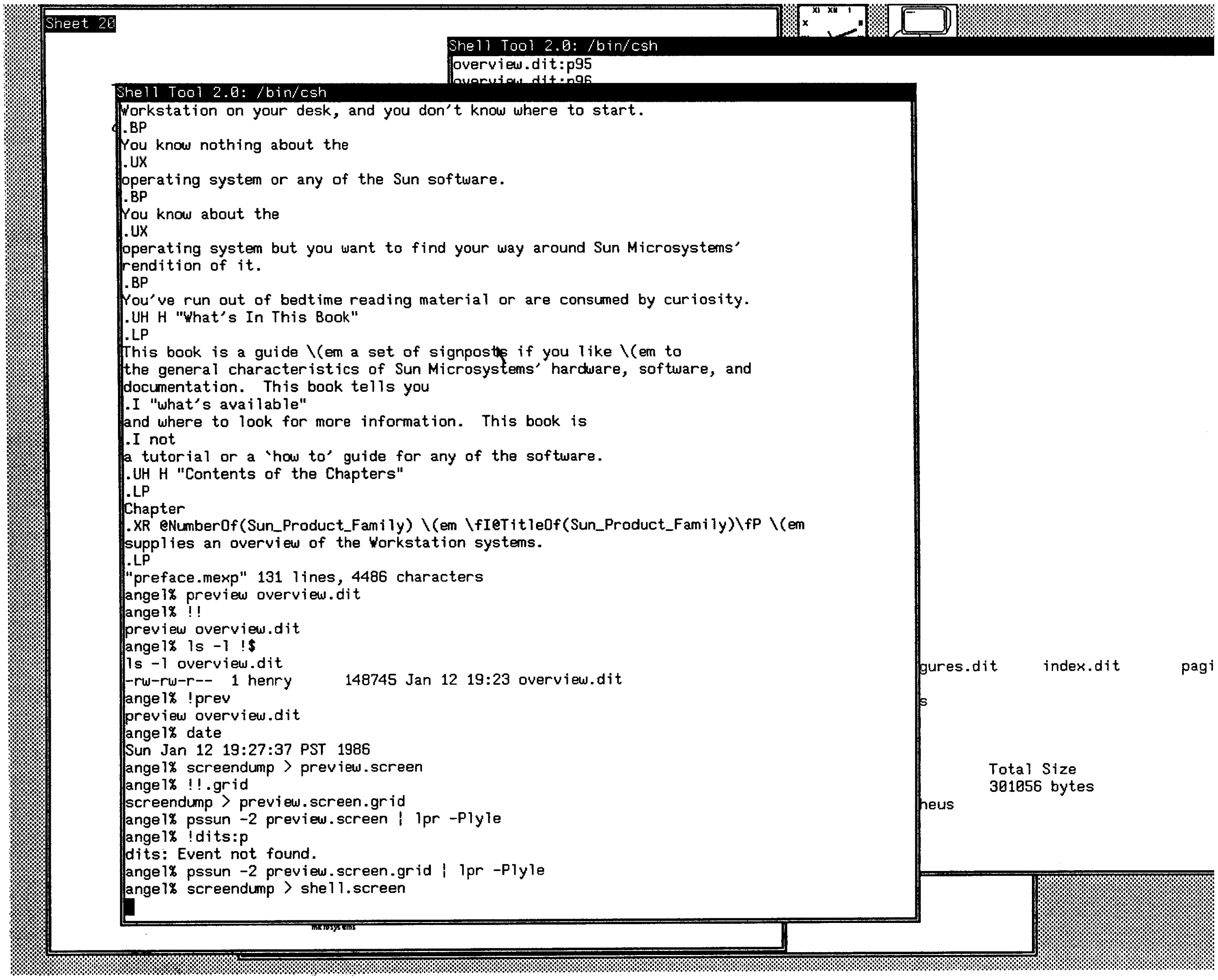
```
Shell Tool 3.0: /bin/csh
orpheus% pwd
/usr/titan/henry
orpheus% lf
lf: Command not found.
orpheus% alias lf ls
orpheus% alias lf ls -CF
orpheus% lf
GP.reference/       login.profile       shell.profile
Usenet.Standard/    manuals/            system.interface.man/
acad/               programming.tools/  system.overview/
c.manual/           programs/           system.prog/
device.drivers/     root.menu           temp.programs/
orpheus% lf acad
acad*         anal.lsp      es.shx        pc.shp        strtok.lsp
acad.cfg      blowup.lsp    expon.lsp     pc.shx        sunfeat.doc
acad.dat      cat.lsp       fact.lsp      printato.lsp  test.dwg
acad.dwg      ccurve.lsp    fib.lsp       rectang.dwg   test.scr
acad.hdx      complex.shp   fprint.lsp    reform.lsp    tic.dwg
acad.hlp      complex.shx   getfrac.lsp   rmash.lsp     tip.lsp
acad.icn      condtest.lsp  getnum.lsp    rpoly.lsp     tower.lsp
acad.lin      copy.lsp      haz.dwg       simplex.shp   txt.shp
acad.mnu      core          henry.dwg     simplex.shx   txt.shx
acad.mnx      count.lsp     italic.shp    spiral.lsp    vertical.shp
acad.pat      delcoma.lsp   italic.shx    sqcirc.lsp    vertical.shx
acad.pgp      ellipse.dwg   loaddwgs*     sqr.lsp       x.dwg
acadl*        es.shp        loadslides*   strblk.lsp
orpheus% lf programs
balance/      index.assist/  revbars/     tsheet/
buzz/         index.entries  scribble/    utilities/
ditroff/      old.ditroff/   transfer/
orpheus% █
```

You 'select' things by pointing at them with the mouse. One application is to select text in one window and 'stuff' it into the input stream of another window — this forms the basis of 'cut and paste' style editing.

See *Windows and Window-Based Tools — Beginner's Guide* for detailed descriptions on using SunView and its applications.

Using the Shells

`shelltool` is the window tool to provide a window-based interface to the standard UNIX system command interpreters (shells).



```
Sheet 20
Shell Tool 2.0: /bin/csh
overview.dit:p95
```

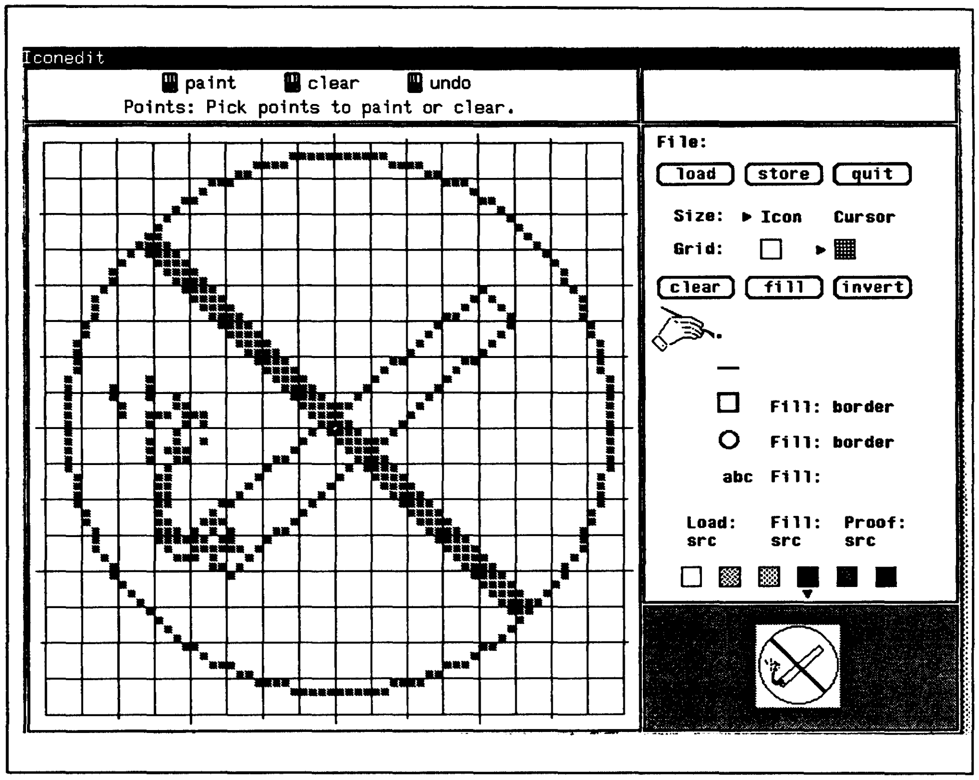
```
Shell Tool 2.0: /bin/csh
Workstation on your desk, and you don't know where to start.
.BP
You know nothing about the
.UX
operating system or any of the Sun software.
.BP
You know about the
.UX
operating system but you want to find your way around Sun Microsystems'
rendition of it.
.BP
You've run out of bedtime reading material or are consumed by curiosity.
.UH H "What's In This Book"
.LP
This book is a guide \(em a set of signposts if you like \(em to
the general characteristics of Sun Microsystems' hardware, software, and
documentation.  This book tells you
.I "what's available"
and where to look for more information.  This book is
.I not
a tutorial or a 'how to' guide for any of the software.
.UH H "Contents of the Chapters"
.LP
Chapter
.XR @NumberOf(Sun_Product_Family) \(em \fI@TitleOf(Sun_Product_Family)\fP \(em
supplies an overview of the Workstation systems.
.LP
"preface.mexp" 131 lines, 4486 characters
angel% preview overview.dit
angel% !!
preview overview.dit
angel% ls -l !$
ls -l overview.dit
-rw-rw-r--  1 henry      148745 Jan 12 19:23 overview.dit
angel% !prev
preview overview.dit
angel% date
Sun Jan 12 19:27:37 PST 1986
angel% screendump > preview.screen
angel% !!.grid
screendump > preview.screen.grid
angel% pssun -2 preview.screen | lpr -Plyle
angel% !dits:p
dits: Event not found.
angel% pssun -2 preview.screen.grid | lpr -Plyle
angel% screendump > shell.screen
```

```
gures.dit     index.dit       pagi
s

        Total Size
        301056 bytes
heus
```

Creating Cursors and Icons

`iconedit` is a window-based drawing editor for designing icons and cursors. Here is an example of `iconedit` used to draw a no-smoking sign:

Figure 4-3 *Creating an Icon with the Icon Editing Utility*

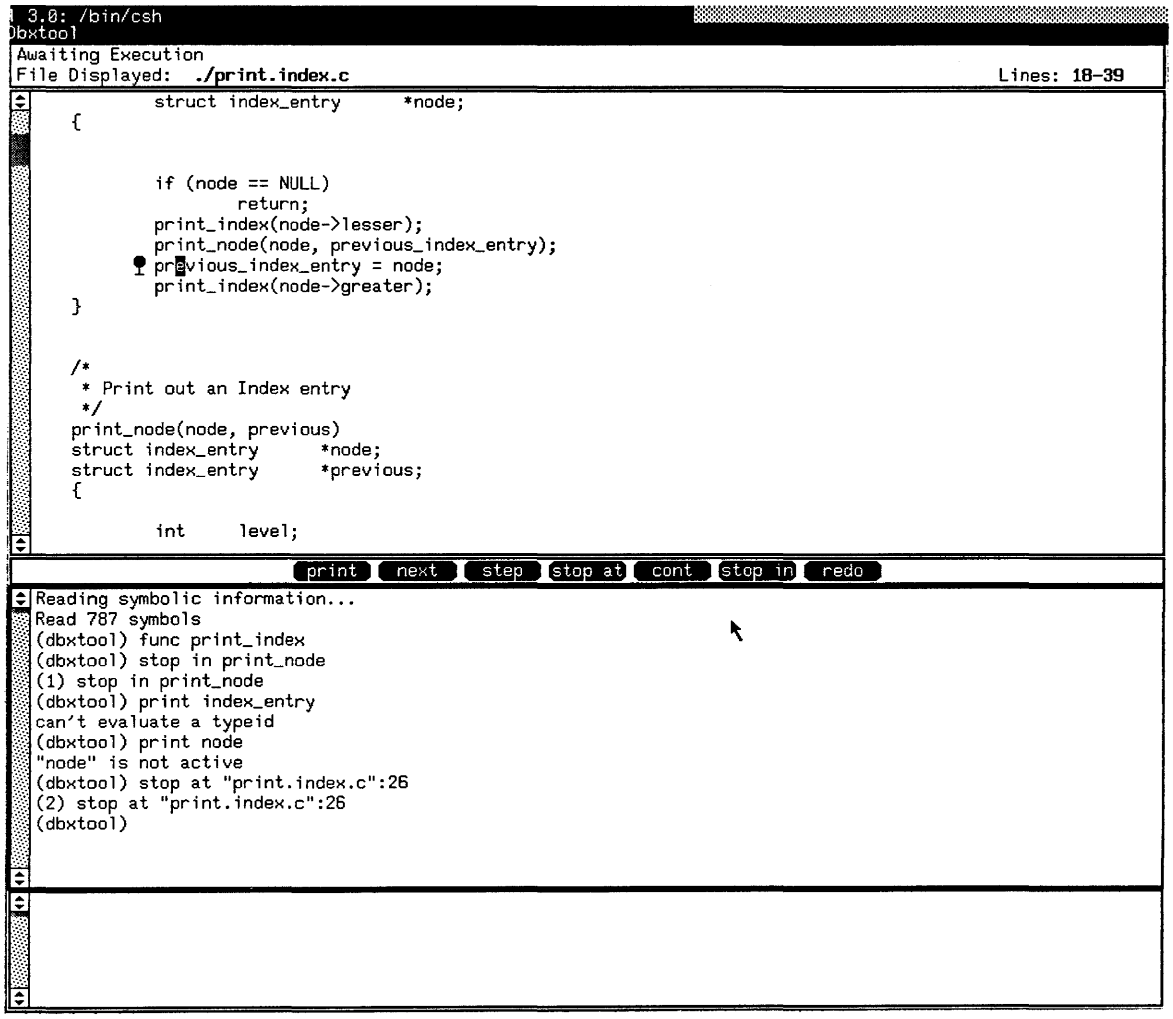


See *Windows and Window-Based Tools — Beginner's Guide* and the `iconedit(1)` manual page for further details.

Debugging Software

dbxtool is a window-based interface to the capabilities of dbx — the symbolic debug utility. Using dbxtool, you point at the object you wish to operate on (say display the value of a variable) and then you 'push a button' to get the operation done. Here is a picture of a dbxtool window with the program's source code displayed in one window and the commands displayed in another window:

Figure 4-4 *Debugging a Program Using the Interactive Debug Utility*



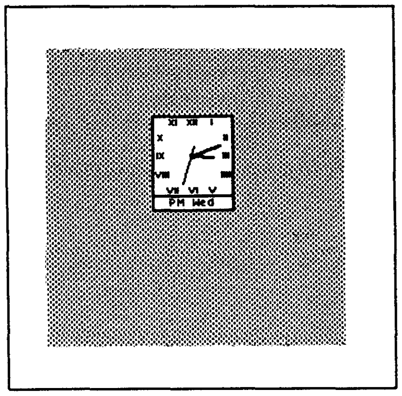
```
Awaiting Execution
File Displayed:  ./print.index.c                                  Lines: 18-39
```

```
            struct index_entry      *node;
    {


            if (node == NULL)
                    return;
            print_index(node->lesser);
            print_node(node, previous_index_entry);
            previous_index_entry = node;
            print_index(node->greater);
    }


    /*
     * Print out an Index entry
     */
    print_node(node, previous)
    struct index_entry      *node;
    struct index_entry      *previous;
    {

            int     level;
```

```
Reading symbolic information...
Read 787 symbols
(dbxtool) func print_index
(dbxtool) stop in print_node
(1) stop in print_node
(dbxtool) print index_entry
can't evaluate a typeid
(dbxtool) print node
"node" is not active
(dbxtool) stop at "print.index.c":26
(2) stop at "print.index.c":26
(dbxtool)
```

See *Debugging Tools for the Sun Workstation* and the dbxtool(1) and dbx(1) manual pages for further details.

Displaying the Time



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

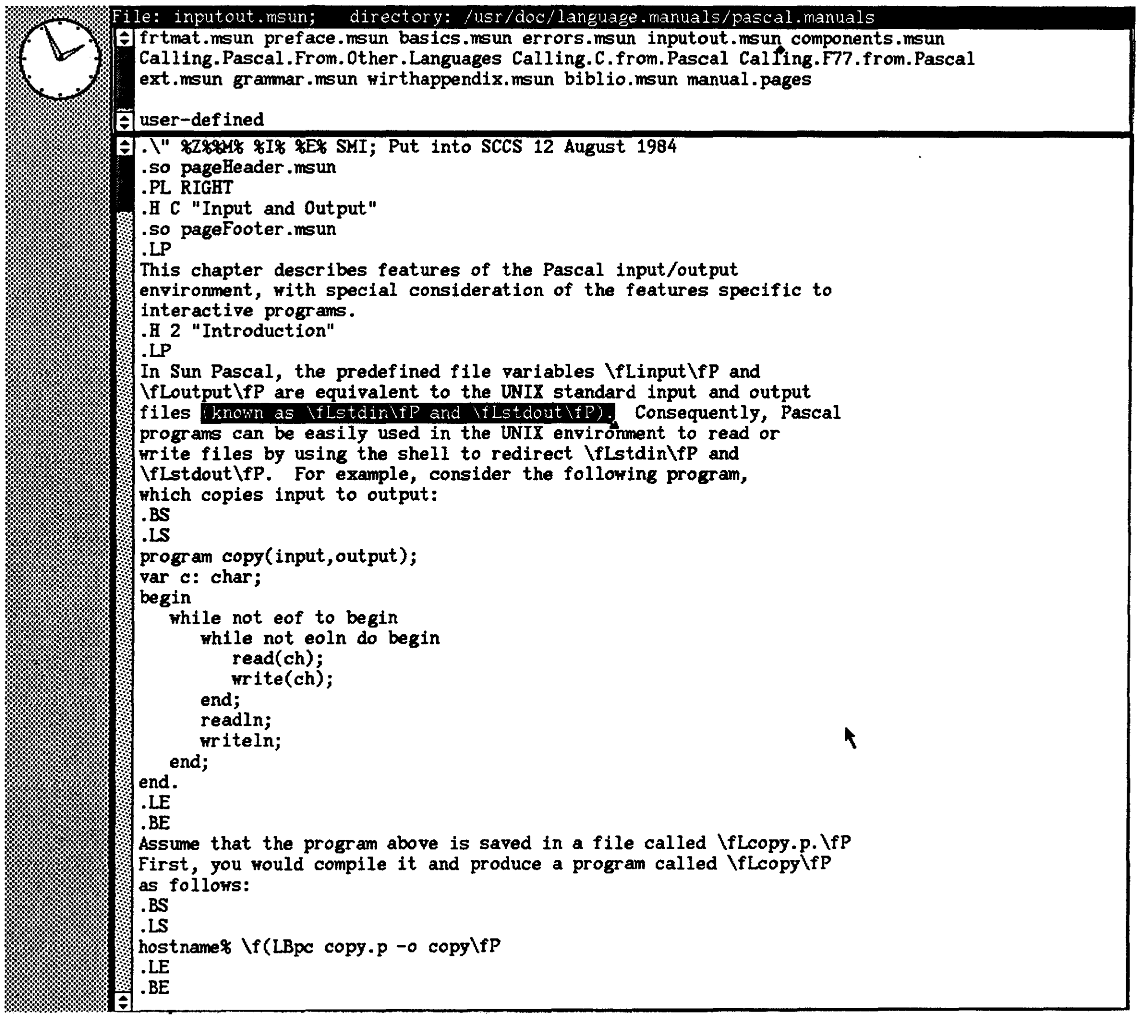
- round or square clock,
- Arabic or Roman numerals on the clock face,
- display the seconds,
- display the day of the week.

