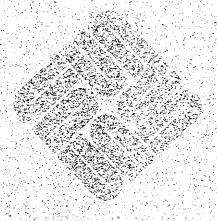
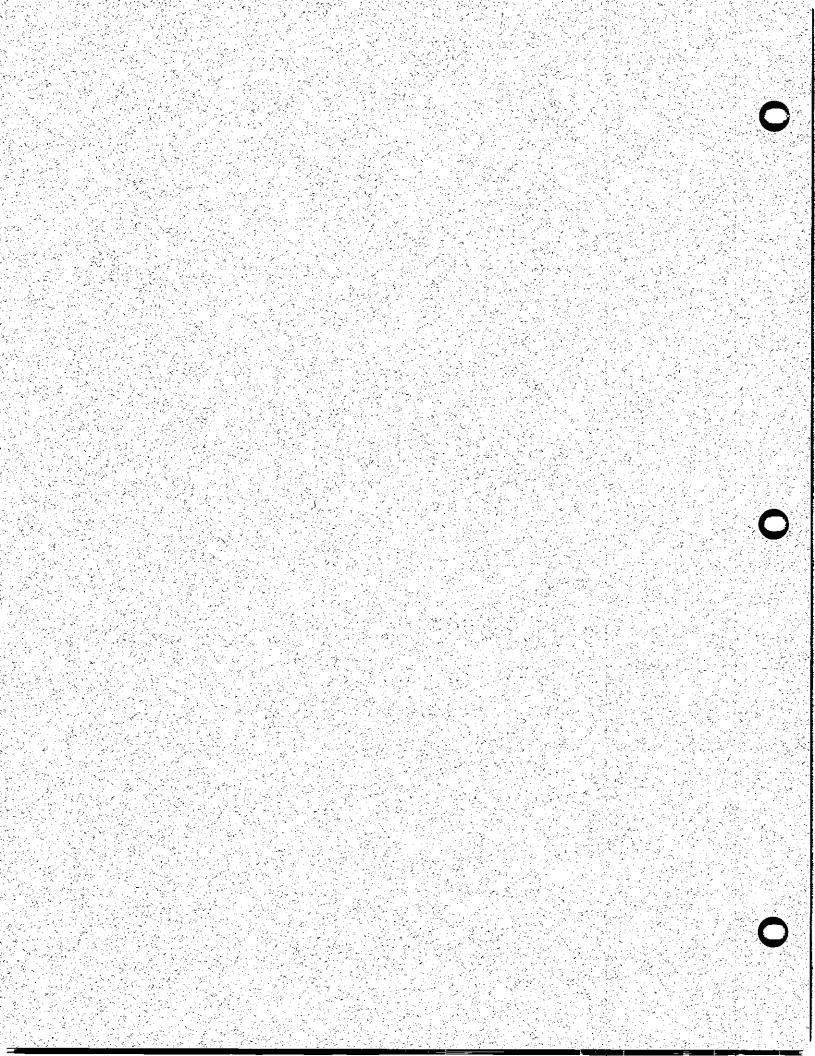


# Software Technical Bulletin *April 1988*

Software Information Services

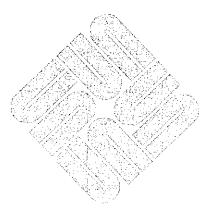






# Software Technical Bulletin April 1988

Software Information Services



Software Technical Bulletins are distributed to customers with software/hardware or software only support contracts. Send comments or corrections to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Ave., M/S 2-312, Mountain View, CA 94043 or by electronic mail to *sun!stb-editor*. U.S customers who have technical questions about topics in the Bulletin should call the Sun Customer Software Services AnswerLine at 800 USA-4-SUN. Other customers should call the numbers listed in *World Hotlines* appearing in Section 1.

UNIX, UNIX/32V, UNIX System III, and UNIX System V are trademarks of AT&T Bell Laboratories. DEC, DNA, VAX, VMS, VT100, WPS-PLUS, and Ultrix are registered trademarks of Digital Equipment Corporation.

Courier 2400 is a trademark of U.S. Robotics, Inc.

Hayes is a trademark of Hayes Microcomputer Products, Inc.

Multibus is a trademark of Intel Corporation.

PostScript and TranScript are trademarks of Adobe Systems, Inc.

Ven-Tel is a trademark of Ven-Tel, Inc.

Sun-2, Sun-2/xxx, Sun-3, Sun-4, Deskside, SunStation, Sun Workstation, SunCore, DVMA, SunWindows, NeWS, NFS, NSE, SPARC™, SunUNIFY™, SunView™, SunGKS, SunCGI, SunGuide, SunSimplify, SunLink, Sun Microsystems, SunOS™, and the Sun logo are trademarks of Sun Microsystems, Inc.

UNIFY™ is a trademark of Unify Corporation.

ENTER, PAINT, ACCELL, and RPT are trademarks of Unify Corporation.

SOL<sup>TM</sup> is a trademark of International Business Machines Corporation.

Applix® is a registered trademark of Applix, Inc.