See *Windows and Window-Based Tools — Beginner's Guide* and the `clocktool(1)` manual page for further details.

Editing Text

`textedit` is a mouse-based text editing utility. There are mouse functions to select regions of text, and the function keys can be used to perform cut and paste operations.

Figure 4-5 *Text Editing Utility*



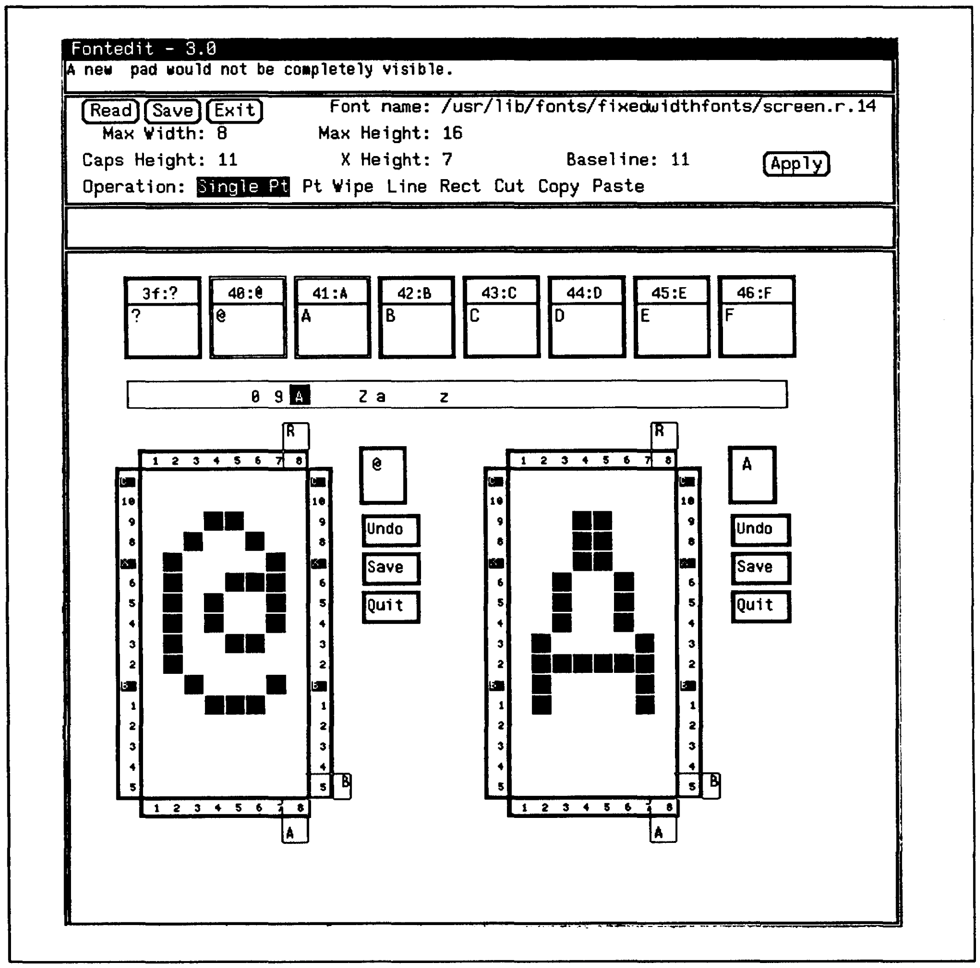
```
File: inputout.msun;   directory: /usr/doc/language.manuals/pascal.manuals
frtmat.msun preface.msun basics.msun errors.msun inputout.msun components.msun
Calling.Pascal.From.Other.Languages Calling.C.from.Pascal Calling.F77.from.Pascal
ext.msun grammar.msun wirthappendix.msun biblio.msun manual.pages

user-defined
.\" %Z%%M% %I% %E% SMI; Put into SCCS 12 August 1984
.so pageHeader.msun
.PL RIGHT
.H C "Input and Output"
.so pageFooter.msun
.LP
This chapter describes features of the Pascal input/output
environment, with special consideration of the features specific to
interactive programs.
.H 2 "Introduction"
.LP
In Sun Pascal, the predefined file variables \fLinput\fP and
\fLoutput\fP are equivalent to the UNIX standard input and output
files (known as \fLstdin\fP and \fLstdout\fP). Consequently, Pascal
programs can be easily used in the UNIX environment to read or
write files by using the shell to redirect \fLstdin\fP and
\fLstdout\fP. For example, consider the following program,
which copies input to output:
.BS
.LS
program copy(input,output);
var c: char;
begin
   while not eof to begin
      while not eoln do begin
         read(ch);
         write(ch);
      end;
      readln;
      writeln;
   end;
end.
.LE
.BE
Assume that the program above is saved in a file called \fLcopy.p.\fP
First, you would compile it and produce a program called \fLcopy\fP
as follows:
.BS
.LS
hostname% \f(LBpc copy.p -o copy\fP
.LE
.BE
```

See *Windows and Window-Based Tools — Beginner's Guide* for further details.

Creating Fonts

fontedit creates and edits fonts. You load up 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. The characters to be edited are displayed in large format and you can make pixel-by-pixel modifications to the characters.

Figure 4-6 *Font Editing Utility*



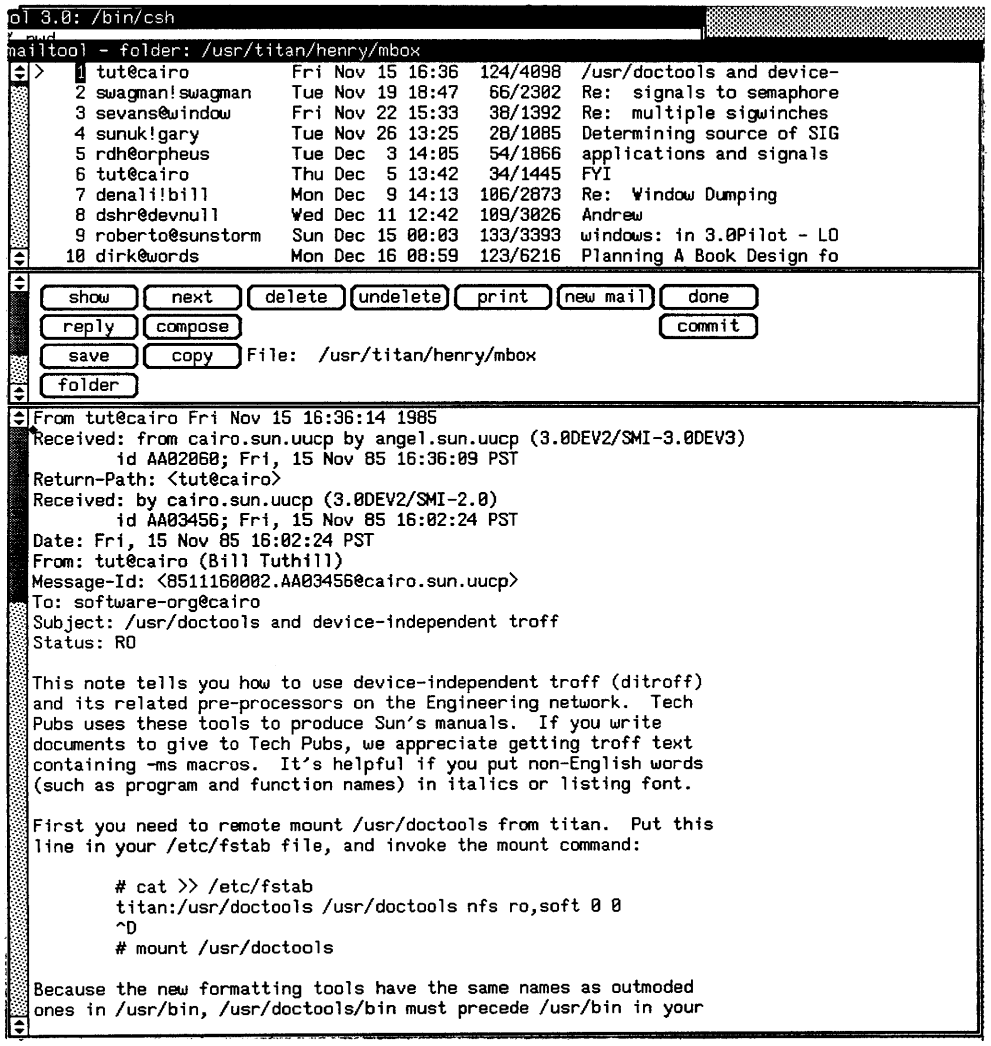
```
A new  pad would not be completely visible.
Font name: /usr/lib/fonts/fixedwidthfonts/screen.r.14
Max Width: 8          Max Height: 16
Caps Height: 11       X Height: 7          Baseline: 11
Operation: Single Pt Pt Wipe Line Rect Cut Copy Paste
```

See *Windows and Window-Based Tools — Beginner's Guide* and the fontedit(1) manual page for further details.

Reading Mail

`mailtool` is a window and mouse based interface to the standard mail reading utility. You can use the standard text facilities of SunView to compose messages and move text between messages.

Figure 4-7 *Mail Utility*



```
>    1 tut@cairo            Fri Nov 15 16:36  124/4098  /usr/doctools and device-
     2 swagman!swagman      Tue Nov 19 18:47   66/2302  Re:  signals to semaphore
     3 sevans@window        Fri Nov 22 15:33   38/1392  Re:  multiple sigwinches
     4 sunuk!gary           Tue Nov 26 13:25   28/1085  Determining source of SIG
     5 rdh@orpheus          Tue Dec  3 14:05   54/1866  applications and signals
     6 tut@cairo            Thu Dec  5 13:42   34/1445  FYI
     7 denali!bill          Mon Dec  9 14:13  106/2873  Re:  Window Dumping
     8 dshr@devnull         Wed Dec 11 12:42  109/3026  Andrew
     9 roberto@sunstorm     Sun Dec 15 00:03  133/3393  windows: in 3.0Pilot - LO
    10 dirk@words           Mon Dec 16 08:59  123/6216  Planning A Book Design fo
```

```
File:  /usr/titan/henry/mbox
```

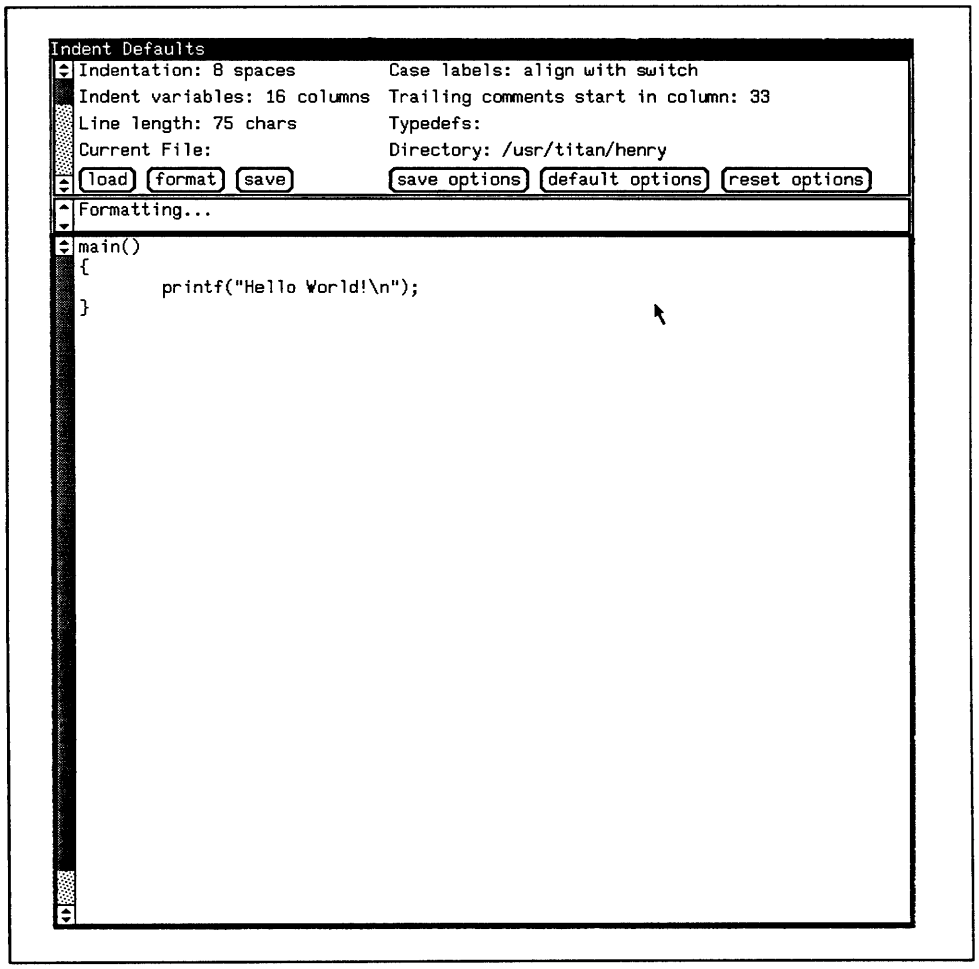
```
From tut@cairo Fri Nov 15 16:36:14 1985
Received: from cairo.sun.uucp by angel.sun.uucp (3.0DEV2/SMI-3.0DEV3)
        id AA02060; Fri, 15 Nov 85 16:36:09 PST
Return-Path: <tut@cairo>
Received: by cairo.sun.uucp (3.0DEV2/SMI-2.0)
        id AA03456; Fri, 15 Nov 85 16:02:24 PST
Date: Fri, 15 Nov 85 16:02:24 PST
From: tut@cairo (Bill Tuthill)
Message-Id: <8511160002.AA03456@cairo.sun.uucp>
To: software-org@cairo
Subject: /usr/doctools and device-independent troff
Status: RO

This note tells you how to use device-independent troff (ditroff)
and its related pre-processors on the Engineering network.  Tech
Pubs uses these tools to produce Sun's manuals.  If you write
documents to give to Tech Pubs, we appreciate getting troff text
containing -ms macros.  It's helpful if you put non-English words
(such as program and function names) in italics or listing font.

First you need to remote mount /usr/doctools from titan.  Put this
line in your /etc/fstab file, and invoke the mount command:

        # cat >> /etc/fstab
        titan:/usr/doctools /usr/doctools nfs ro,soft 0 0
        ^D
        # mount /usr/doctools

Because the new formatting tools have the same names as outmoded
ones in /usr/bin, /usr/doctools/bin must precede /usr/bin in your
```

See *Mail and Messages — Beginner's Guide* and the `mail(1)` manual page for further details.

Setting up Defaults

Many UNIX system utilities use files of user-created parameters to control their actions. `defaultedit` brings these scattered environment files into a single tool that can create and edit the parameters.

Figure 4-8 *Default Editing Utility*



```
Indent Defaults
Indentation: 8 spaces          Case labels: align with switch
Indent variables: 16 columns   Trailing comments start in column: 33
Line length: 75 chars          Typedefs:
Current File:                  Directory: /usr/titan/henry
```

```
Formatting...
```

```
main()
{
        printf("Hello World!\n");
}
```

See *Windows and Window-Based Tools — Beginner's Guide* for further details.

5

Using the Shells

5

Using the Shells

Other than the user interface provided by the Sun window system, one of the principal means of getting the system to do work for you is by typing commands to the *Shell*. The Shell is the system's *command interpreter*. Early in the development of the UNIX system, the idea arose of having a single command interpreter to provide a more coherent command interface to the user than would have been provided had every command done its own analysis of the command line.

There are two main Shells in use on the Sun operating system:

- the *C Shell* is the command language interpreter written at the University of California, Berkeley.
- The *Bourne Shell* (named after S. R. Bourne) was the standard UNIX system Version 7 Shell.

There are many similarities in the user interface the two Shells provide. The major external differences between the two Shells are that the C Shell provides job control and history. Other major differences are largely internal to the Shells in the area of their programming interface.

5.1. External Features of the Shells

Major features that both Shells provide to the user interface are:

- Analyse the command line. Run the indicated command, passing the remainder of the command line as arguments to the command. The Shell runs each command as a separate *process*. Such a process can be run 'while you wait', in the *foreground*, or a command can be run in the *background*, so that the Shell returns immediately and you can continue typing more commands.
- Redirect the standard input, standard output, and standard error files of the command as defined by the user.
- The Shell can arrange that the standard output of one process is connected to the standard input of the next process in line. Multiple processes can be connected together in this way. Such a connection is called a *pipeline*, obviating creating and managing temporary files.
- Filename expansion. Both Shells provide some *metacharacters* to indicate aggregates of files. An aggregate of files is said to be *matched* if there are metacharacters in the specification. The ? metacharacter matches any

single character in a filename. The * metacharacter matches any string of characters (including the empty string) in a filename (so that, for instance, * on its own matches all files in a directory. The [and] metacharacters form *character classes* that indicate ranges of letters or digits in filenames, so that, for instance, `[a-m]*` means all files starting with the letters a through m.

- Both Shells provide user-settable search path for finding commands. Both Shells provide can execute a user-settable profile upon login. The profile is used to 'tailor' your environment to your special needs.

5.2. Programming Features of the Shells

Both Shells can be used as a programming language. Collections of Shell commands can be placed in a file and the filename typed just like any other system command. Shell *scripts* as they are called provide the advantage that you don't need to run a compiler and loader and recompile to change things. Shell scripts frequently provide a good first cut at an application by using the Shell programming features in conjunction with connecting existing UNIX system commands together. Shell scripts provide for what is often called *rapid prototyping* or instant gratification.

Programming features that are common to both Shells include:

- Set and access variables.
- Substitute arguments from the command line.
- `if ... then ... else` constucts for conditional execution of commands.
- `case` switches for selecting groups of commands.
- `while` loops for to execute 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.
- Take special action on traps and interrupts. You use these features so that you can do 'clean up' action if a Shell process is interrupted.

5.3. The Shells

Shells available on the Sun operating system include the C Shell, the Bourne Shell, and the Remote Shell.

C Shell

`csh` is the C Shell. Users normally access `csh` when they use a Sun system — the Bourne Shell is less frequently used. The C-Shell is the command language interpreter written at the University of California, Berkeley as part of the 4.2 BSD operating system upon which Sun Microsystems bases their standard oprtating system.

The C Shell provides a flexible user interface. Its major external features are:

- A *history* facility for reissuing previous commands. The Shell maintains a history buffer of the last *n* commands, where *n* is specified by the user. You can display the history list. You can refer to commands in the history list by their event number in the list or by searching for substrings of the actual name of the command. There are many other aspects to this feature.

- Job control. You can run jobs in the background or the foreground. Foreground jobs can be sent to the background and background jobs can be brought to the foreground. You can find out what jobs are running and kill them if required. You can suspend a foreground job, do somthing else, then resume the foreground job.

- An *alias* mechanism. You can define an alias for frequently typed commands.
- A *history substitution mechanism*. In conjunction with the history mechanism described above, you can substitute parts of previous commands either single-shot, or globally.
- Optionally announce the presence of mail as mail arrives in the system.

The major programming features of the C Shell include:

- The syntax of Shell commands resembles the C programming language, whereas the Bourne Shell resembles Algol-68.
- Compose compound commands using the programming constructs described above in the general discussion.

See *Doing More with UNIX — Beginner's Guide* and the `csh(1)` manual page for further details.

Remote Shell

`rsh` is the *remote Shell* — you can execute a command on another system. The remote Shell looks just like the regular Shell in that you can connect the output of `rsh` to another process on your own machine via a pipe.

See the `rsh(1)` manual page for further details.

Bourne Shell

`sh` is the Bourne Shell, the UNIX system version 7 command language interpreter. The Bourne Shell is named after S. R. Bourne of Bell Laboratories. The Bourne Shell has most of the same external features as the C Shell. The Bourne Shell lacks the history facility, the job-control facility, and the history substitution facility. Its internal programming features are modelled after the Algol-68 programming language. One school of thought maintains that writing Shell scripts is easier with the Bourne Shell.

See *Doing More with UNIX — Beginner's Guide* and the `sh(1)` manual page for further details.

5.4. Shell Related Utilities

By themselves, the Shells do not attempt to do everything — that is against the philosophy of the system. Instead, the Shells call on the services of some other utilities that do specialized jobs.

Displaying Command Line Arguments

`echo` displays the remainder of its command line. `echo` is useful for diagnostics or prompts in Shell programs, or for inserting data into a pipeline.

See *Doing More with UNIX — Beginner's Guide* and the `echo(1)` manual page for further details.

Conditional Testing

`test` tests for use in Shell conditionals. Some of its features include:

- String comparison

- Determine 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).
- Boolean combinations of the above.

See *Doing More with UNIX — Beginner's Guide* and the `test(1)` manual page for further details.

Evaluating Expressions

`expr` evaluates expressions that appear on its command line as arguments. Some of its features include:

- String computations
- Integer arithmetic
- Pattern matching.

See *Doing More with UNIX — Beginner's Guide* and the `expr(1)` manual page for further details.

Waiting on Process

`wait` waits for termination of asynchronously running processes.

See the `wait(1)` manual page for further details.

Suspending Execution

`sleep` suspends execution for a specified time.

See the `sleep(1)` manual page for further details.

Blocking Hangups

`nohup` is mainly used for dialup lines. It runs a command immune to hanging up the telephone.

See the `nice(1)` manual page for further details.

Changing Priority

`nice` runs a command in low or high priority.

See the `nice(1)` manual page for further details.

Killing Processes

`kill` terminates specified processes or jobs.

See *Doing More with UNIX — Beginner's Guide* and the `kill(1)` manual page for further details.

Scheduling Actions

`at` schedules a one-shot action for an arbitrary time.

See *Doing More with UNIX — Beginner's Guide* and the `at(1)` manual page for further details.

Diverting Output

`tee` passes data between processes and diverts a copy into one or more files.

See *Doing More with UNIX — Beginner's Guide* and the `tee(1)` manual page for further details.

6

User Access and Commands

6 User Access and Commands

Access to the Sun operating system is via a system of *user accounts* and optional *passwords*. You can log in to your local workstation, and once logged in, you can remotely log in to other systems using `rlogin`. Access to remote hosts in geographically remote areas can be gained via `tip` described in chapter 8 — *Communications Facilities*.

Gaining Access to the System

`login` is the command to gain initial access to the system.

- Sign on as a new user
- Verify password and establish user's individual and group (project) identity
- Adapt to characteristics of a terminal
- Establish working directory
- Announce presence of mail
- Publish message of the day
- Execute user-specified profile
- Start command interpreter or other initial program.

See *Getting Started with UNIX — Beginner's Guide* and the `login(1)` manual page for further details.

Changing Your Password

`passwd` establishes your initial password or changes an existing password

- Users can change their own passwords
- Passwords are encrypted for security.

See *Getting Started with UNIX — Beginner's Guide* and the `passwd(1)` manual page for further details.

Logging Out

`logout` logs you out of the system. You can execute an optional `.logout` file containing cheery 'goodbye' messages, or making your screen dark, or some such.

See *Getting Started with UNIX — Beginner's Guide* and the `logout(1)` manual page for further details.

6.1. Useful Commands

The remainder of this chapter describes some small and sueful commands. Chapter 7 — *Working With Files* — discusses files, directories, and the commands that operate on them.

Displaying Date and Time

`date` displays today's date and time. `date` has considerable knowledge of calendric and horological peculiarities. The super-user can use `date` to set system date and time.

See *Getting Started with UNIX — Beginner's Guide* and the `date(1)` manual page for further details.

Finding Out Who is Logged In

`who` lists presently logged in users, ports and login times. Provides optional history of all logins and logouts.

The `w` command is a Berkeley enhancement that displays what command people are running as well as who is logged in.

See *Getting Started with UNIX — Beginner's Guide* and the `who(1)` and `w(1)` manual pages for further details.

Displaying What is Going On

`ps` displays active processes

- list your own or everybody's processes
- and provide optional status information: state and scheduling info, priority, attached terminal, what process is waiting for, and size.

See *Doing More with UNIX — Beginner's Guide* and the `ps(1)` manual page for further details.

Logging into Another Machine

`rlogin` logs you in to another machine in the local network.

See *Mail and Messages — Beginner's Guide* and the `rlogin(1)` manual page for further details.

Running Commands on Other Machines

`rsh` executes a shell on a remote host in the local network.

See *Mail and Messages — Beginner's Guide* and the `rsh(1)` manual page for further details.

Reminder Service

`calendar` provides an automatic reminder service for events of today and tomorrow.

See *Getting Started with UNIX — Beginner's Guide* and the `calendar(1)` manual page for further details.

Calculators — `dc`

`dc` is an interactive programmable desk calculator

- Has named storage locations as well as conventional stack for holding integers or programs
- Unlimited precision decimal arithmetic
- Appropriate treatment of decimal fractions

- Arbitrary input and output radices, in particular binary, octal, decimal and hexadecimal
- Reverse Polish operators:

  + – * /
  remainder, power, square root, load, store, duplicate, clear, print, enter program text, execute.

See *Games, Demos, and Other Pursuits — Beginner's Guide* and the `dc(1)` manual page for further details.

Calculators — `bc`

`bc` is a C-like interactive interface to the `dc` desk calculator described above

- `bc` has all the capabilities of `dc` with a high-level syntax
- Arrays and recursive functions
- Immediate evaluation of expressions and evaluation of functions upon call
- Arbitrary precision elementary functions exp, sin, cos, atan
- Go-to-less programming.

See *Games, Demos, and Other Pursuits — Beginner's Guide* and the `bc(1)` manual page for further details.

7

Working With Files

7

Working With Files

A *file* is the principal vehicle for organizing and operating on information in the system. The *hierarchical file system* is one of the Sun operating system's major strengths.

What is a File?

A *file* is a collection of data in some container somewhere. A file has several *attributes* in the Sun operating system — the major and most important attribute being the *name* of the file. Other attributes include the access permissions, size, the date and time the file was created, and the date and time the file was last changed.

What is a Directory?

A file resides in a *directory*. A directory is simply a special kind of file that has the property of being able to 'contain' other files and directories. Since a directory can contain other directories, the hierarchy can in principle extend to unlimited depth.

The Sun operating system attempts to treat 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.

Manipulating Files and Directories

There are two broad classes of utilities that operate on files and directories:

- One class of programs creates, removes, and renames files and directories.
- The other class of programs copies and manipulates the contents of files and directories.

In general, you might think of the first class of programs as manipulating the attributes of files and directories, while the second class of programs is for manipulating the data stored in the containers.

Access Permissions

When you create a directory or file, one of its attributes is its access permissions — who can access the directory or file, and what operations they can perform. The access permission are called the *mode* of the file in UNIX system jargon. Three types of users are recognized as being able to 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.

- The public — everybody else other than the owner and the members of the owner's group.

For each type of users, there are three kinds of access to a directory or file, namely:

read	the user can look at the contents of the file or can look inside the directory,
write	the user can change the contents of the file or can create other directories or files inside a directory, or
execute	the user can type the name of the file as a command to the system or can perform certain operations that involve traversing through the directory.

Working Directory

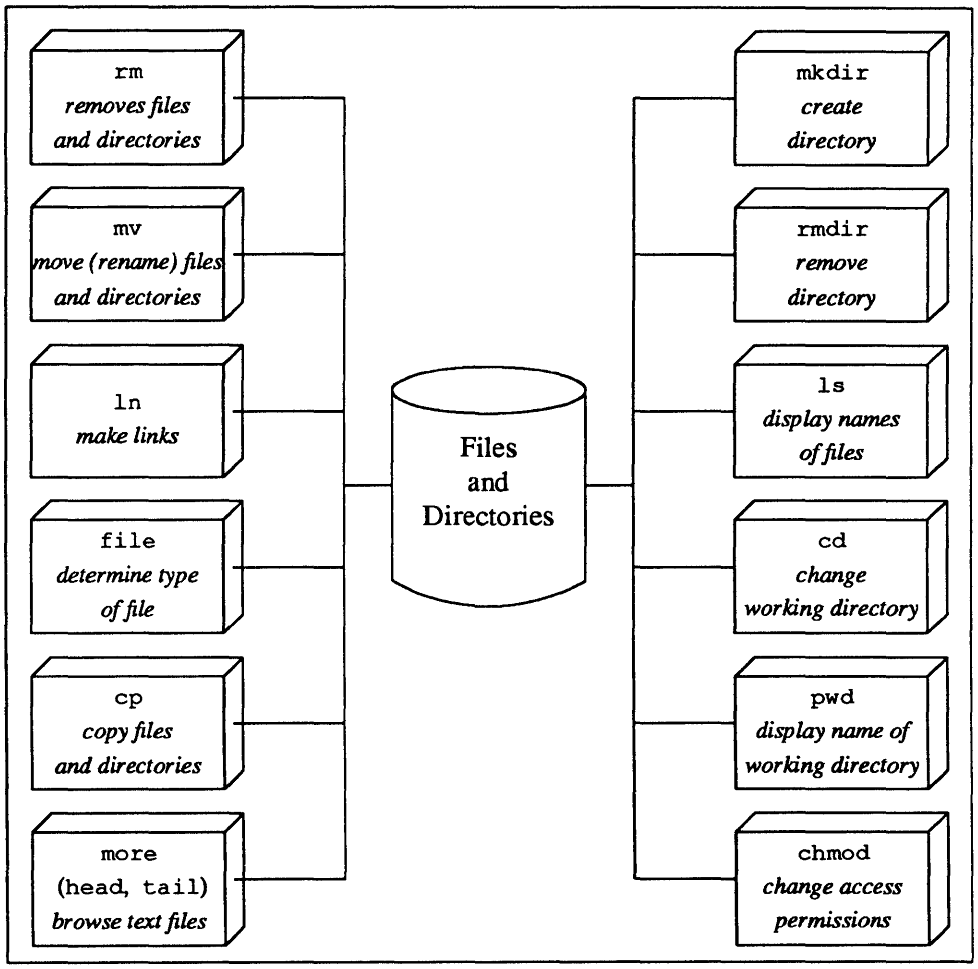
You can change directory so that you are 'positioned' at a given place in the file system hierarchy. The place where you are positioned is called the *current directory* or the *working directory*.

Home Directory

When your new user account is created, you are also given an initial working directory. You are 'positioned' at this directory when you log in. This initial directory is called your *home* directory.

Figure 7-1 below shows the commonly used commands that operate on files and directories.

Figure 7-1 *Common Operations on Files and Directories*



7.1. Managing Files and Directories

A simple file is created simply by mentioning its name in some operation such as `cat` (described later), or using one of the system text editors described in chapter 9. A directory must be created by a special command.

Finding Out What Files You Have

`ls` displays the names of files and directories. A plain `ls` command just displays a list of files sorted in alphabetical order. You can alter the display in may ways:

- reverse alphabetical order
- by time of last access
- mark the display showing directories distinct from files and mark executable (program) files.
- optional information via a 'long listing' format — size, owner, group, date last modified, date last accessed, permissions, i-node number.

See *Doing More with UNIX — Beginner's Guide* and the `ls(1)` manual page for further details.

Changing Working Directory

You use the `cd` command to change your working directory ('position' yourself) to a specified place in the directory hierarchy.

If you type a `cd` command without any specified directory name, you get back to your home directory.

See *Doing More with UNIX — Beginner's Guide* and the `cd(1)` manual page for further details.

Finding Out Where You Are

When you need an answer to the question 'where am I?' in the directory hierarchy, you use the `pwd` (print name of working directory) to display the name of your working directory.

See *Doing More with UNIX — Beginner's Guide* and the `pwd(1)` manual page for further details.

Creating a New Directory

`mkdir` makes a new directory.

See *Doing More with UNIX — Beginner's Guide* and the `mkdir(1)` manual page for further details.

Removing a Directory

`rmdir` removes a directory from the file system. The directory must be empty (contain no files or other directories) before it can be removed.

See *Doing More with UNIX — Beginner's Guide* and the `rmdir(1)` manual page for further details.

Changing Access Permissions

`chmod` changes the *mode* (the access permissions) on one or more files. Only the owner of the file, or the super-user, can use the `chmod` command.

See *Doing More with UNIX — Beginner's Guide* and the `chmod(1)` manual page for further details.

Changing Group Ownership

Users can be placed together in *groups* (loosely associated with a project) in the system. When files are created they are associated with a group as well as an owner. `chgrp` changes the group (project) to which a file belongs.

See *Doing More with UNIX — Beginner's Guide* and the `chgrp(1)` manual page for further details.

Moving and Renaming

`mv` moves a file or files from one place to another in the directory hierarchy. `mv` has the side effect of being able to rename files or directories. Why is this? You can move a file from one place to another — the destination name can be different from the source name, so moving a file with different source and destination names within the same directory effectively renames the file without 'really' moving it. `mv` can also move whole directory hierarchies from place to place in the file system.

See *Doing More with UNIX — Beginner's Guide* and the `mv(1)` manual page for further details.

Making Links

You can make a 'link' or an 'alias' to an existing file using the `ln` command.

- *Hard* links can only be created within a file system and make a real physical link in the structure.
- *Symbolic* links are more like a macro or string substitution and can span file systems. The difference between the two kinds of links becomes very apparent when removing files, as descibed below.

See *Doing More with UNIX — Beginner's Guide* and the `ln(1)` manual page for further details.

Removing Files and Directories

`rm` removes files (and directories with special options). The `rmdir` command removes directories.

- Only the *name* of the file goes away if any other names are linked to the file — this is an important effect of symbolic links as discussed above — you can end up with a symbolic link that doesn't point to anything.
- `rm` can step through a directory deleting files interactively — asking you if you want to delete the file
- With a special `-r` (recursive) option, `rm` can delete entire directory hierarchies.

See the `rm(1)` and `rmdir(1)` manual pages for further details.

Finding Specified Files and Directories

`find` prowls the directory hierarchy finding every file that meets specified criteria. Any directory may be considered to be the root or starting place for the search. `find` can be asked to perform specified operating system commands on each file that matches the specified criteria. Criteria include:

- 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,
- Boolean combinations of above.

See the `find(1)` manual page for further details.

Display Statistics of File System

`df` displays the amount of free space on file systems.

`du` displays a summary of total space occupied by all files in a hierarchy.

See the `df(1)` and `du(1)` manual pages for further details.

7.2. File Manipulation Facilities

Commands described in the last section concentrated on creating directories and files and manipulating their attributes. Now we get to some of the command for manipulating the contents of files and directories.

Determining the Type of a File

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

- ASCII text — a plain text file.
- Directory — a repository of other files.
- Troff, nroff, or eqn input text — input for one of the text formatting packages described in chapter 15.
- C program text — Source code for the C programming language. `file` can also perform this determination for FORTRAN program text.
- Executable file (one that has been compiled by one of the compilers)

See the `file(1)` manual page for further details.

Concatenating Files

`cat` is one of the simplest UNIX system commands. `cat` stands for 'catenate' and its basic operation is to concatenate one or more files together (join them end to end) and place the result onto the standard output. `cat` has a variety of everyday uses:

- Inserting data into a pipeline,
- Can optionally display non-printing characters,
- Can optionally number lines,
- Buffering output that comes in dribs and drabs.

`cat` works on any file regardless of contents.

See the `cat(1)` manual page for further details.

Copying Files and Directories

`cp` copies files (and optionally whole directory hierarchies) from place to place.

- `cp` can copy a set of files to a directory
- `cp` works on any file regardless of contents
- `cp` used with the `-r` (recursive) option can copy entire directory hierarchies.

See the `cp(1)` manual page for further details.

Remote Copy

`rcp` is *remote copy* and is used to copy files and directories from other machines in the local network.

See the `rcp(1)` manual page for further details.

Browsing Through File

`more` is a one-way file browser.

- `more` takes into account the size characteristics of your terminal and displays a page of a file at a time.
- You scroll forward a page or a line at a time by typing keys.
- You can skip forward to selected patterns in the file.

See the `more(1)` manual page for further details.

Binary Compare

`cmp` performs binary comparison on a pair of files. `cmp` reports the first place that it finds a difference in the data. Other facilities described in chapter 9 perform more detailed comparisons on the differences between files.

See the `cmp(1)` manual page for further details.

Displaying Beginning or End of File

head displays the first *n* lines of input.

tail displays the last *n* lines of input.

See the head(1) and tail(1) manual pages for further details.

Splitting a File

split carves a file into a number of pieces of a size specified by the user. You may need to do this at times, especially when dealing with some older utilities that have limitations on the number of lines they can digest at one time.

See the split(1) manual page for further details.

Converting Data Formats

dd translates physical file formats for exchanging data with foreign systems.

See the dd(1) manual page for further details.

Checksum a File

sum sums the words of a file, providing convenient checksum.

See the sum(1) manual page for further details.

7.3. Summary of File and Directory Commands

Here is an alphabetical list of the file and directory commands in this chapter.

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

Program Name	*Function*
cat	concatenate files
cd	change working directory
chgrp	change group
chmod	change access permissions
cmp	binary compare 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 head of file
ln	make links to file
ls	display file and directory names
mkdir	create directory
more	page through file
mv	more (rename) file
pwd	display name of working directory
rcp	remote copy files
rm	remove file
rmdir	remove directory
split	split file
sum	checksum file
tail	display tail of file

8

Communications Facilities

8

Communications Facilities

The Sun system provides an electronic mail facility, access to the USENET network, and facilities for transferring files to and from remote machines. Communications facilities are in four broad categories:

- Communication between users on a single time-sharing machine.
- Communication between users on different machines in the local network.
- Communication between users on different host machines in geographically distributed locations.
- Facilities to access remote machines.

The first two categories can be considered roughly equivalent since the local network facilities makes such acess transparent in many cases.

8.1. Local Communications Facilities

You can communicate with other users on the same host or on other hosts in the local network in a variety of ways.

Talking Directly with Another User

`write` establishes direct workstation or terminal communication with another user on the same machine.

`talk` establishes direct workstation or terminal communication with another user on a different machine in the local network.

`mesg` inhibits receipt of messages from `write` and `talk`.

See *Mail and Messages — Beginner's Guide* and the `write(1)`, `talk(1)`, and `mesg(1)`, manual pages for further details.

Sending and Receiving Mail

`mail` is an electronic mail system that can send messages to users on the smae machine, on another machine in the local network, and to remote machines (using the capabilities of `uucp` described below).

- Send a message to one or more users
- Read and dispose of each message individually
- The presence of mail is announced by `login` and optionally by `csh`
- Save messages in files or forward them
- Support for items such as 'Subject:' and 'Cc:' fields.

See *Mail and Messages — Beginner's Guide* and the `mail(1)` manual page for further details.

8.2. Remote Communications Facilities

When you want to access machines outside of the local network you must use one of the facilities described here.

Accessing Remote Machines Directly

`tip` establishes full-duplex connection for logging in to remote UNIX systems via dialup lines

- provide transparent interface to remote machine
- transmit files
- take remote input from local file or put remote output into local file.

See the `tip(1)` manual page for further details.