SunAlis™ is a trademark of Sun Microsystems, Inc. and is derived from Alis, a product marketed by Applix, Inc.

SunINGRES™ is a trademark of Sun Microsystems, Inc. and is derived from INGRES, a product marketed by Relational Technology, Inc.

VEGA Delux is a trademark of Video Seven, Inc.

Micro Enhancer Delux is a trademark of Everex Systems, Inc.

VxWorks is a trademark of Wind River Systems, Inc.

Cabletron is a trademark of Cabletron Systems.

Apple and Laser Writer<sup>™</sup> are trandemarks of Apple Computer, Inc.

PostScript® is a registered trademark of Adobe Systems, Inc.

#### Copyright © 1988 by Sun Microsystems.

This publication is protected by Federal Copyright Law, with all rights reserved. No part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, or by any means manual, electric, electronic, electro-magnetic, mechanical, chemical, optical, or otherwise, without prior explicit written permission from Sun Microsystems.

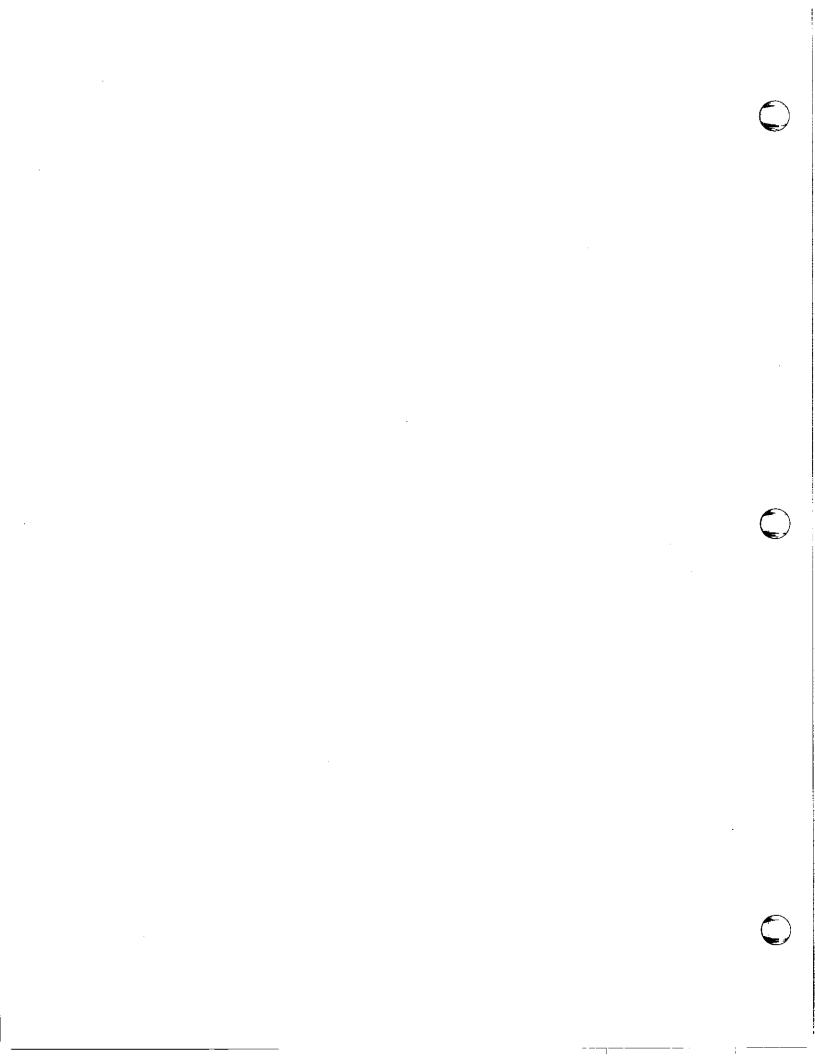
# Contents

	<b>5</b> 93
17-47-017-027-027-027-027-027-027-027-027-027-02	593
100010-41	595
	. 643
	643
	644
	646
	647
	2000000
	。 653
765000, 750000, 77	

SunOS 4.0 Diagrams	666
Section 5 QUESTIONS, ANSWERS, HINTS, AND TIPS	679
Q&A, and Tip of the Month	679
Section 6 THE HACKERS' CORNER	687
VMS Tape Backup	687
Section 7 CUMULATIVE INDEX: 1988	715

# **NOTES & COMMENTS**

NOTES & COMMENTS	593
Editor's Notes	593
Software Release Levels	
Dependency Tables	
World Hotlines	
STB Duplication	
Errata	605



### **NOTES & COMMENTS**

**Editor's Notes** 

**Editor's Notes** 

Expanded Current Sun Software Products and Release Level Tables

Hardware and Software Release Dependency Tables

World Hotlines

STB Duplication Permission

**Errata Corrections** 

The April 1988 Software Technical Bulletin (STB) editor's notes include notes on the monthly software product release tables, new hardware and software release dependency tables, world hotlines for use by customers outside the U.S., STB duplication permission, errata corrections, an announcement of SunOS release 3.5.1, a release report describing SunOS release 4.0, and the Hackers' Corner.

The five tables showing current Sun software product release levels appear monthly. These tables show release levels for operating systems, communications products, unbundled languages, and unbundled applications.

New for this month's STB issue are the hardware and software release dependency tables. These tables detail what combinations of hardware and software products were released under a particular SunOS release level. The tables will be updated and reprinted quarterly. SunOS release levels supporting the hardware and software products are indicated by an 'x' in the appropriate table.

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

This notice is published monthly, giving customers useful information regarding ordering and duplicating additional STB copies.

This month's errata contains corrections to the November 1987 and the January 1988 STBs. These corrections were submitted by Sun customers, one of whom discovered a bug in using *hostid(1)* on Sun-2s.



Please note that Sun customers are urged to send in corrections to STB material for inclusion in the next 'Errata' column. Send your corrections to sun!stb-editor.

SunOS Release 3.5.1

The announcement for bug fixes contained in SunOS release 3.5.1 appears in this month's STB. See this article for a list of bugs fixed, and availability and ordering information for this release.

SunOS Release 4.0 Report and Diagrams

This month's In Depth feature includes a description of SunOS release 4.0 and how it compares to 4.3BSD and SVID. A series of nine charts summarize a comparison of SunOS 4.0 with 4.3BSD and SVID.

Please note that information contained in this article is subject to change, pending further development during the beta testing of SunOS 4.0. However, most information is expected to remain unchanged.

The Hackers' Corner

This month's **Hackers' Corner** includes program code to read a VMS-generated backup tape. It also converts the files to the UNIX format, and writes the files to disk.

Again, please note that such applications, scripts, or code are not offered as released Sun products, but as items of interest to enthusiasts wanting to try out something for themselves. They may not not work in all cases, and may not be compatible with future SunOS releases. Please consult your local shell script or programming expert regarding any application, script, or code problems.

Thanks.

The STB Editor



### **Software Release Levels**

As of February 17, 1988

### **Operating Systems**

Product Name	Current Release
SunOS (Sun-2 and Sun-3 Operating System)	3.5
Sys4 (Sun-4 Operating System)	3.2, Rev. 2

#### **Communications Products**

Product Name	Current Release
SunLink BSC3270	3.0
SunLink SCP	3.0
SunLink BSCRJE	5.0
SunLink Local 3270	5.0
SunLink SNA3270	5.0
SunLink Peer-to-Peer	5.0
SunLink IR	5.0
SunLink DDN	5.0
SunLink DNI	5.0
SunLink OSI	5.0
SunLink MCP	5.0
SunLink TE100	4.0
SunLink X.25	5.0
SunLink SCA	5.0



#### **Unbundled Languages**

Product Name	Current Release
Sun FORTRAN* (for Sun-2 and Sun-3 systems)	1.0
Sun FORTRAN* (for Sun-4 systems)	1.05
NSE	2.0
NeWS	1.0
Sun Common Lisp-D	2.1
Sun Common Lisp-E	1.1
Modula-2	2.0
Cross Compilers	2.0

\*Sun FORTRAN Note: The £77 compiler is automatically included with SunOS Release 3, which includes SunOS Releases 3.2, 3.4, and 3.5. Sun FORTRAN Release 1.0 (for Sun-2 and Sun-3 systems) and Sun FORTRAN Release 1.05 (for Sun-4 systems) are value-added products that support VMS extensions to the £77 compiler, and must be purchased separately from the operating system.

#### **Unbundled Graphics**

Product Name		Current Release
SunGKS	-	2.1

#### **Unbundled Applications**

Product Name	Current Release
SunAlis	2.1
SunINGRES	5.0
SunSimplify	1.0
SunUNIFY	2.0
Transcript	2.0
SunIPC	1.1
PC-NFS	2.0
SunTrac (for Sun-2 and Sun-3 systems)	1.0
SunTrac (for Sun-4 systems)	1.0/3.2



#### Current Sun Software Products and Release Levels

The preceding tables contain lists of current Sun software products and their respective current release levels.

You will note that the Software Technical Bulletin (STB) contains articles from time to time that detail technical changes in a given software product's next available release.

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

These tables appear monthly in the STB for your convenience.



#### **Dependency Tables**

Hardware and Software Release Dependency Tables The following series of tables illustrate support of hardware and software products by the Sun Operating System (SunOS) level in which the products were introduced. Key hardware features and software product support are shown in the left-hand column of each table. The Sun system and corresponding SunOS level(s) in which the product is supported are shown across the top of each table.

**Key to Codes Used in Tables** 

Two codes are used in the tables, as follows.

Key	Translation
х	Available and supported in this SunOS release
BT	Requires extra boot tape

These tables will be updated as needed, and appear in future issues of the STB on a quarterly basis.