Transferring Data Between UNIX Systems

`uucp` performs spooled file transfers between two UNIX machines

- provide automatic queuing until line becomes available and remote machine is up
- copy between two remote machines.

In general, the facilities of `uucp` are not used directly by users but are there are a service for programs such as the mail system to use. See *System Administration for the Sun Workstation* and the `uucp(1)` manual page for further details.

9

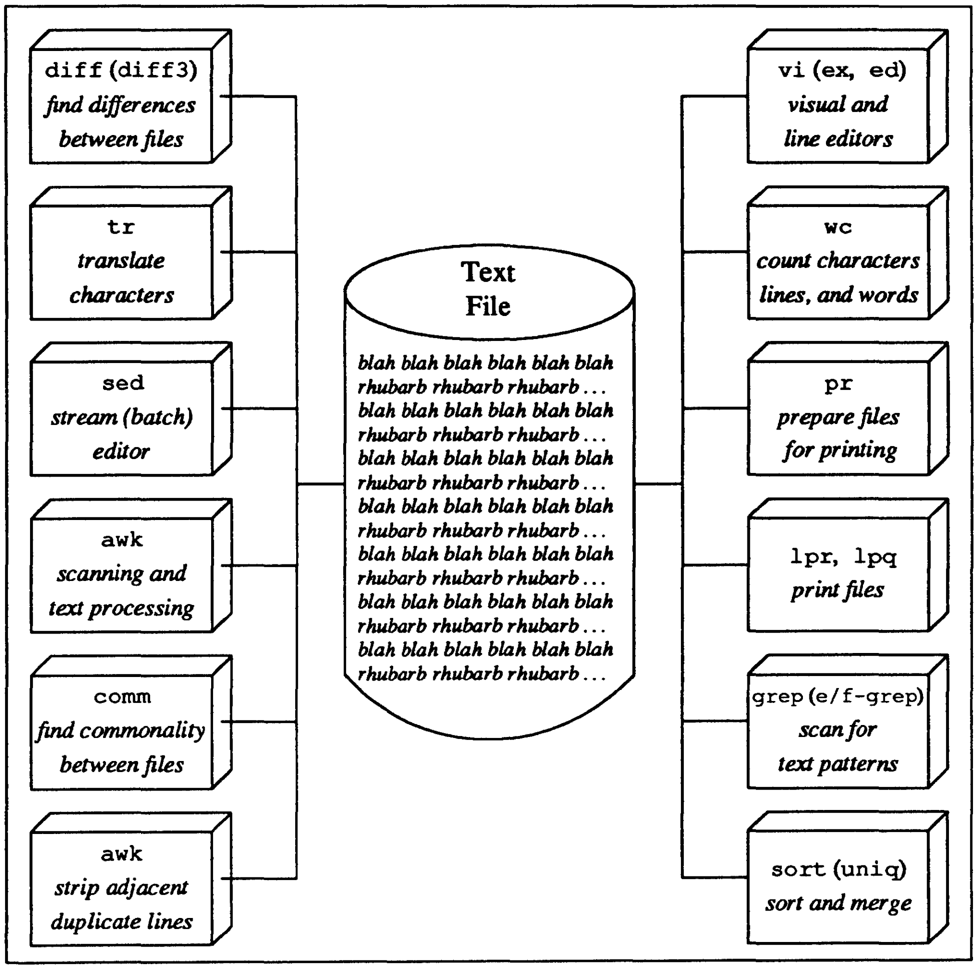
Text Editing and Text Processing

9

Text Editing and Text Processing

Text processing has been one of the strong areas in the UNIX system. Given that you have a file full of text, there are many utilities to do useful work for you. The diagram shows a quick summary of some of the common text editing and processing utilities.

Figure 9-1 *Commonly Used Text Processing Utilities*



diff (diff3)
find differences between files
tr
translate characters
sed
stream (batch) editor
awk
scanning and text processing
comm
find commonality between files
awk
strip adjacent duplicate lines
Text File
blah blah blah blah blah blah rhubarb rhubarb rhubarb ...
vi (ex, ed)
visual and line editors
wc
count characters lines, and words
pr
prepare files for printing
lpr, lpq
print files
grep (e/f-grep)
scan for text patterns
sort (uniq)
sort and merge

The following sections describe the functions and abilities of the text processing software in more detail and provide pointers for further information.

9.1. Printing Text Files

While `lpr` is designated as the utility for sending files to the printer, `lpr` by itself does no transformations on the text whatsoever and place no interpretations on the data in the text file — the UNIX system doesn't have the notion of 'carriage-control' characters as in other systems. `pr` (described below) can be used to *pr*epare a file for printing via `lpr`.

Preparing Files for Printing

`pr` prepares files for printing by a printer program — usually `lpr`. `pr` is a simple but flexible utility. `pr` can:

- Place a title, date, and page number on every page.
- Arrange text into multiple columns.
- Merge several files into multiple columns in parallel.
- Specify the number of columns on a page and the number of lines on a page.

See *Using UNIX Text Utilities on the Sun Workstation* and the `pr(1)` manual page for further details.

Printing Files Offline

`lpr` is the line printer control program for spooling arbitrary files to printer for off-line printing. One option to `lpr` can call up the `pr` program described above to paginate the text being 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`).

See *Using UNIX Text Utilities on the Sun Workstation* and the `lpr(1)`, `lpq(1)`, and `lprm(1)` manual pages for further details.

9.2. Interactive and Non-interactive Text Editors

Text editors and other text manipulation software comprise a strong part of the facilities offered in the UNIX system. All text processing utilities lean heavily on the use of *regular expressions* to specify *text patterns* for searching. Major capabilities of searching via regular expressions include:

- match single characters or strings of characters
- match *any arbitrary character*
- match *classes of characters*, for instance, 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*.
- match alternative patterns — this is 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,

Visual Editor

`vi` is the principal UNIX system tool for creating and editing text. `vi` is a screen-oriented display editor, providing 'what you see is what you get editing' for either line-oriented or full screen terminals. Capabilities include regular expression searching and user-specific settings.

See *Using UNIX Text Utilities on the Sun Workstation* and the `vi(1)` manual page for further details.

Line Oriented Editor

`ex` is the line-oriented parent of `vi`, based on the original `ed` editor. `ex` subsumes all functions of `ed`.

See *Using UNIX Text Utilities on the Sun Workstation* and the `ex(1)` manual page for further details.

Original UNIX Line Oriented Editor

`ed` was the original line-editor for the UNIX system. `ed` has been superseded by more powerful display editors such as `vi`, but `ed` still has its place, most notably some utilities generate `ed` commands for automatic editing. `ed` is an interactive context editor providing random access to all lines of a file. Main features are:

- find lines by number or pattern — patterns may include specified characters, don't care characters, choices among characters, repetitions of these constructs, beginning of line, and end of line
- 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
- optional encryption for extra security.

See *Using UNIX Text Utilities on the Sun Workstation* and the `ed(1)` manual page for further details.

Batch or Stream Editing

`sed` is a non-interactive stream text editor version of `ed` for processing large files

- `sed` can perform a sequence of editing operations on each line of an input stream of unbounded length
- Lines may be selected by address or range of addresses
- `sed` provides control flow and conditional testing, multiple output streams, and multi-line capability.

See *Using UNIX Text Utilities on the Sun Workstation* and the `sed(1)` manual page for further details.

9.3.. Information Processing and Text Manipulation

Information Processing in UNIX includes utilities originally intended for statistical text processing. Some categories include:

- Counting lines, words, and characters in a file
- Searching for specific patterns in a file
- Transliterating characters
- Sorting the contents of a file

Counting Things in Files

`wc` counts the lines, 'words' (blank-separated strings) and characters in a file.

See *Using UNIX Text Utilities on the Sun Workstation* and the `wc(1)` manual page for further details.

Translating Characters

`tr` transliterates characters in a file:

- Do one-to-one character translation according to an arbitrary code
- May coalesce selected repeated characters
- May delete selected characters.

See *Using UNIX Text Utilities on the Sun Workstation* and the `tr(1)` manual page for further details.

Scanning for Text Patterns

`grep` is one of a family of programs that search for patterns in a file.

Whence 'GREP'?

The name `grep` stands for 'Global Regular Expression Print' and is derived from the old `ed` line editor global command where you'd type

```
g/RE/p
```

to print every line that contains the specified *RE* (regular expression).

- Display all lines in a file that satisfy a regular expression.
- Display all lines that fail to match
- Display count of matches
- Display first match in each file.

Three flavors of 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.

See *Using UNIX Text Utilities on the Sun Workstation* and the `grep(1)`, `egrep(1)`, and `fgrep(1)`, manual pages for further details.

Sorting Files

`sort` is the main general-purpose sort utility available on UNIX. `sort` sorts or merges ASCII files line-by-line. This program is radically different from the 'traditional' sort-merge utilities found on other computer systems in that `sort` does not expect its input in fixed width fields starting in specific columns. Instead, `sort` breaks lines of a file into *fields* separated by whitespace (you can specify the field delimiter). The 'punched card mentality' doesn't have a place here. Some of the features that `sort` offers are:

- No limit on input size
- Sort up or down
- Sort lexicographically or on numeric key

- Multiple keys located by delimiters or by character position
- May sort upper case together with lower into dictionary order
- Optionally suppress duplicate data.

Removing Successive Duplicate Lines

`uniq` collapses successive duplicate lines in a file into one line. This facility is also obtained in the `sort` utility, but it is useful to have the facility available as a separate function. `uniq` can report on lines that were originally unique, duplicated, or both, and can display a redundancy count for each line.

Sorting Topologically

`tsort` is a topological sort that converts a partial order into a total order.

See *Using UNIX Text Utilities on the Sun Workstation* and the `sort(1)`, `uniq(1)`, and `tsort(1)` manual pages for further details.

Scanning Patterns and Processing Text

`awk` is a pattern scanning and processing language that makes it easy to specify many data transformation and selection operations. `awk` can be thought of as a 'programmable report generator'. `awk` contains all the capabilities of the `grep` family of pattern scanners, but `awk` can be programmed to manipulate the data from the text file and perform computations as well.

Whence 'AWK'?

The name `awk` does not have any magical significance but is the initial letters of the authors — 'Aho, Weinberger, and Kernighan'. Cutesy, cutesy, yes?

- `awk` searches its input file for specified *patterns* and performs *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.
- Data is treated as string or numeric as appropriate.
- `awk` breaks its input into *records* and *fields* — fields are referenced as *variables*. 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 can be formatted as desired. Output can be directed to multiple output streams.

See *Using UNIX Text Utilities on the Sun Workstation* and the `awk(1)` manual page for further details.

Displaying Differences Between Files

`diff` compares two files and report differences. `diff` is so named because it is a *differential* file comparator — it does more than just report that there is a mismatch. `diff` can also:

- Report line changes, additions and deletions necessary to bring two files into agreement
- May produce an editor script to convert one file into another — the editor script so generated 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.

See *Using UNIX Text Utilities on the Sun Workstation* and the `diff(1)` and `diff3(1)` manual pages for further details.

Finding Commonality Between Files

`comm` identifies common lines in two sorted files. Output in up to 3 columns shows lines present in first file only, present in both, and/or present in second only.

See *Using UNIX Text Utilities on the Sun Workstation* and the `comm(1)` manual page for further details.

Checking Spelling

`spell` looks for spelling errors by comparing each word in a document against a 25,000-word list that includes proper names

- Handles common prefixes and suffixes
- Collects words to help tailor local spelling lists.

See *Using UNIX Text Utilities on the Sun Workstation* and the `spell(1)` manual page for further details.

Searching for Words in a Sorted File

`look` searches for words in sorted file (usually a dictionary) that begin with a specified prefix.

See *Using UNIX Text Utilities on the Sun Workstation* and the `look(1)` manual page for further details.

Encrypting and Decrypting Files

`crypt` encrypts and decrypts files for greater security.

See *Using UNIX Text Utilities on the Sun Workstation* and the `crypt(1)` manual page for further details.

Joining Records in File

`join` combines two files by joining records that have identical keys.

See *Using UNIX Text Utilities on the Sun Workstation* and the `join(1)` manual page for further details.

9.4. Summary of Text Processing Utilities

Here is an alphabetical list of the text utilities described in this chapter.

Table 9-1 *Summary of Editing and Text Processing Programs*

Program Name	*Function*
`awk`	Scan patterns and process text
`comm`	Find commonality between files
`crypt`	Encrypt
`decrypt`	Decrypt files
`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

10

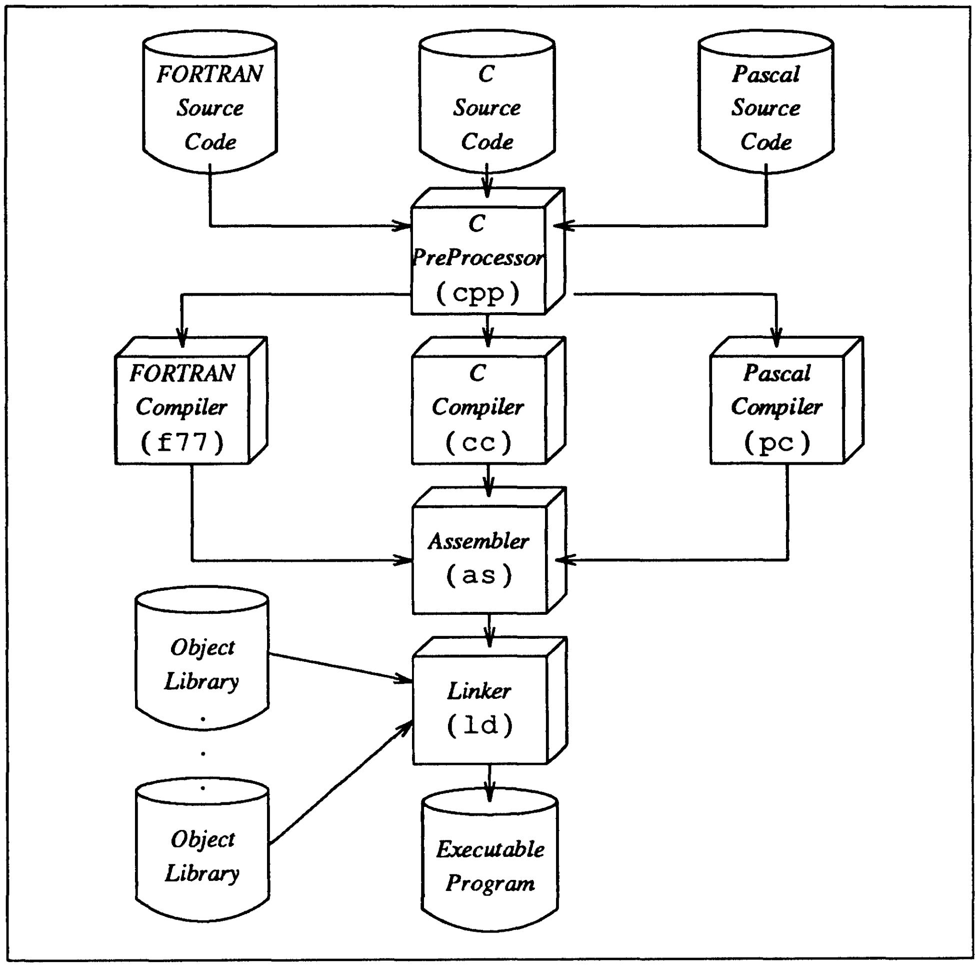
Programming Languages

10

Programming Languages

Sun Microsystems supports compilers for C, FORTRAN 77, and Pascal. Sun Microsystems have made major improvements to the quality and performance of all three languages. Programming at the assembler language level is also supported. The link editor combines object code modules into final executable programs as shown in the diagram below.

Figure 10-1 *Flow from Source Code to Compiled Program*



Chapter 11 contains information about the various tools available to assist the software development process.

10.1. C Programming Language

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

C is a general purpose language designed for structured programming:

- Generalized initialization, block structure, long integers, unions, and explicit type conversions
- Enhanced to take 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 intended 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
- Runtime library gives access to all system facilities.

Compiling C Programs

`cc` is the C compiler which can compile and/or link edit programs in the C language. The C compiler has been enhanced to take arbitrary length variable names, allowing readable names and supporting interfaces with other languages such as FORTRAN 77 and Pascal.

See the `cc(1)` manual page for further details.

Symbolic Definitions and Conditional Compilation

`cpp` is a the C preprocessor. `cpp` has facilities for:

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

See the `cpp(1)` manual page for further details.

Checking Validity of C Programs

`lint` 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. `lint` can do full cross-module checking of separately compiled programs, and can check the for correct use of library functions.

See *Programming Utilities for the Sun Workstation* and the `lint(1)` manual page for further details.

Formatting C Programs

`indent` is a formatter for arranging C source code into standard styles:

- Several differnt 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`.
- You can set up a *profile* of your own options for `indent`.

10.2. FORTRAN

FORTRAN 77 is the latest standard from the ANSI FORTRAN standards committee.

Compiling FORTRAN Programs

`f77` is a full compiler for ANSI Standard FORTRAN 77. Major features are:

- compatible with C and supporting tools at object level
- optional source compatibility with FORTRAN 66
- free format source
- optional subscript-range checking, detection of uninitialized variables
- all widths of arithmetic: 2-byte and 4-byte integer
- 4-byte and 8-byte real; 8-byte and 16-byte complex.

See *FORTRAN Programmer's Guide for the Sun Workstation* and the `f77`(1) manual page for further details.

FORTRAN Preprocessor

`ratfor` stands for 'Rational FORTRAN' and was developed before the advent of FORTRAN 77 to add rational control structure like C's to FORTRAN. Highlights of the `ratfor` language are:

- compound statements — statements can be grouped into blocks.
- structured programming constructs — if-else, do, for, while, repeat-until, break, next.
- symbolic constants via a macro facility.
- file inclusion.
- free format source independent of FORTRAN's column-oriented restrictions.
- translates relational operators like > and >= into the more obscure FORTRAN forms of `.GT.` and `.GE.`

`ratfor` produces genuine FORTRAN to carry away and may be used with FORTRAN 77.

See *FORTRAN Programmer's Guide for the Sun Workstation* and the `ratfor`(1) manual page for further details.

10.3. Pascal Programming Language

The Pascal system for the Sun Workstation is derived from the Pascal compiler and interpreter implemented by William N. Joy and Charles Haley at the University of California at Berkeley. The Pascal system is an ANSI Pascal compiler and interpreter system.

Other tools in the Pascal system include:

`px` a Pascal execution profiler

`pxref`
: a program for making cross-referenced listings of Pascal programs.

See *Pascal Programmer's Guide for the Sun Workstation* and the `pc`(1), `px`(1), and `pxref`(1), manual pages for further details.

10.4. Assembler

`as` is the machine-level assembler for the Sun hardware family.

- `as` creates object program consisting normally 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
- 'conditional jump' instructions become branches or branches plus jumps depending on distance.

See *Assembly Language Reference Manual for the Sun Workstation* and the `as(1)` manual page for further details.

10.5. Linker

`ld` is the link editor

- Combine relocatable object files.
- Insert required routines from specified libraries
- Resulting code is sharable by default.

See the `ld(1)` manual page for further details.