### System Hardware and Operating System Dependencies

FEATURE	Sun-4	Sun-2 and Sun-3			
	Release Sys4-3.2	Release 3.2	Release 3.3	Release 3.4	Release 3.5
System Hardware Architecture:				**************************************	
Sun-3/60			<u>"</u>	BT	х
Sun-3/E			7.11	ВТ	x
Sun-3/100 and Sun-3/200		х	х	х	x
Sun-4/110	х	. ,			
Sun-4/2xx	х		· · · -		
System Hardware Features:					·
ALM2	х	· · · · · · · · · · · · · · · · · · ·	,	-	x
32 MB memory board	x				
900 MB disk	х				<del></del>
327 MB SCSI				x	X
Double buffering				· · · · · · · · · · · · · · · · · · ·	X
Operating System Installation:					<del> </del>
Remote tape installation	х	х	х	х	x
Diskless Sun-2 and Sun-3					
installation on a Sun-3 Server		х	х	Х	X
Diskless Sun-3 and Sun-4 installation on a Sun-4 Server*	x		······		



### **Bug Fixes and Improvements**

	Sun-4	Sun-2 and Sun-3			
FEATURE	Release Sys4-3.2	Release 3.2	Release 3.3	Release 3.4	Release 3.5
QIC-24 Distribution Media (Sun-3, Sun-4)	х				
SunPro make				х	Х
filemerge				х	x
Subnets			Х	х	Х
SCSI Disconnect/Reconnect			х	х	Х
SunOS Release 3.3 bug fixes			х	х	х
SunOS Release 3.4 kernel bug fixes			<u> </u>	х	х
SunOS Release 3.4 SunView bug fixes	Х			X	х
SunOS Release 3.5 bug fixes					х

### **Bundled Software Products**

FEATURE	Sun-4	Sun-2 and Sun-3			
	Release Sys4-3.2	Release 3.2	Release 3.3	Release 3.4	Release 3.5
FORTRAN-77		х	х	x	Х
pc (Pascal)		х	х	х	X
SunView Rel. 1.5		Х	х		
SunView Rel. 1.7	X			х	x



### **Unbundled Software Products**

	Sun-4	Sun-2 and Sun-3							
FEATURE	Release Sys4-3.2	Release 3.2	Release 3.3	Release 3.4	Release 3.5				
Sun FORTRAN Rel. 1.05	x			<del></del>					
Sun Pascal Rel. 1.05	х								
NeWS Rel. 1.1	х	Х	х	X					
NSE Rel. 1.0		-	<u> </u>	x	x				
Cross-Compilers Rel. 2.0:			<u>.                                    </u>		I				
Sun-2 to Sun-3 or Sun-4		X	X	x	x				
Sun-3 to Sun-2 or Sun-4		х	X	X	x				
Sun-4 to Sun-2 or Sun-3	х			<del></del>					
Sun Fortran Rel. 1.0		x	Х	X	x				
SunINGRES Rel.5.0		Х	X	X	x				
SunUNIFY Rel. 2.0		Х	X	X	<u>x</u>				
SunSimplify Rel. 1.0		Х	Х	X	X				
SunAlis Rel. 2.1	-	X	X	x	X				
SunGKS Rel. 2.1	х	X	X	x	X				
Modula-2 Rel.1.0		X	X	x	X				
Modula-2 Rel.2.0				x	x				
PC-NFS Rel. 2.0	х	x	x	x	X				
PC-NFS Toolkit	X	x	x	- X	x				
SunIPC Rel. 1.1	<del>-</del>	x	x	x	x				
Sun Common Lisp Rel. 2.1		X	x	X	X				
TranScript Rel 2.0		x	x	$\frac{x}{x}$	X				
SunTrac Rel. 1.0		x	X	$\frac{x}{x}$	<u>x</u>				
SunTrac Rel. 1.0-3.2	х								



# **SunLink Communications Software Products**

	Sun-4	Sun-2 and Sun-3							
FEATURE	Release Sys4-3.2	Release 3.2	Release 3.3	Release 3.4	Release 3.5				
BSCRJE Rel. 5.0		х		х	х				
Channel (N/A for Sun-2) Rel. 5.0		Х	<u></u>	X	X				
Local 3270 (N/A for Sun-2) Rel. 5.0		х		х	X				
DDN Rel. 5.0		х		х	X				
DNI Release 5.0		x		x	X				
DNI Release 5.1	Х								
IR Rel. 5.0		x		х	Х				
MCP (N/A for Sun 2) Rel. 5.0		х		х	х				
OSI Rel. 5.0		х		х	x				
SNA 3270 Rel. 5.0		х	<u> </u>	X	X				
SNA 3270 Rel. 5.1	X								
SNA Peer-to-Peer Rel. 5.0		х		x	X				
X.25 Rel. 5.0		х		х	х				
X.25 Rel. 5.1	х			<u></u>	1				
TE100 Rel. 5.0		х		х	х				
TE100 Rel. 5.1	х			<u> </u>	<u> </u>				



### World Hotlines

World Hotlines	Sun Customers throughout the world have service hotlines available for both software and hardware support questions. The service hotlines are shown below. If your country is not shown in the table, please phone your local Sun sales office.									
Australia	Sun Australia Lionel Singer Group	(011-61-2) 957-2522 (011-61-2) 957-2655								
Canada	Montreal Branch Ottawa Vancouver Branch Western Branch	(514) 879-1914 (613) 748-9617 (604) 641-1296 (403) 295-0150								
France	Paris Sun Microsystems France SA	(33) 1 4630 2324								
Germany	Munich Sun Microsystems GmbH	(49) 89/95094-321								
Japan	C. Itoh Data Systems Nihon Sun	(011-81-3) 497-4676 (011-81-3) 221-7021								
The Netherlands	Soest Sun Microsystems Nederland BV	(31) 2155 24888								
Switzerland	Zurich Sun Microsystems Schweiz AG	(41) 1 828 9555								
United Kingdom	Albany Park Sun Microsystems UK Ltd	(44) 0276 691052								
United States	All, including Puerto Rico	1-800-USA-4-SUN								

All countries outside the

USA, Europe, and northern Africa



Intercon

(415) 691-6775

#### **STB Duplication**

#### **Duplicating the STB**

Your company's software support contract includes a monthly issue of the STB, which contains a quarterly, updated Customer Distributed BugsList (CDB). Each month, the copy of your STB is mailed to your company's primary contact person or department. Sites with more than one contract may receive more than one STB copy, depending on how the contracts are set up.

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

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

#### Direct STB Purchase

The STB is sent to the primary contact person named in all software support contracts. Sun is looking into methods by which customers holding these contracts may purchase extra copies directly.

Look to this column for an announcement regarding the purchase of extra STB copies.

#### **Further Questions**

If you have any questions, comments, or articles regarding the STB or CDB, please send your ideas and questions to *sun!stb-editor*.



Errata

**Errata** 

November 1987 STB

January 1988 STB

This month's errata includes changes to the November 1987 and to the January 1988 Software Technical Bulletins.

The November 1987 STB issue 'Tip of the Month' appears on page 798 and describes the use of *hostid(1)* to determine information about the machine you are using.

The command output should be an eight-digit string. A customer has pointed out that *hostid(1)* does not display the first digit, a zero, in the case of a Sun-2. Therefore, if you have a Sun-2 with a VME bus, the first digit you see is a '2'. This is misleading since it suggests you may be running a Sun-4.

A bug is being filed to correct this problem with *hostid(1)* when run on a Sun-2. The command output is correct for Sun-3s.

The following paragraphs contain corrections to the article entitled 'Yellow Pages' starting on page 32, and to 'Network Transfers' starting on page 114.

□ Yellow Pages, page 34:

Change the pathnames to ypwhich, ypcat, and ypmatch from /etc to /usr/bin.

Change the pathname to yppoll from /etc to /usr/etc/yp.

□ Network Transfers, page 116:

In the last paragraph, change -r to -n.



.

# **ARTICLES**

ARTICLES	609
SunOS Release 3.5.1	609
dbxWorks Special	615
Using plot(1G)	617
Using vmstat(8)	
SunOS Sys4-3.2 Usage	625
Asynchronous Modems	626
Sun Modula-2 Release 2.0	629
C Hunt	633
NeWS Release 1.1	625

.

### **ARTICLES**

#### SunOS Release 3.5.1

#### SunOS 'Dot Dot' Releases

SunOS Release 3.5.1 Availability

Sun Microsystems releases tapes containing bundled patches every two or three months, between other SunOS releases. These releases are called 'Dot Dot' releases. This article contains the announcement of SunOS release 3.5.1, a list of specific fixes, fix reference numbers, and a synopsis of each corrected problem.

SunOS release 3.5.1 is available at no charge to Sun customers holding software support contracts. Sun customers under warranty may receive the release if it fixes an observed problem. Other Sun customers wishing to purchase this release may do so for \$200 USD.

To request or order a release, please call 1-800-USA-4-SUN and request the release by its 'Dot Dot' number (3.5.1), or by the Order Management and Retrieval (OMAR) number appearing in the next paragraph and in the Customer Support price list. For Sun Europe customers, please call your local support group or sales representative.

Please note that SunOS release 3.5.1 may be installed only on those systems already running SunOS release 3.5.

# SunOS Release 3.5.1 Ordering Information

Use the information below to order SunOS release 3.5.1.

A list of release contents appears at the end of this article. Use these two lists to determine whether you need either release.