11

Software Development Tools

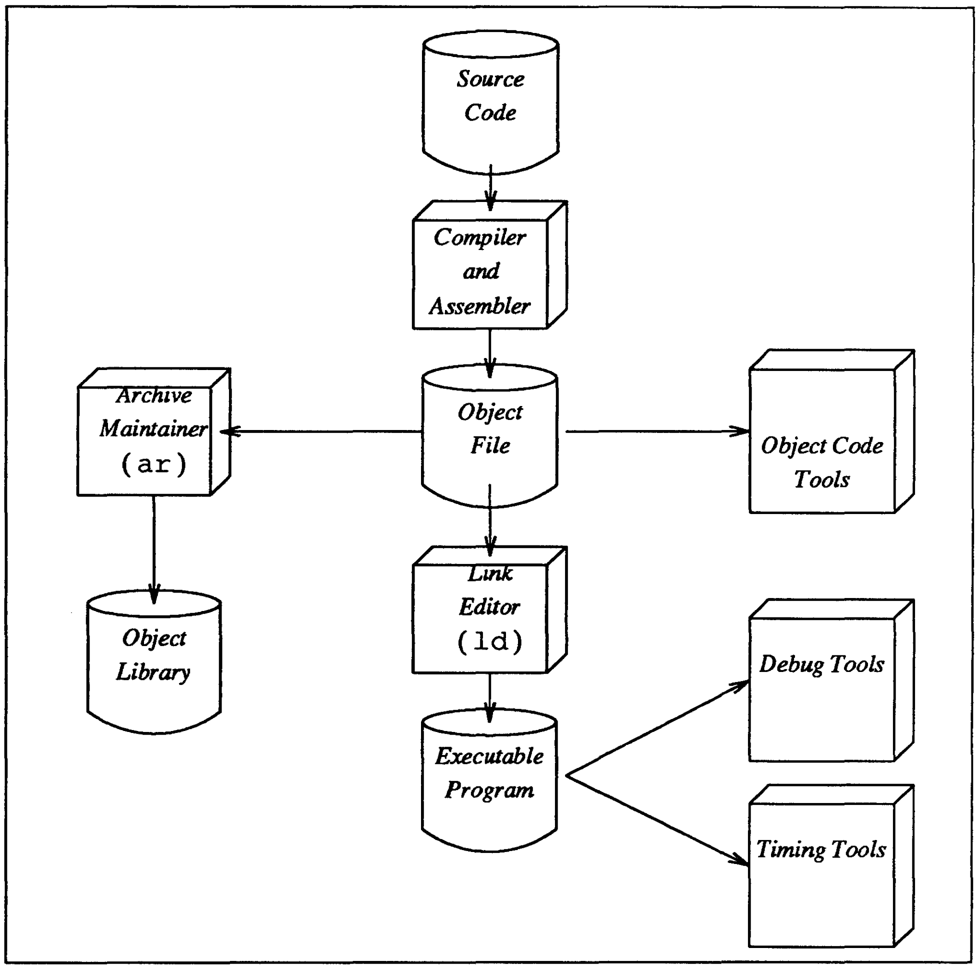
11 Software Development Tools

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.

11.1. Programming Tools to Work With Object Code

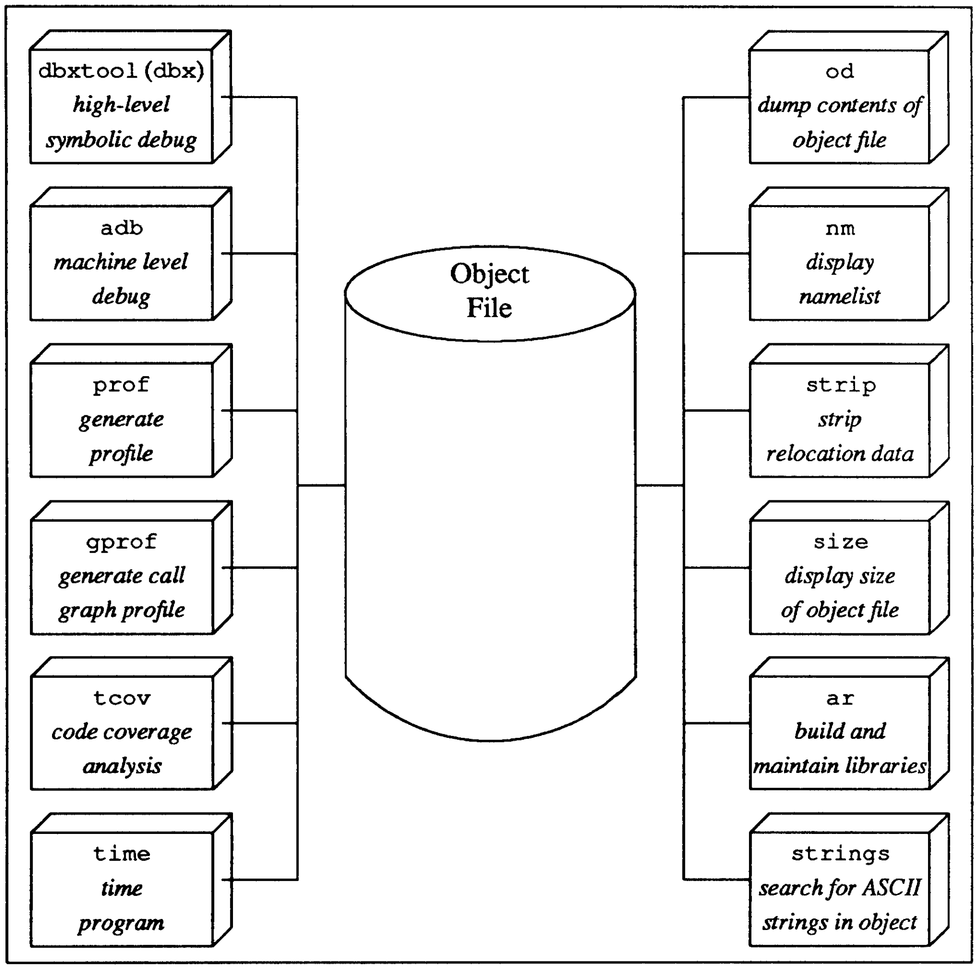
The diagram below shows the flow from source code to object code, plus the major groups of tools that can be used with the object code. A later diagram shows more detail.

Figure 11-1 *Major Object Code Programming Tools*



Now on a more detailed level, here are the more commonly used tools to work with object code.

Figure 11-2 *Commonly Used Tools to Work with Object Code*



Debugging Programs at the High Level

`dbxtool` and `dbx` are part and parcel of the same debugging capability. `dbxtool` is a window-based source level debugger. `dbxtool` is based on `dbx`, 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` are:

- Multiple source language debugging.
- Can display a stack backtrace 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.
- Can be used to debug the kernel.

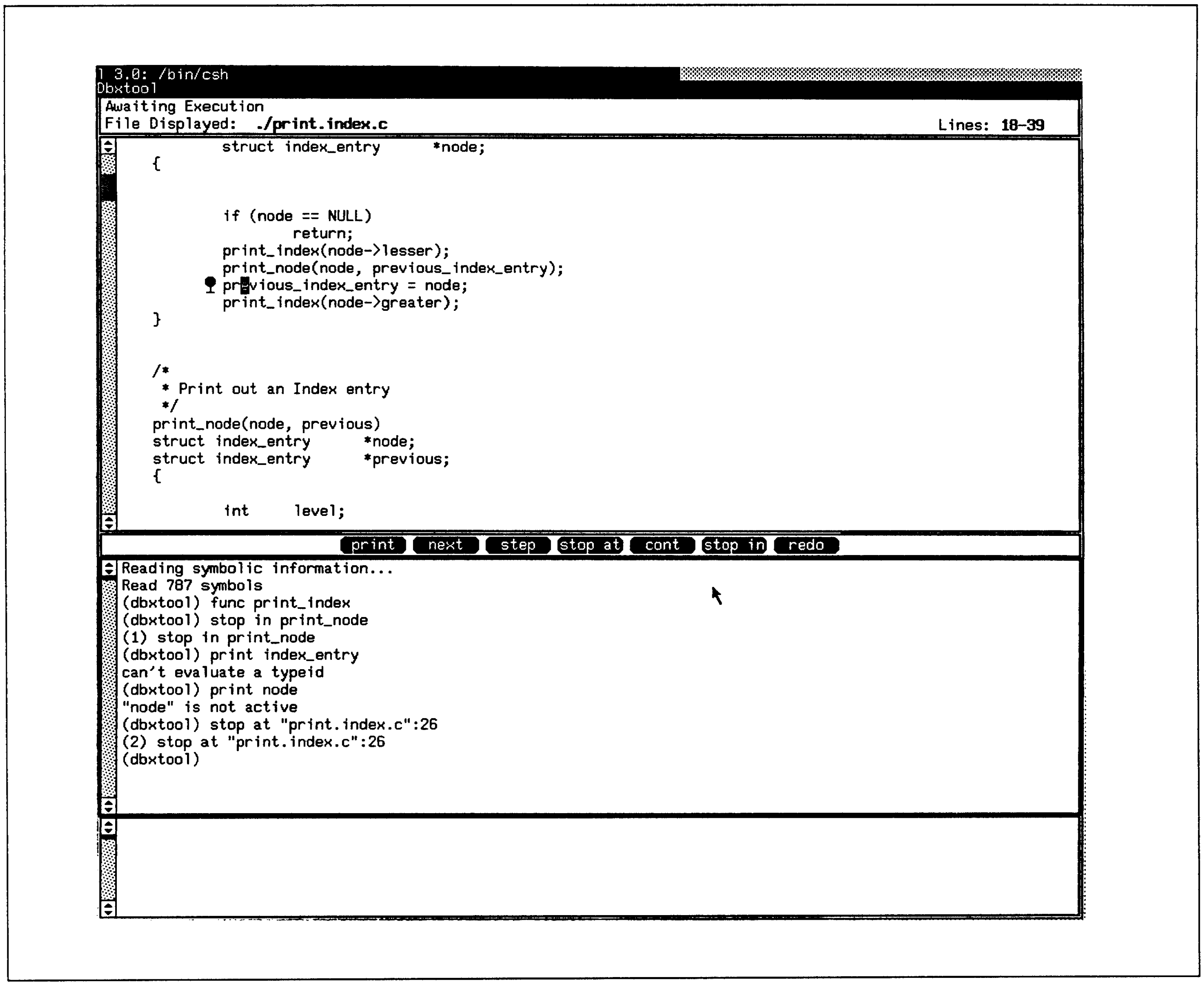
`dbxtool` is a window and mouse-based interface to `dbx`. All the features of `dbx` are available in `dbxtool` except that you can perform the most common operations (print, next, step, stop at, stop in, cont, and redo) by 'pushing a button' in the control panel.

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.
- *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 shows a `dbxtool` window with a program being operated on.

Figure 11-3 *Example* dbxtool *Window*


```
1 3.0: /bin/csh
Dbxtool
Awaiting Execution
File Displayed:  ./print.index.c                                          Lines: 18-39
          struct index_entry      *node;
  {


          if (node == NULL)
                  return;
          print_index(node->lesser);
          print_node(node, previous_index_entry);
          previous_index_entry = node;
          print_index(node->greater);
  }


  /*
   * Print out an Index entry
   */
  print_node(node, previous)
  struct index_entry      *node;
  struct index_entry      *previous;
  {

          int     level;
```

```
Reading symbolic information...
Read 787 symbols
(dbxtool) func print_index
(dbxtool) stop in print_node
(1) stop in print_node
(dbxtool) print index_entry
can't evaluate a typeid
(dbxtool) print node
"node" is not active
(dbxtool) stop at "print.index.c":26
(2) stop at "print.index.c":26
(dbxtool)
```

See *Debugging Tools for the Sun Workstation* the dbxtool(1) and dbx(1) manual pages for further details.

Debugging at the Machine Level

adb 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 local and 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
- Disassembled machine instructions.

See *Debugging Tools for the Sun Workstation* and the `adb(1)` manual page for further details.

Building and Maintaining Libraries

`ar` is the library maintenance utility. `ar`'s principal use is to build and maintain object code libraries for use by `ld` — the link editor. `ar` can be used as a general utility for collecting groups of files into a single unit. `ar`'s major features 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)
- retrieve files from archive.

See the `ar(1)` manual page for further details.

Dumping File Contents

`od` dumps the contents of any file. Output options include

- any combination of octal or decimal by words, octal by bytes, ASCII, opcodes, hexadecimal
- Range of dumping is controllable.

See the `od(1)` manual page for further details.

Displaying the Namelist

`nm` displays the namelist (symbol table) of an object program. Provide control over the style and order of names that are printed.

See the `nm(1)` manual page for further details.

Displaying Size of a Program

`size` displays the memory requirements of one or more object files.

See the `size(1)` manual page for further details.

Stripping Relocation and Symbol Table

`strip` removes the relocation and symbol table information from an object file to save space.

See the `strip(1)` manual page for further details.

Serach for ASCII Strings in Binary File

`strings` is a useful tool to locate ASCII strings in a binary file.

See the `strings(1)` manual page for further details.

11.2. Performance Analysis Tools

Sun Microsystems supplies several facilities for monitoring performance of software, ranging from a simple command to report the time a program takes to execute, to a code-coverage tool to provide detailed statement-by-statement analysis of a program.

Timing a Program

`time` is a simple system command that just produces a report on how much time a given program takes to execute.

See *Programming Utilities for the Sun Workstation* and the `time(1)` and `/bin/time(1)` manual pages for further details.

Profiling a Program

`prof` 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.

See *Programming Utilities for the Sun Workstation* and the `prof(1)` manual page for further details.

Generating a Call Graph Profile

`gprof` is a step up from `prof`. `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, the callees of a specific routine, and the number of times a routine was called or called another routine.

See *Programming Utilities for the Sun Workstation* and the `gprof(1)` manual page for further details.

Analysing Code Coverage

`tcov` 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.
- 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.

See *Programming Utilities for the Sun Workstation* and the `tcov(1)` manual page for further details.

11.3. Program Generation Tools

Frederick Brooks pointed out[2] 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 produuct* (a whole system that must work together and be documented). The UNIX 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` (maintaining history).

[2] ***The Mythical Man Month***

Building and Maintaining Programs

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

- You specify *via* a control file (called a *makefile*) *what* you want to build — called a *target*, *what* things the target depend on — called *dependencies*, and *how* to go about constructing the target from its dependents — *rules*.
- make has innate rules that specify the dependencies between object files and the C compiler, FORTRAN compiler, yacc, lex and so on.

See *Programming Utilities for the Sun Workstation* and the make(1) manual page for further details.

Maintaining History

sccs is the Source Code Control System. sccs maintains and controls multiple versions of text files. sccs maintains the multiple versions in an SCCS database.

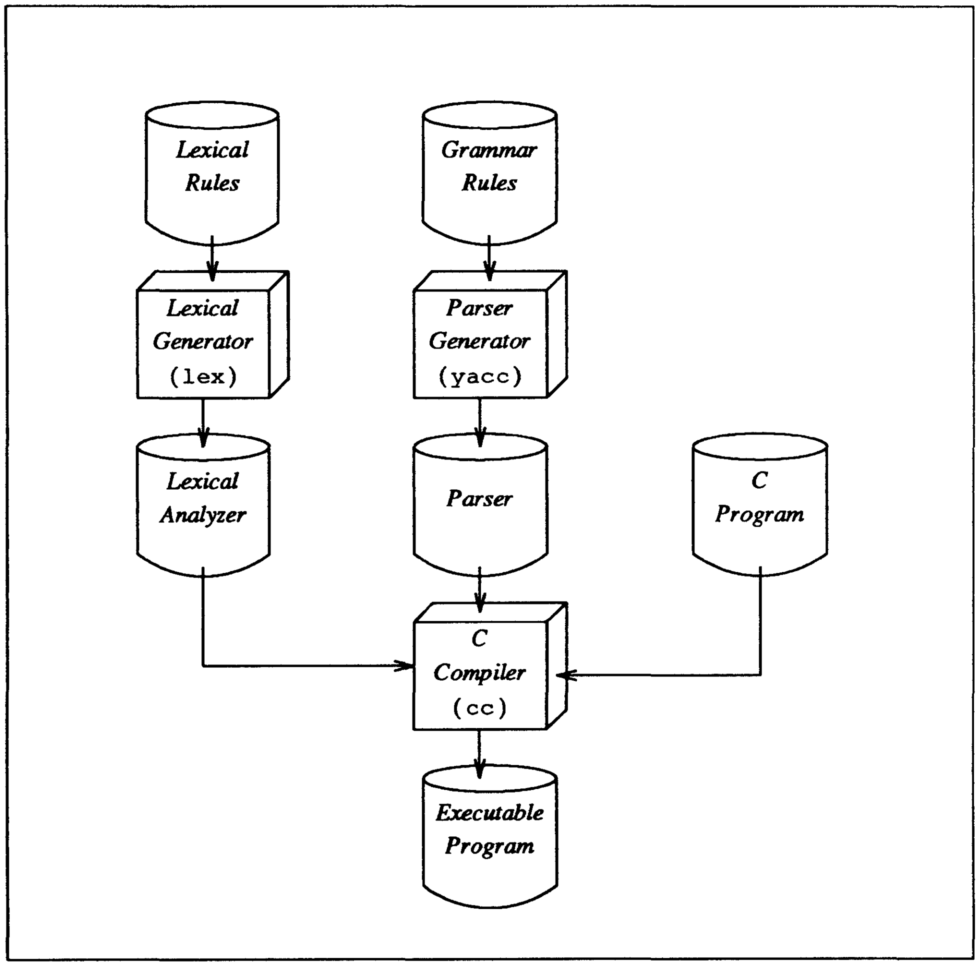
- You initially create an SCCS database for a file to establish the initial version.
- You get a read-only copy of a file from the database for compiling or any other activity that doesn't involve changing the file.
- You edit a writeable copy of a file from the database for making changes.
- You delta 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.

See *Programming Utilities for the Sun Workstation* and the sccs(1) manual page for further details.

11.4. Compiler Development Tools

lex and yacc started life as utilities 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 described pictures for document production (pic). lex and yacc are constructed to work together, as shown in Figure 11-4 below.

Figure 11-4 `lex` *and* `yacc` *In Program Development*



Generating Lexical Analyzers

`lex` generates lexical analyzers. Highlights are:

- 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
- Resulting lexical analyzers interface cleanly with *yacc* parsers.

See *Programming Utilities for the Sun Workstation* and the `lex(1)` manual page for further details.

Generating Syntactic Parsers

`yacc` is an LR(1)-based compiler writing system

- During execution of resulting parsers, arbitrary C functions may be called to do code generation or semantic actions
- BNF syntax specifications
- Precedence relations
- Accepts formally ambiguous grammars with non-BNF resolution rules.

See *Programming Utilities for the Sun Workstation* and the `yacc(1)` and `eyacc(1)` manual pages for further details.

11.5. Other Programming Tools

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

Macro Processing

m4 is a general purpose stream-oriented macroprocessor that recognizes macros anywhere in text

- Syntax fits with functional syntax of most higher-level languages
- m4 can evaluate integer arithmetic expressions.

See *Programming Utilities for the Sun Workstation* and the m4(1) manual page for further details.

Calculators — dc

dc is an interactive programmable desk calculator

- Has named storage locations as well as conventional stack for holding integers or programs
- Unlimited precision decimal arithmetic
- Appropriate treatment of decimal fractions
- Arbitrary input and output radices, in particular binary, octal, decimal and hexadecimal
- Reverse Polish operators:

  + − * /
  remainder, power, square root, load, store, duplicate, clear, print, enter program text, execute.

See *Games, Demos, and Other Pursuits — Beginner's Guide* and the dc(1) manual page for further details.

Calculators — bc

bc is a C-like interactive interface to the dc desk calculator described above

- bc has all the capabilities of dc with a high-level syntax
- Arrays and recursive functions
- Immediate evaluation of expressions and evaluation of functions upon call
- Arbitrary precision elementary functions exp, sin, cos, atan
- Go-to-less programming.