#### SunOS Release 3.5.1

Description	CPU-type	Media S	Size	OMAR#	Unit Pr	ice
RTF, & Tape	68010	1/4"	DO	Γ2-01-3.5.1	\$200	
RTF, & Tape	68010	1/2"	DO	Γ2-02-3.5.1	\$200	
RTF, & Tape	68020	1/4"	DO:	Г3-01-3.5.1	\$200	
RTF, & Tape	68020	1/2"	DO	Г3-02-3.5.1	\$200	

#### SunOS Release 3.5.1

A list of SunOS 3.5.1 fixes, fix reference numbers, and a synopsis for each solved problem appears below.

boot, Reference Number: 1005328

Synopsis: booting a bogus file from tty fails to finish

boot, Reference Number: 1005731

Synopsis: kernels larger than 0xa0000 will not boot successfully

□ clock, Reference Number: 1007764

Synopsis: the tod clock driver calculates the wrong value in leap years

crash, Reference Number: 1007745

Synopsis: crash dumps broken on systems with SCSI

ethernet, Reference Number: 1006375

Synopsis: ie0: lost interrupt: resetting

gp, Reference Number: 1006687

Synopsis: GP destroys pattern when replicating to left

gp, Reference Number: 1006691

Synopsis: GP does not process texture reference point correctly

gp, Reference Number: 1008267

Synopsis: problem erasing and redrawing same area using GP and null



gp2\_mapping, Reference Number: 1007093

Synopsis: GP2 bug in gpone driver

□ gp2 restart, Reference Number: 1008072

Synopsis: GP2 reset problem

□ in.routed, Reference Number: 1007398

Synopsis: in.routed in 3.5 can sometimes drop core

kernel, Reference Number: 1002990

Synopsis: /sys directory does not contain vax directory/header files

maxusers, Reference Number: 1004461

Synopsis: default maxusers value in kernel configuration files too small

msgrcv, Reference Number: 1006823

Synopsis: msgrcv(2) panics system with 'bad rmfree'

nd, Reference Number: 1008082

Synopsis: under heavy loads and hangs

printer, Reference Number: 1004208

Synopsis: writing to /dev/printer causes system panic

profiling, Reference Number: 1007904

Synopsis: profiling not turned on properly

□ rc, Reference Number: 1005034

Synopsis: EtHost files are not removed from /tmp at boot time

scsi diag, Reference Number: 1008033

Synopsis: SCSI drivers need to return more explicit error codes

scsi driver, Reference Number: 1007919

Synopsis: latest 5380 logic devices chip for Sun 3/60 needs mods to si.c driver



□ scsi\_driver, Reference Number: 1007921

Synopsis: problem restarting SCSI disk driver after timeout

□ scsi host, Reference Number: 1007653

Synopsis: SCSI host adapters are not set up to process sc\_conf.c correctly.

scsi\_id, Reference Number: 1008034

Synopsis: SCSI tape driver needs to return id of tape device.

scsi\_tape, Reference Number: 1007920

Synopsis: intermittent SCSI tape driver hangs

sendmsg, Reference Number: 1005177

Synopsis: sendmsg, recvmsg, writev, readv fail when iov\_len is zero

□ shared mem, Reference Number: 1006702

Synopsis: application crashes 3/200 series kernel

show, Reference Number: 1007619

Synopsis: /usr/demo/show: panics system if use a bad rasterfile

□ shutdown, Reference Number: 1001124

Synopsis: after remote /etc/shutdown, console is still in raw mode

shutdown, Reference Number: 1005353

Synopsis: remote shutdown leaves tty console unresponsive

silo, Reference Number: 1006666

Synopsis: silo overflows from mouse

sunpro, Reference Number: 1006484

Synopsis: sunpro make does not evaluate dynamic macro \$? in implicit rule



sunpro, Reference Number: 1006485

Synopsis: sunpro make does not search current directory to execute command

sunpro, Reference Number: 1006518

Synopsis: Dynamic macro, \$0, fails to find current target and make hangs

sunpro, Reference Number: 1006544

Synopsis: make command line arguments are now context sensitive

u sunpro, Reference Number: 1006590

Synopsis: sunpro make does not allow name of makefile to be a target

u sunpro, Reference Number: 1006595

Synopsis: sunpro make does not expand some dynamic macros with -e flag

sunpro, Reference Number: 1006791

Synopsis: sunpro make sometimes misses double-colon rules

□ sunpro, Reference Number: 1006794

Synopsis: sunpro make wrongly gives fatal error for dependencies

□ sunpro, Reference Number: 1007009

Synopsis: backslash confuses sunpro make about line numbers

□ sunpro, Reference Number: 1007010

Synopsis: sunpro make does not properly expand macros with "\" 'in them

sunpro, Reference Number: 1007931

Synopsis: long file names in libraries break .make.state

sunpro, Reference Number: 1007951

Synopsis: make continues building from sources known to be inconsistent



sunpro, Reference Number: 1008029

Synopsis: append-style assignments with no value add garbage to value of macro

t\_intrc, Reference Number: 1004782

Synopsis: t\_intro name conflicts in header files

uio.h, Reference Number: 1004750

Synopsis: /usr/include/sys/uio.h does not prevent multiple include

uuxqt, Reference Number: 1007738

Synopsis: uuxqt broken

xy, Reference Number: 1007524

Synopsis: xy and xd watchdog routines incorrect

□ zs, Reference Number: 1002756

Synopsis: zs interrupts invisible to perfmeter



#### dbxWorks Special

# Remote Cross-Debugging for Real-Time Development

Sun Consulting now has **dbxWorks** available, a Consulting Special which provides source-level cross-debugging of a process running on a VxWorks target. Sun Consulting's dbxWorks replaces the standard dbx, is fully compatible with dbxtool, and has no impact on local debugging.