See *Games, Demos, and Other Pursuits — Beginner's Guide* and the bc(1) manual page for further details.

11.6. Summary of Language Utilities

Here is an alphabetical list of the utilities described in this chapter.

Figure 11-5 *Summary of Language Processing Programs*

Program Name	*Function*
adb	Debug at the machine Level
ar	Build and maintain libraries
dbx	Debug programs at the high Level
dbxtool	Debug programs at the high 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`	Analyse code coverage
`time`	Time a program
`yacc`	Generate syntactic analyzers

12

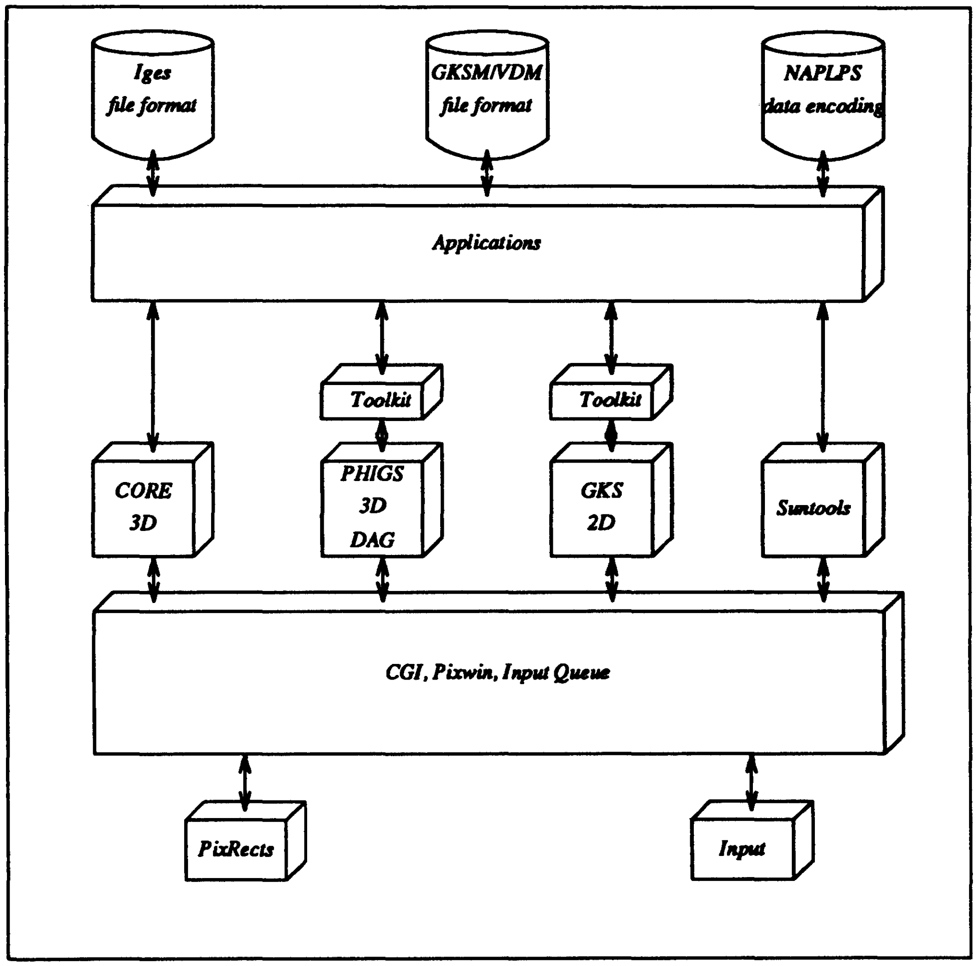
Graphics Tools

12

Graphics Tools

Sun Microsystems support for graphics leans heavily on standards. Figure 12-1 below illustrates the relationships between the various graphics packages.

Figure 12-1 *Graphics Standards*



Note that not all of the graphics packages shown in the diagram are actually supplied as Sun products — to date, Sun Microsystems supply these graphics packages:

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.

See *SunCore Reference Manual for the Sun Workstation* for further details.

SunGKS is an implementation of the ANSI *Graphical Kernel Standard* (GKS). The *Graphical Kernel System* (GKS) is a graphics standard designed for 2D interactive computer graphics on workstations. **SunGKS** conforms to level 2C (Workstation Independent Segment Storage and full input) of the GKS standard.

See *SunGKS Reference Manual for the Sun Workstation* for further details.

SunCGI is an implementation 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 which 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.

See *SunCGI Reference Manual for the Sun Workstation* for further details.

Pixrect is an implementation of the is a low-level RasterOp graphics library for writing device-independent applications for Sun products. The **Pixrect** library is the lowest level interface to the display device, sitting just above the device drivers. **Pixrect** does *not* provide support for input devices or overlapping windows.

See *Pixrects Reference Manual for the Sun Workstation* for further details.

13

Sun Operating System — Internal Features

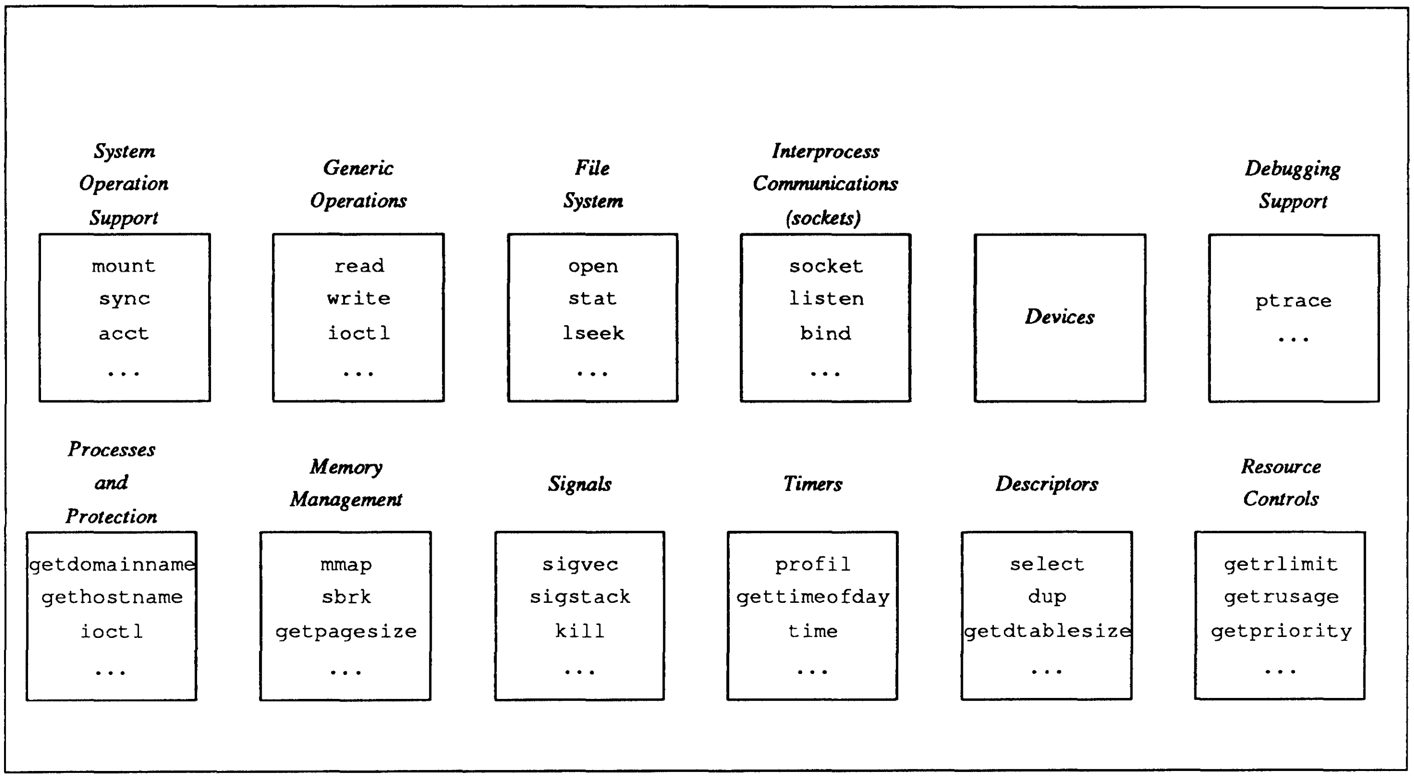
13

Sun Operating System — Internal Features

Sun Microsystems provides its own enhanced version of the UNIX operating system as the basic operating software to exploit the resources of Sun hardware products. This chapter descibes the major features of the operating system.

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 more usually, applications make use of library functions that in turn use the kernel's primitive functions.

Figure 13-1 *Kernel Primitive Functions*

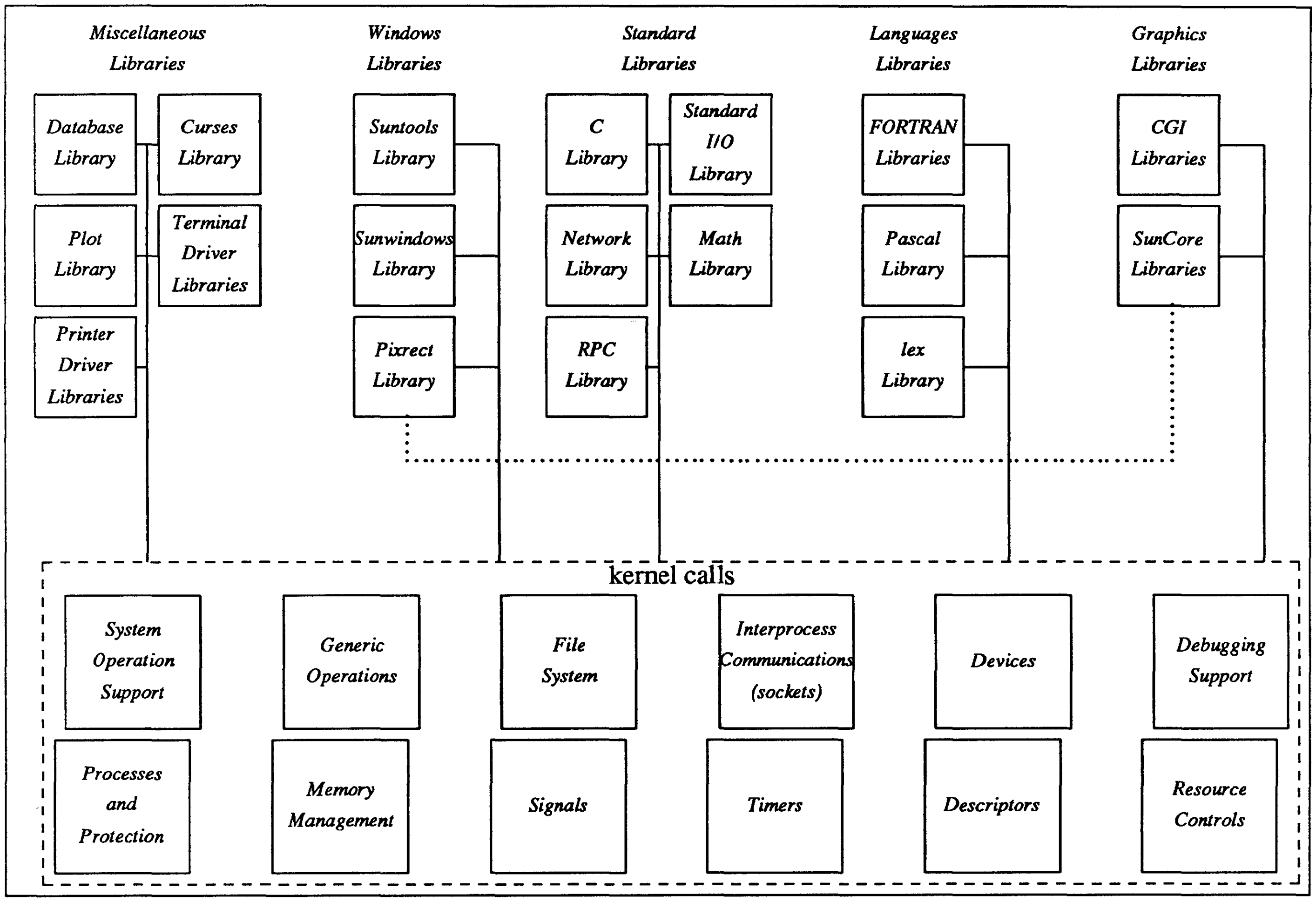


The kernel primitive functions can be associated into the functional groups shown in figure 13-1 above. The *UNIX Interface Overview* and section 2 of the

UNIX Interface Reference Manual contain details on the kernel functions.

As mentioned above, most applications will use one or more standard *library packages* that in turn call the kernel. The main groups of library packages are illustrated in figure 13-2 below.

Figure 13-2 *Standard Library Packages*



There is in fact a minor piece of misdirection in the picture above — the Standard I/O library is in fact a part of the C library, but is shown separate here for illustrative reasons. To find out more about the various libraries you should consult these manuals:

Library	***Described in Manual(s)***
C Library	UNIX Interface Reference Manual (section 3)
Standard I/O Library	Programming Tools for the Sun Workstation UNIX Interface Reference Manual (section 3S)
Network Library	Networking on the Sun Workstation UNIX Interface Reference Manual (section 3N)

RPC Library	Networking on the Sun Workstation UNIX Interface Reference Manual (section 3R)
Math Library	UNIX Interface Reference Manual (section 3M)
FORTRAN Libraries	FORTRAN Programmer's Guide for the Sun Workstation
Pascal Library	Pascal Programmer's Guide for the Sun Workstation
`lex` Library	Programming Tools for the Sun Workstation
CGI Libraries	CGI Reference Manual for the Sun Workstation
SunCore Libraries	SunCore Reference Manual for the Sun Workstation
Suntools Library	SunView System Programmer's Guide
Sunwindows Library	SunView System Programmer's Guide
Pixrect Library	Pixrect Reference Manual for the Sun Workstation
Database Library	UNIX Interface Reference Manual (section 3X)
Curses Library	UNIX Interface Reference Manual (section 3X)
Terminal Driver Libraries	UNIX Interface Reference Manual (section 3X)
Plot Library	UNIX Interface Reference Manual (section 3X)

Other major features of the operating system are described below.

Device-independent I/O and redirection

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

Virtual memory

Supports processes up to 256 megabytes (given adequate disk space for paging) for greatly enhanced amount of available main memory and reduced delays when running programs as only the parts of the program needed to be loaded in core are in fact loaded.

Job Control Facilities

The 4.2BSD operating system and Sun Microsystems operating software provides facilities for *job control* that were missing in previous implementations of the UNIX operating system. 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 to the user via the C-Shell command interpeter described in chapter 5.

13.1. Networking Facilities

The Sun 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.

Figure 13-3 *Network Architecture*

7 — Application	`mail`	`rcp`	`rlogin`	`rsh`
	`ftp`	`NFS`	`YP`	`telnet`
6 — Presentation	XDR			
5 — Session	RPC			
4 — Transport	TCP		UDP	
3 — Network	IP (Internetwork)			
2 — Data Link	Ethernet	Point to Point		
1 — Physical	Ethernet	Point to Point		

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). Both RPC and XDR are portable, providing a kind of standard I/O library for interprocess communication. Thus programmers now have a standardized access to sockets without having to be concerned about the low-level details of the `accept()`, `bind()`, and `select()` procedures.

Interprocess Communication

The network (interprocess) communications facilities dervied from 4.2BSD are based on the *socket*.

Network communication using standard protocols.
- Inter-process 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.
- Communications *via* the socket mechanisms provide remote logins, copies, and Shells over the local network.

Remoted Procedure Call (RPC) services provide an easier to use layer of abstraction than does the socket mechanisms. However, you can find the appropriate information in *Networking on the Sun Workstation* in the sections on Interprocess Communication. Sections 2 and 3 of the *UNIX Interface Reference Manual* contain details of the function calls.

Reference Documentation

Programmers wishing to use the networking facilities can refer to one or more of these documents that are a part of the general manual called *Networking on the Sun Workstation* :

Network Services Guide
is for users who have a general interest in network services. It explains the network file system and the yellow pages facilities in some detail. Although it is not a manual for system administrators, the material is heavily slanted in that direction.

Remote Procedure Call Programming Guide
is for programmers who wish to write network applications using remote procedure calls, thus avoiding low-level system primitives based on sockets. Readers must be familiar with the C programming language, and should have a working knowledge of network theory.

External Data Representation Protocol Specification
is for programmers writing complicated applications using remote procedure calls, who need to pass complicated data across the network. It is also a reference guide for system programmers implementing Sun's Network File System on new machines.

Remote Procedure Call Protocol Specification
is a reference guide for system programmers implementing Sun's Network File System on new machines. It is of little interest to programmers writing network applications.

Network File System Protocol Specification
is a reference guide for system programmers implementing Sun's Network File System on new machines. It is of little interest to programmers writing network applications.

Yellow Pages Protocol Specification
is a reference guide for system programmers implementing a Yellow Pages database facility on new machines. It is of little interest to programmers writing network applications.

Inter-Process Communications Primer
taken from Berkeley's 4.2 release, is for system programmers who need to use low-level networking primitives based on sockets. Since remote procedure calls are easier to use than sockets, this primer is of little interest to most network programmers.

Network Implementation
describes the low-level networking primitives in the 4.2 UNIX kernel. It is of interest primarily to system programmers and aspiring UNIX gurus.

14

SunGuide — Window Programming Interface

14

SunGuide — Window Programming Interface

SunView is the external user's view of the user interface, SunView is built on top of *SunGuide* — Sun General User Interface Design Environment — is a general toolkit for building window-based applications

The window system *programming interface* is a collection of subroutine libraries. There are three major levels of abstraction in the programming interface to the window system, and a programmer can use any or all of these three layers to write applications.

The user interface (external features) of the window system is described in chapter 4 — *SunView — User Environment.*

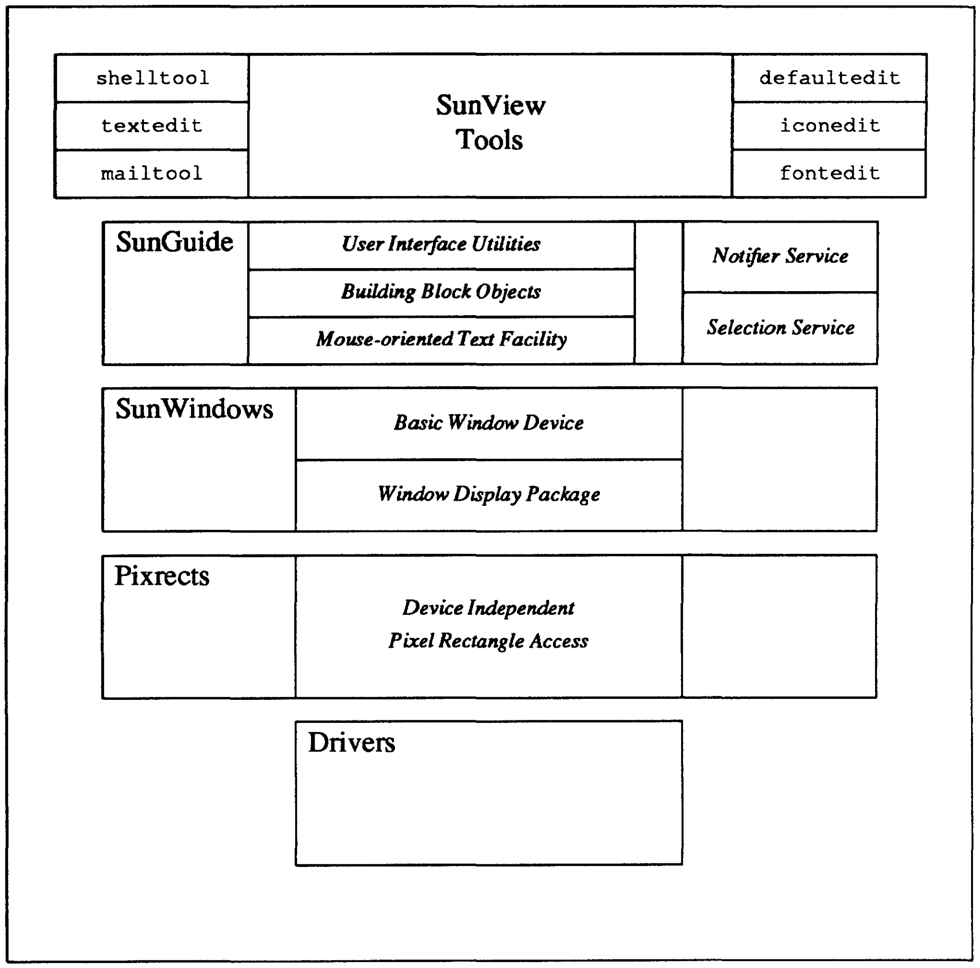
The three major layers of SunView are:

- SunGuide
- SunWindows
- Pixrects

Pixrects are at the lowest layer of the system, close to the hardware drivers.

Here is a general diagram of the layering.

Figure 14-1 *Layering of SunView*



SunGuide Facilities

SunGuide contains the abstractions — the toolkit — to build applications software:

User Interface Utilities include

- the *notifier* — the agent that distributes events among multiple applications.
- the *selection service* — the agent that manages text selections.

Building Blocks include

- *text subwindows* for displaying textual data.
- *canvas subwindows* for displaying graphical information.
- *scrollbars* for scrolling backwards and forwards in a file.
- *Control Panels* containing 'pushbuttons' for selecting actions.

- *Menus* to bring up choices for the user to select among.

There is a general-purpose mouse-oriented *text facility* containing functions to select, extend, cut, and paste.

SunWindows Facilities

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

Pixrects

Pixrects are a set of low-level device-independent routines that manage *pixel rectangles*. The Pixrect layer interfaces to the hardware device drivers.

Further Reading

For further details on programming for the Sun Window System you should read:

- *SunView Application Programmer's Guide*.
- *SunView System Programmer's Guide*.
- *Pixrects Reference Manual for the Sun Workstation* for details of the low-level routines that interface directly to the frame buffer.

15

Document Production

15

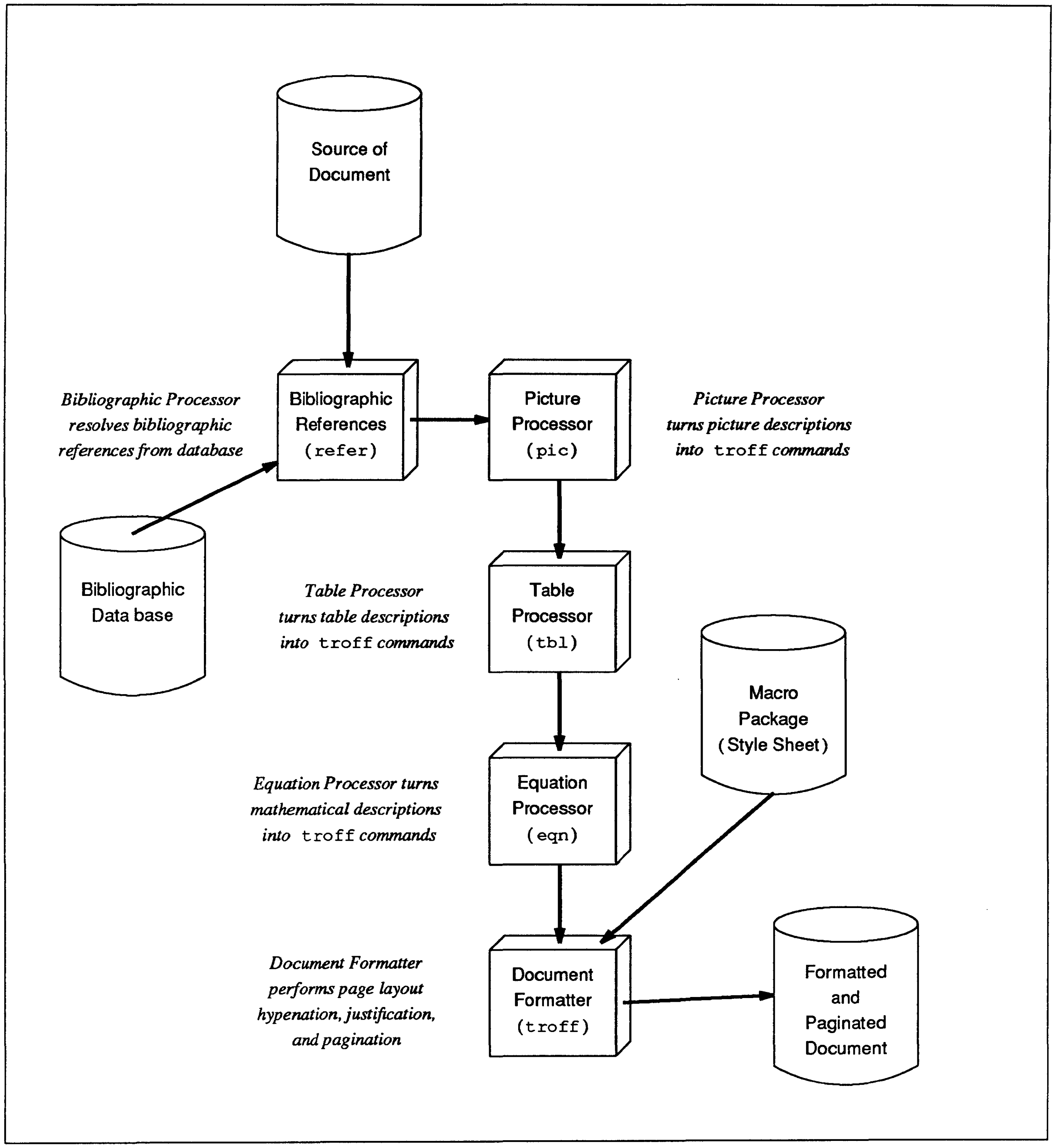
Document Production

Bell Laboratories is one of the world's largest research institutions and as such, generates vast amounts of technical papers. Early in its life, the UNIX system was enhanced with software to assist the job of producing technical documents. The major tool that evolved for the job was called `troff` — a *text formatter* oriented towards driving a second-generation phototypesetter called a C/A/T.

Today, `troff` is the main utility for formatting documents. Although `troff` is capable of very complex feats of typographic formatting, the nature of the devices for which `troff` was originally intended resulted in a program that must be instructed in excruciating detail as to how a document should be laid out. Not only must users be familiar with the details of each `troff` request, but they must also be knowledgeable of typography terms and concepts. Over the years of its life, `troff` has been surrounded by sundry utilities and tools to ease users' jobs. *Macro packages* are perhaps the most important tool to help users in generating documents. A macro package may be thought of as a *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* take over special effects such as mathematical equations, tables, and line drawings. The picture below conveys some of the flavor of the extensive battery of tools available for producing documents on the UNIX system.

Figure 15-1 *Document Formatting Model with Macro Package*


Source of Document
Bibliographic Processor resolves bibliographic references from database
Bibliographic References (refer)
Picture Processor (pic)
Picture Processor turns picture descriptions into troff commands
Bibliographic Data base
Table Processor turns table descriptions into troff commands
Table Processor (tbl)
Macro Package (Style Sheet)
Equation Processor turns mathematical descriptions into troff commands
Equation Processor (eqn)
Document Formatter performs page layout hypenation, justification, and pagination
Document Formatter (troff)
Formatted and Paginated Document

15.1. Formatting Documents

`troff` and `nroff` are the major document formatting programs available on the UNIX system. Although there are now 'What You See Is What You Get' (WYSIWYG) document preparation packages available for the Sun Workstation, `troff` and `nroff` still have their place in a variety of very difficult formatting applications where WYSIWYG systems don't as yet fit the bill.

`troff` was originally written (very specifically) for a specific second-generation phototypesetter. `troff` can be used with appropriate conversion utilities to drive other types of devices. `nroff` drives ASCII terminals of all types. `troff` and `nroff` accept the same input language and are capable of elaborate formatting feats when appropriately programmed

- completely definable page format keyed to dynamically planted 'interrupts' at specified lines
- maintains 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 macros invocable in mid-line
- 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.
- `troff` provides terminal output for rough sampling of final output
- `nroff` produces multicolumn output on the workstation (or terminal capable of reverse line feed), or through the `col` postprocessor.

See *Formatting Documents on the Sun Workstation* and *Using nroff and troff on the Sun Workstation* and the `troff(1)` and `nroff(1)` manual pages for further details.

15.2. Macro Packages

Using raw `troff` codes by themselves to describe a document has been likened to writing microcode for a typesetter. Such a process is extremely time-consuming and difficult to make any substantive changes in style or layout afterwards. For this reason, `troff` incorporates a macro facility where frequently-used sequences of `troff` requests are collected together into named chunks, and then you call the chunks 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 canned requests for use with `nroff` and `troff`. Some of the features that `-ms` provides 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 callouts.
- 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.
- Draft dates.

See *Formatting Documents on the Sun Workstation* for further details.

`-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 than the `-ms` macro package.

See *Formatting Documents on the Sun Workstation* for further details.

`-me` Macro Package

The `-me` macro package — another package of canned formatting requests — is also available. The `-me` macro package was developed at the University of California at Berkeley specifically for the computer science environment there.

See *Formatting Documents on the Sun Workstation* for further details.

15.3. `troff` Preprocessors

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

Mathematical Typography

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

Example of `eqn`

Formulas are written in a style as if you were 'talking' them over the telephone to someone else:

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

to produce:

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

neqn is a version of eqn for nroff. neqn 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.

Main features of the eqn language are:

- Automatic calculation of size changes for subscripts, sub-subscripts, and such for equation 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' of formulae for matrices, conditional alternatives, etc.
- Integrals, sums, etc., with arbitrarily complex limits
- Diacriticals: dots, double dots, hats, bars, etc.
- Easily learned by nonprogrammers and mathematical typists.

See *Formatting Documents on the Sun Workstation* and the eqn(1) manual page for further details.

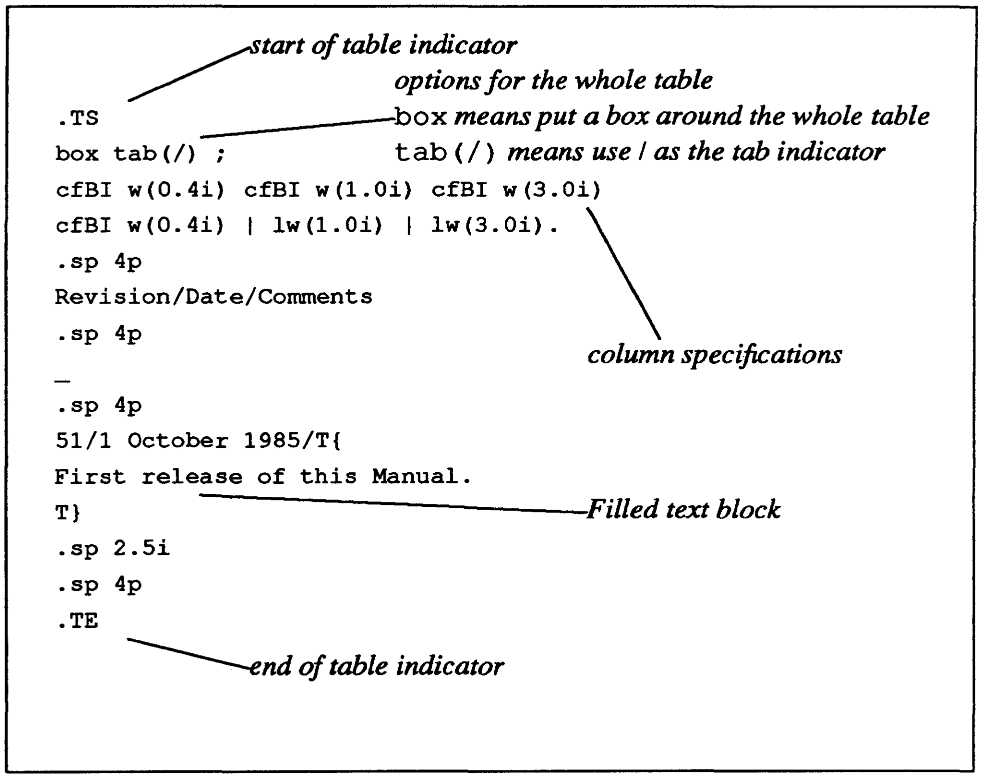
Laying Out Tables

tbl is a preprocessor for nroff and troff that translates simple descriptions of table layouts and contents into detailed typesetting instructions