Sun Consulting has received many questions about real-time support on Sun hardware. In some cases, customers using Sun workstations for software development want a similar development environment for use with real-time applications and have also requested a real-time executive on Sun hardware.

Wind River Systems, a third-party vendor in Emeryville, California have ported their real-time executive, VxWorks, to the Sun-3/E. Among its other capabilities, VxWorks provides the user with a remote login facility across the standard ethernet. VxWorks also provides symbolic debugging, though at an assembler language level, rather than at the source language level.

With dbxWorks, source-level debugging is possible in the VxWorks environment. The interface is essentially identical to that for local debugging using dbx and dbxtool. This interface includes source display, variable tracing, setting breakpoints, and single-stepping. The only difference is that the target process being debugged runs on a remote system on the network, under a non-UNIX operating system, VxWorks.

The VxWorks debugging facilities are still available. The user pops up an extra window; remote login provides access to VxWorks's assembler- and process-level debugging. Because the process being debugged is running on a remote system, its output will appear in this remote-login window, or on the VxWorks system console, rather than in the dbxtool window.

Sun Consulting's dbxWorks special is available for SunOS release 3.4 at this time; availability under 3.5 and future 3.x releases will be announced. VxWorks version 3.21 is required for proper operation of dbxWorks.

Ordering Information

For further information or to place an order for dbxWorks or VxWorks, contact your sales sales representative, or refer to the contacts shown below.

dbxWorks

Lisa Ventresca (415) 691-2438

sun!lisav

VxWorks

Wind River Systems, Inc.

(415)428-2623 sun!wrs!inquiries



For Further Information on SunOS and Real-Time

For those interested in a discussion of current SunOS capabilities and real-time applications, see the article entitled 'SunOS and Real Time' appearing on page 297 of the March 1988 STB, part number 812-8801-03.



Using plot(1G)

Using *plot(1G)*: A Working Example

The plot(1G) program reads plotting instructions, detailed in plot(5), from the standard input and produces plotting instructions suitable for a particular terminal on the standard output.

This article contains a working example for using plot(1G), including a sample input file.

tar, Optimum Block Size, and plot(1G) Output

You can use plot(1G) output to see test results to determine the best block size for archiving tapes. This example uses a Sun-3/75 running SunOS release 3.2.

This example input was derived from taring the directory /usr/lib with a long-running shell script that looped through successive tar commands. The test started with a block size of 100, incremented the block size by 50, and continued to a block size of 2000. Each tar iteration was timed using time(IV).

The output you will see using the shell script and input data file in this article is graphed using graph(1G), plot(1G), and tektool(1). graph(1G) outputs a binary format readable by plot(1G), which is not otherwise easily comprehended.

The Example: A Procedure

To see the graphic results, use the procedure shown below.

- 1. Save the shell script near the end of this article in a file named display data.
- Enter the command 'chmod 755 display\_data'.
- 3. Save the example input file at the end of this article in a file named tar\_data.
- Enter the command 'display\_data tar\_data'.

The plot(1G) Output

Unfortunately, plot(1G) does not print y-axis values along the y-axis, so reading the graph takes some care. It is a graph of the y-axis, dependent variable, time (in seconds), plotted against the x-axis, independent variable, block size (in kilobytes).

The y-axis runs from 0 to 500 seconds in increments of 60 seconds, and the x-axis runs from 100 to 2000 kilobytes in increments of 50 kilobytes. The numeric ranges are described cryptically at the bottom of the plot by:

100 -x - 2000 0 -y - 500



The resulting curve is fairly dramatic. tar of /usr/lib takes less than 150 seconds for blocking factors over 600 kilobytes. It is less so for subtrees smaller than /usr/lib, such as /etc, but it is still interesting.

#### For Further Information

See the manual pages for the commands listed below for further information.

- $\Box$  graph(1G)
- $\Box lpr(1)$
- $\neg plot(1G)$
- $\neg plot(3X)$
- □ *plot(5)*
- □ tektool(1)
- $\Box$  time(1V)

#### The Display Script

The display script is shown next, to be saved into the file display\_data.

```
#!/bin/csh -f
#
awk ' {printf "%d %s\n", $1,$4}' $1 | tr : ' ' | \
awk ' {printf("%d %d\n",$1,($2*60)+$3)}' >/tmp/graphfile

tektool -c \
"cat /tmp/graphfile| graph -x 100 2000 50 -y 0 500 60|plot -Ttek;\
sleep 60"
rm /tmp/graphfile
```