- 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
- 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
```

start of table indicator
options for the whole table
box means put a box around the whole table
tab(/) means use / as the tab indicator
column specifications
Filled text block
end of table indicator

Here are the results of formatting the above table source:

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

See *Formatting Documents on the Sun Workstation* and the `tbl(1)` manual page for further details.

Bibliographic References

`refer` 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.

There are some supporting utilities that make `refer` easier to use. `addbib` creates and extends the bibliographic database, `sortbib` sorts the bibliographic database by author, date, or lother criteria, and `roffbib` can format the entire bibliographic database as a bibliography or annotated bibliography. See *Formatting Documents on the Sun Workstation* and the `addbib(1)`, `sortbib(1)`, `roffbib(1)`, `indxbib(1)`, and `lookbib(1)` manual pages for further details.

15.4. Other Document Preparation Tools

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

`spell` is described in chapter 9 and can check the spelling of a document against an on-line dictionary.

See *Formatting Documents on the Sun Workstation* and the `spell(1)` manual page for further details.

Handling Reverse Paper Motions

`col` converts files with reverse line feeds into canonical form for one-pass printing. `col` is used mainly in conjunction with `nroff` to deal with those printers that can't do reverse paper motions.

See *Formatting Documents on the Sun Workstation* and the `col(1)` manual page for further details.

Stripping `troff` *Constructs*

`deroff` removes `troff` requests from a source file. The spelling checker facility, among others, uses `deroff` to get rid of all the `troff` requests which would otherwise show up as mis-spelled words.

See *Formatting Documents on the Sun Workstation* and the `deroff(1)` manual page for further details.

Checking Syntax

`checknr` 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 chunks of a document disappearing into a black hole. Such problems are often caused by, say, starting a display and forgetting to end it.

`checkeq` checks a document for correctly specified equations.

See *Formatting Documents on the Sun Workstation* and the `checknr(1)` and `checkeq(1)` manual pages for further details.

Generating a Permuted Index

`ptx` generates a permuted, or keyword-in-context, index from text files.

See *Formatting Documents on the Sun Workstation* and the `ptx(1)` manual page for further details.

Interpreting `troff` *Output*

`pti` is the phototypesetter interpreter designed for those sites that need to examine the codes that `troff` generates destined for the C/A/T phototypesetter. The C/A/T phototypesetter requires a binary code whose format is complicated beyond all belief. `pti` displays a readable interpretation of these codes for those who are (say) writing software to convert C/A/T codes to other formats.

See *Formatting Documents on the Sun Workstation* and the `pti(1)` manual page for further details.

15.5. Summary of Document Preparation Utilities

Here is an alphabetical list of the document preparation utilities described in this chapter.

Figure 15-2 *Summary of Document Preparation Programs*

Program Name	***Function***
`checkeq`	Check correctly specified equations
`checknr`	Check `nroff` and `troff` constructs
`col`	Filter out reverse paper motions
`deroff`	Strip out `nroff` and `troff` constructs
`eqn`	Language for specifying mathematical equations
`-man`	Macro package to format the on-line manual pages
`-me`	Macro package derived from U. C. Berkeley
`-ms`	Popular macro package for technical memorandum
`nroff`	Document formatter oriented to typewriter-like devices
`pti`	Interpreter for `troff` output files
`ptx`	Generate permuted (keyword in context) index
`refer`	Bibliographic database processor
`spell`	Check spelling
`tbl`	Language for describing columnar layouts
`troff`	Document formatter oriented to typesetter

16

System Administration

16

System Administration

System Administration is a whole collection of tasks that must be done to install new versions of the operating system and application software, keep the hardware and software running, troubleshoot, and upgrade.

In general, you must be the *super-user* to manage the system. The super-user (also called *root*) has powers to override other users' permissions and in general can wreak much havoc if out of control.

System administration can be divided into these rough areas:

- Initially setting up the hardware and installing the software.
- Managing the system on a daily basis — adding new user accounts.
- Fixing things when they're broken. Rebooting the system.
- Adding printers and terminals and making the system aware of their existence.

Manuals You Should Read

When you receive your system you should read the *Installation Manual* to find out how to unpack and set up the hardware.

Then the manual entitled *Installing UNIX on the Sun Workstation* is a guide to configuring the system using the `setup` utility mentioned below, and to installing the UNIX operating system.

Finally, *System Administration for the Sun Workstation* contains a collection of 'folklore' that you need to manage the system.

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

16.1. Setup and Installation

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

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 devices drivers and their data structures occupies real memory. When you install your system, you can reconfigure the kernel to only include those drivers for your particular system. `config` is the utility that creates the necessary files and commands to configure the kernel. the procedure for reconfiguring the kernel is well described in *Installing UNIX on the Sun Workstation*.

16.2. Startup and Shutdown Procedures

This system attempts to automate the boot procedures as much as possible. There are automatic boot procedures to bring up Sun UNIX. Automatic reboot and file consistency checks and repair in the event of system crash.

The system normally boots itself automatically when power is turned on. However, the super-user can shut down the system and reboot it at will.

Halting and Rebooting

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

`reboot` performs an automatic reboot.

`sync` forces all outstanding I/O on the system to complete — used to shut down gracefully.

Checking File Systems

`fsck` is an interactive file system check and repair program. `fsck` is usually called into play automatically during the boot procedure.

- Display gross statistics: number of files, number of directories, number of special files, space used, and free space
- Report duplicate use of space
- Retrieve lost space
- Report inaccessible files
- Check consistency of directories
- List names of all files.

`fsck` supersedes `dcheck`, `icheck`, and `ncheck`.

16.3. Day to Day Adminstration Tasks

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

Becoming Super User

`su` changes your user ID so that you become the super-user temporarily with all the rights and privileges thereof.

Changing Ownership of a File

`chown` changes the ownership of one or more files. At times you may need to 'give a file away' to someone else other than the original owner. This is usually true when you create a new account on the system.

Scheduling Events

`cron` schedules regular actions at specified times

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

Mounting and Unmounting File Systems

`mount` attaches a device containing a file system to the tree of directories. `mount` protects against nonsense arrangements.

`umount` removes the file system contained on a device from the tree of directories. `umount` protects against removing a busy device.

Creating New File Systems

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

`mkfs` makes a new file system on a device.

Creating Special Files

`mknod` 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.

Dumping and Restoring Files

`tar` manages file archives on magnetic tape

- Collect files into an archive
- Update tape archive by date
- Replace or delete tape files
- Display table of contents
- Retrieve from archive.

`dump` dumps the file system stored on a specified device, selectively by date, or indiscriminately.

`restore` restores a dumped file system, or selectively retrieves parts thereof.

16.4. System Accounting Facilities

The UNIX system was originally a times-sharing system intended to support multiple users on a single CPU. To this end there are facilities to perform accounting functions. `ac` publishes cumulative connect time report. Connect time by user or by day and for all users or for selected users.

`sa` publishes Shell accounting report:

- give usage information on each command executed, number of times used, total system time, user time and elapsed time, optional averages and percentages, and sorting on various fields
- note that the timing information on which the reports are based can be manually cleared or shut off completely.

17

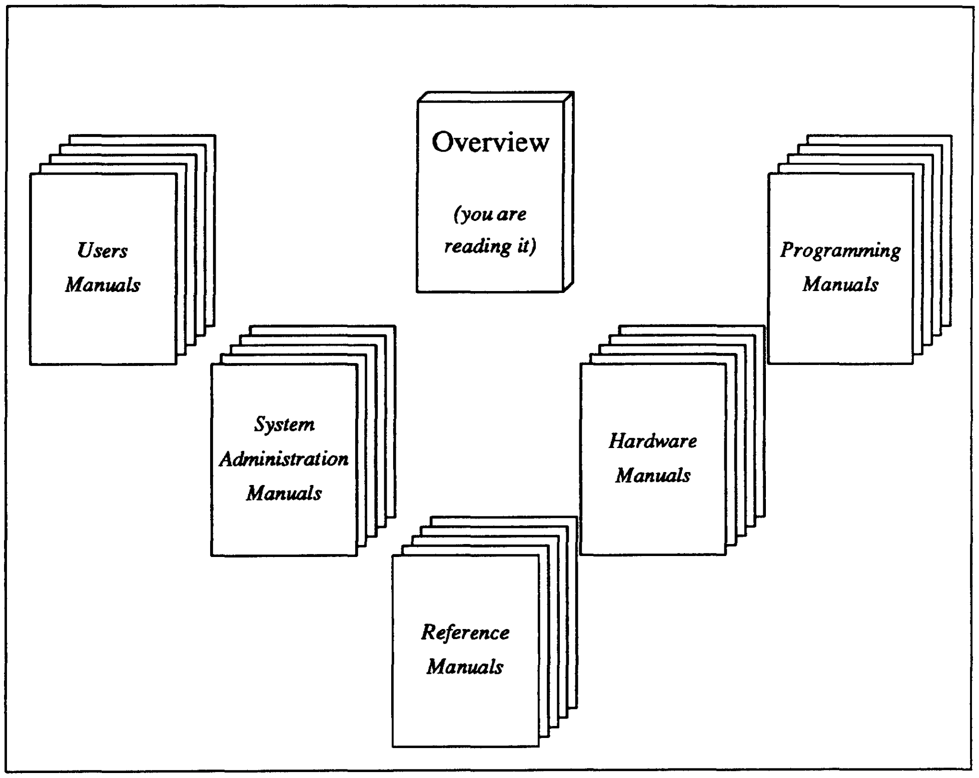
Sun Technical Documentation

17

Sun Technical Documentation

Sun Microsystems supplies a comprehensive package of documentation to supplement its hardware and software products. Much of the technical documentation is completely new and describes products and package that Sun Microsystems have added to the original base of UNIX software. Other areas of the documentation evolved from the original UNIX system manuals that were produced at Bell Laboratories and at University of California, Berkeley. The Technical Publications group at Sun Microsystems have made extensive revisions and improvements to the original base. Sun Microsystems' manuals are grouped into five document 'families' as shown in Figure 17-1.

Figure 17-1 *Families of Technical Manuals*

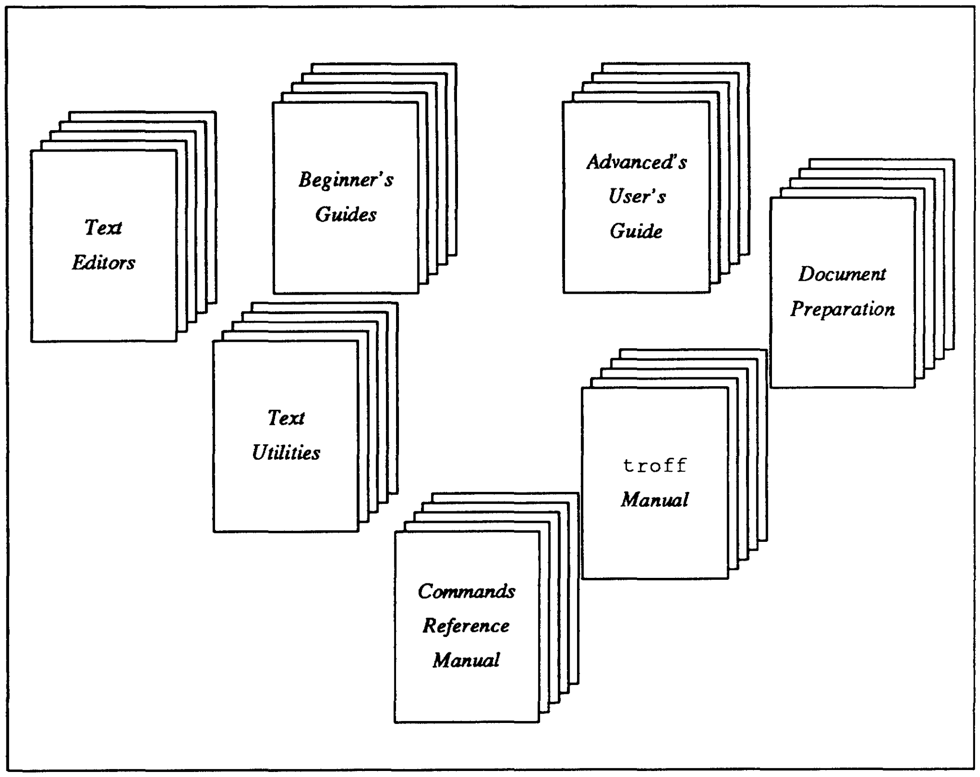


Some families are oriented specifically to *users*, with the emphasis of using the existing base of applications software. In this category we group the User's Manuals and the System Administration Manuals.

17.1. User's Guides

User guides in Sun Microsystems documentation look as in this picture

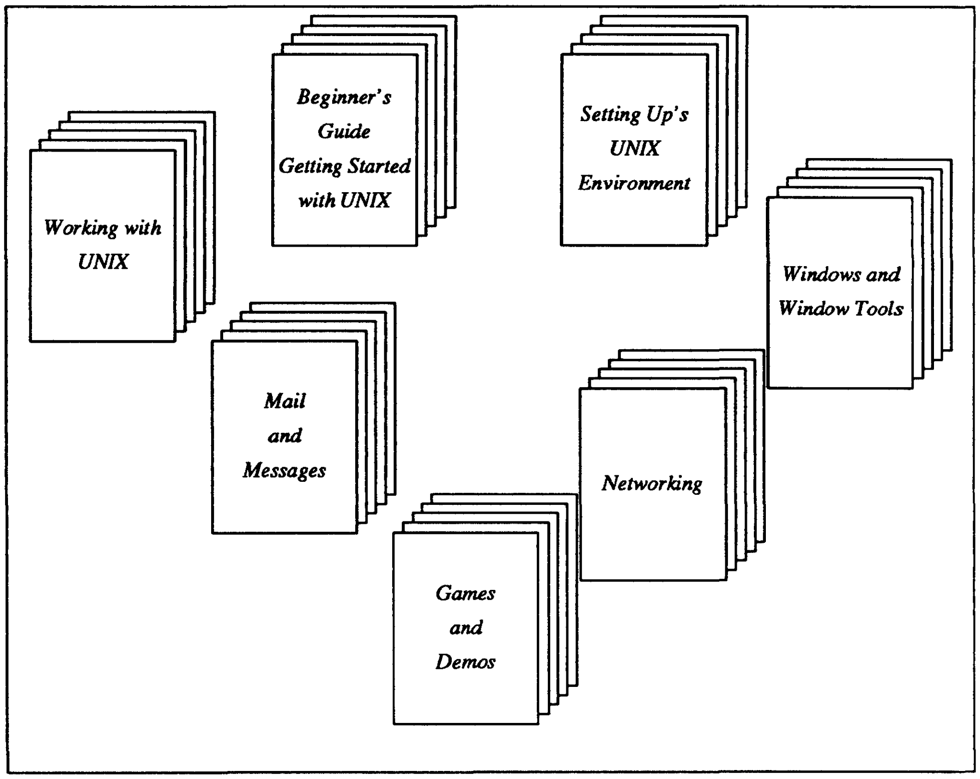
Figure 17-2 *Users Guides*



17.2. Beginner's Guides

There are a collection of *Beginner's Guides* aimed at people who are new to the UNIX operating system and the Sun environment. These beginner's guides are as shown in this picture:

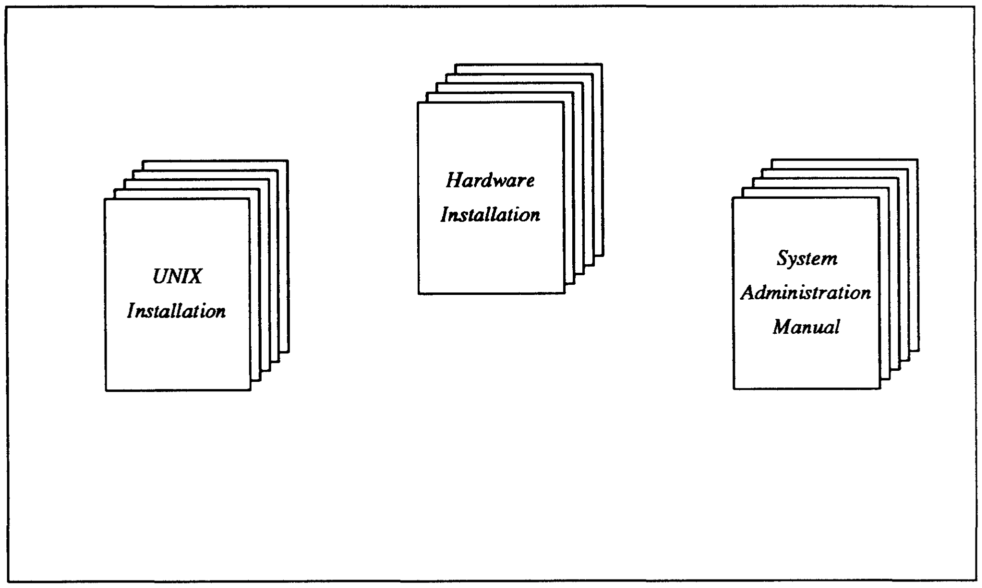
Figure 17-3 *Beginner's Guides*



17.3. System Administration Guides

Includes system installation, configuration, boot, and maintenance procedures, mail system and networking facility set-up information, and a reference manual for commands and utilities of use to system managers.

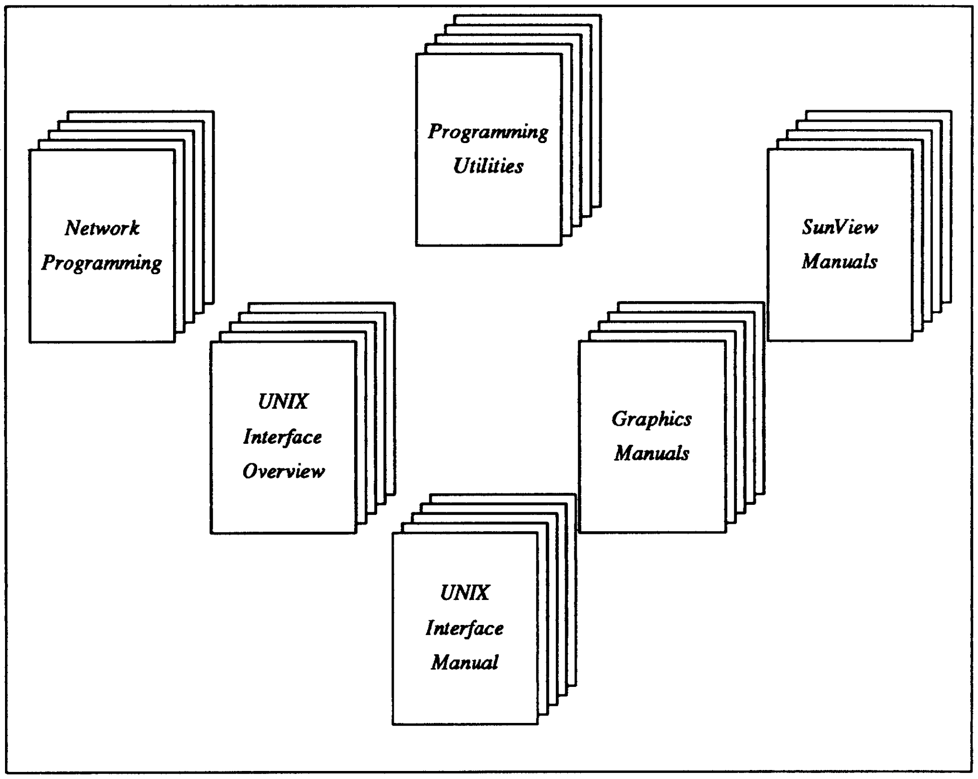
Figure 17-4 *System Administration Guides*



17.4. Programming Manuals

Programming manuals cover a number of distinct areas. The overall picture of programming documentation looks like this:

Figure 17-5 *Programming Manuals*



17.5. Reference Manuals

There are two major *reference manuals* in Sun's suite of documentation:

- *UNIX Interface Reference Manual for the Sun Workstation*
- *Commands Reference Manual for the Sun Workstation*

These two manuals were derived from the 'man' pages of the original UNIX system *Programmer's Reference Manual*.

Commands Reference Manual for the Sun Workstation

The *Commands Reference Manual* is one of the two major *reference manuals* in the suite of documentation. The Commands Reference Manual is a 'dictionary'-oriented list of the commands that the user can type to the operating system. The Commands Reference Manual is oriented towards users using the everyday commands of the system to do work, and also to the system administrator managing the operation of the system. There are also descriptions of the public files and macro packages, plus a section on games and other trivial pursuits.

The organization of the Commands Reference Manual follows loosely the 'traditional' model of the UNIX system *Programmer's Reference Manual*.

UNIX Interface Reference Manual for the Sun Workstation

The *UNIX Interface Reference Manual* is one of the two major *reference manuals* in the suite of documentation. The UNIX Interface Reference Manual is a 'dictionary'-oriented list of the programming interfaces to the operating system and applications libraries. The UNIX Interface Reference Manual is oriented towards programmers writing C-language programs. It contains descriptions of system calls, subroutines from various libraries, characteristics of special files (devices), and formats of files.

The organization of the UNIX Interface Reference Manual follows loosely the 'traditional' model of the UNIX system *Programmer's Reference Manual*.

17.6. History of Sun Microsystems Manuals

Here is a description of how Sun Microsystems' technical manuals differs from the 'traditional' UNIX system documentation. We focus here on the problems that the marketplace perceives with 'standard' UNIX system documentation and what we at Sun Microsystems have done to correct this.

Bell Laboratories Manuals

Taking the Version 7 UNIX system as a starting point, the UNIX system manuals appeared in three books:

- *The UNIX Programmer's Reference Manual Volume 1* contained the 'man' pages — an eight-section manual containing an alphabetically ordered collection of pages describing various aspects of both the user (command) interface to the UNIX system and its utility software, and the programming interfaces to the operating system and application libraries.
- *The UNIX Programmer's Reference Manual Volume 2A* contained a collection of papers describing software to assist developing computer programs.
- *The UNIX Programmer's Reference Manual Volume 2B* contained a collection of papers describing text manipulation and document preparation tools.

Berkeley Enhancements to the Manuals

The Computer Science Research Group at the University of California at Berkeley made many enhancements to the UNIX operating system. They also added *The UNIX Programmer's Reference Manual Volume 2C* containing papers on the `ex` and `vi` text editors, the Berkeley implementation of the Pascal programming language, the (now defunct) `sdb` debugger, and various papers on the document preparation tools including *The Berkeley Font Catalog*. The Berkeley group also added a huge number of new 'man' pages to reflect both new commands and new programming interfaces in the Berkeley UNIX system.

Problems with the UNIX Manuals

When Sun Microsystems 'inherited' the Bell Laboratories and Berkeley UNIX system manuals we set out to improve the documentation in response to the problems as users in the marketplace perceived them. Here are some of the criticisms that were heard:

- Installation and Administration instructions were largely non-existent — much of the necessary detail is scattered throughout a collection of apparently unrelated pages organized in alphabetical order.
- The system is inaccessible *via* the manuals.

- Almost impossible to find needed information in the UNIX manuals. Inadequate motivation. Lack of locators (tables of contents, indexes).
- The marketplace levels strong criticism against companies who just copy the UNIX manuals and ship them without further change.
- The 'man' page paradigm is largely useless in a system attempting to describe user interfaces based on graphics and bit-mapped screens with menus and icons.

Sun Microsystems Strategy

We set out with the goal of producing high-quality user documentation to make the system easier to access. At the start we had to define our target audience — who are these manuals aimed at?

Target Audience

> Scientific and technical users with (possibly some) computer programming experience who are neither expert UNIX programmers nor expert UNIX users. We could assume a reasonable level of technical expertise on the part of our users and not feel that we had to write for the mass-market.

The ramifications of having this kind of audience are that we could change the organization of the whole slew of UNIX manuals to bring them closer to what non-UNIX users would expect. The documentation must be addressed more specifically to users doing a day-to-day job, rather than towards users doing computer-science research for the benefit of other computer scientists.

Evolution *versus* Revolution

We resisted the strong temptation to trash the Bell Laboratories and Berkeley manuals and start over from scratch — that was seen as too big a step. We had to work with what was supplied and improve on that. We focused on making the manuals reflect the system as accurately as possible — truth first, beauty second. Our starting point was an incomplete version of 4.2 BSD so we were tracking a moving target.

Our first priority was a setup and installation guide for the Sun Workstation and the UNIX operating system running on that hardware. This manual evolved over time into the current *Installing UNIX on the Sun Workstation* and *System Administration for the Sun Workstation* manuals.

Functional Organization

The secondary priority was to identify the logical 'families' into which the existing and future documentation would fit and write the manuals to that model. The narrative material from the 'man' pages was to be moved into a series of 'textbooks' organized in functional rather than alphabetically, leaving the 'man' pages as the terse reference dictionary they were originally intended to be.

18

Third Party Software

18 Third Party Software

Sun Microsystems' success in the marketplace can be attributed in large measure to the number of third party software vendors that have implemented their specialized application software to run on Sun systems.

18.1. Catalyst — the Third Party Program

The third party program is an ongoing part of Sun's marketing organization. At the time of writing (Autumn 1985) there are more than 350 third party vendors in the *Catalyst* program. Here are the major application areas as listed from the Catalyst catalog. For detailed information and people to contact, please refer to the appropriate section of the Catalyst catalog.

- Add on hardware.
- Architectural and Civil Engineering.
- Artificial Intelligence.
- Communications.
- Data Base Management Systems.
- Earth Resource Engineering.
- Electrical Engineering.
- Graphics.
- Mathematics and Statistics.
- Mechanical Engineering.
- Office Automation.
- Software Engineering.
- Miscellaneous.

18.2. 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 encourage exchange of information between Sun Workstation users and Sun Microsystems, *and* to collecting and disseminating techniques, software, documentation, procedures, and related interesting information.

Donated Software Tape

The Sun User Group offer 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 thhat Sun Microsystems supports directly.
- Editors and spreadsheets.
- Mail handlers.
- Games.
- 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.

Index

Revision History

Revision	*Date*	*Comments*
A	15 February 1986	First release of this Manual.

Notes

Notes

Notes

Notes

Notes