The Example Input File

The example input is shown next, to be saved into the file tar data.



```
100 2.8u 5.8s 6:59 2% 2+7k 1076+9io 0pf+0w
150 2.8u 6.0s 5:04 2% 2+11k 1084+10io 0pf+0w
200 3.0u 6.0s 4:08 3% 2+14k 1091+14io 0pf+0w
250 2.4u 6.1s 3:38 3% 2+15k 1087+13io 0pf+0w
300 3.2u 6.1s 3:18 4% 2+18k 1104+16io 0pf+0w
350 2.7u 5.8s 2:54 4% 2+21k 1086+11io 0pf+0w
400 2.7u 6.2s 2:45 5% 2+24k 1084+11io 0pf+0w
450 2.7u 5.9s 2:34 5% 2+26k 1089+15io 0pf+0w
500 3.1u 5.9s 2:27 6% 2+28k 1086+13io 0pf+0w
550 2.9u 6.3s 2:21 6% 2+31k 1082+12io 0pf+0w
600 2.7u 6.4s 2:15 6% 2+33k 1080+12io 0pf+0w
650 3.1u 6.1s 2:17 6% 2+34k 1080+12io 0pf+0w
700 2.9u 6.3s 2:10 7% 2+37k 1083+12io 0pf+0w
750 3.0u 6.1s 2:09 7% 2+39k 1083+14io 0pf+0w
800 3.0u 6.3s 2:15 6% 2+40k 1086+14io 0pf+0w
850 2.6u 6.2s 2:05 7% 2+42k 1084+14io 0pf+0w
900 3.0u 5.8s 2:11 6% 2+45k 1084+14io 0pf+0w
950 2.9u 6.4s 2:04 7% 2+48k 1085+14io 0pf+0w
1000
        2.9u 6.0s 2:02 7% 2+48k 1083+14io 0pf+0w
1050
        3.1u 6.3s 2:03 7% 2+48k 1079+14io 0pf+0w
1100
        3.5u 5.4s 2:00 7% 2+51k 1082+12io 0pf+0w
1150
        3.3u 6.1s 2:12 7% 2+53k 1080+15io 0pf+0w
1200
        3.4u 5.8s 1:58 7% 2+56k 1087+14io 0pf+0w
1250
        3.3u 6.1s 2:03 7% 2+56k 1079+14io 0pf+0w
1300
        3.3u 6.3s 2:05 7% 2+56k 1082+14io 0pf+0w
1350
        3.2u 5.9s 1:59 7% 2+57k 1084+14io 0pf+0w
1400
        3.5u 6.4s 2:06 7% 2+56k 1079+12io 0pf+0w
        3.7u 6.3s 2:07 7% 2+56k 1085+14io 0pf+0w
1450
1500
        3.2u 6.0s 2:03 7% 2+58k 1082+14io 0pf+0w
1550
        3.4u 6.0s 2:06 7% 2+63k 1080+7io 0pf+0w
1600
        3.lu 6.1s 2:09 7% 2+67k 1079+14io 0pf+0w
1650
        3.0u 6.5s 2:15 7% 2+68k 1083+14io 0pf+0w
1700
        2.9u 6.2s 2:04 7% 2+69k 1085+14io 0pf+0w
1750
        3.5u 6.0s 2:05 7% 2+68k 1079+14io 0pf+0w
1800
        3.1u 6.5s 2:04 7% 2+69k 1082+14io 0pf+0w
1850
        3.1u 6.1s 2:09 7% 2+69k 1084+13io 0pf+0w
1900
        3.2u 6.2s 2:13 7% 2+70k 1082+13io 0pf+0w
1950
        3.5u 6.2s 2:24 6% 2+70k 1083+13io 0pf+0w
```



#### Using vmstat(8)

#### vmstat(8) and Memory Management

Customers using *vmstat(8)* may confuse the meaning of two of the output fields in the cases of the amount of active virtual memory being used and the size of the free list. *vmstat(8)* reports both of these amounts in Kbytes.

The 'avm' column shows the amount of available virtual memory. Please note that the 'fre' column refers to the free list and does not refer to any amount of memory that is 'free' to use at that time. Thus, these two amounts do not add together to any total of available physical or virtual memory. This is because 'avm' reports virtual memory and 'fre' reports free physical memory.

# The vmstat(8) Manual Page and Sample Output

See the *vmstat*(8) manual page for a complete description of output fields. In general, *vmstat*(8) displays a summary of the virtual memory activity since the system was last booted. You can specify an interval (in seconds) or a count (in times) that you want your virtual memory activity reported.

An example of *vmstat(8)* output is shown below. Note that the 'avm' and 'fre' columns do not total either the physical or virtual memory on your system.

machin	e% <b>vms</b>	tat 5																	
procs	me	mory						p	age		(	lis	c f	ault	s			сри	1
r b w	avm	fre	re	at	рi	рo	fr	de	sr	s0	s1	s2	жЗ	in	sy	CS	us	sy	id
1 0 0	7288	616	0	0	0	0	0	72	0	0	0	0	0	3	28	7	4	1	95
0 0 0	7288	472	0	0	0	0	0	128	0	0	0	0	0	1	277	7	2	0	98
0 1 0	7704	392	0	0	16	8	40	88	52	11	4	0	0	17	213	27	6	11	83
1 0 0	7784	424	0	0	8	8	32	72	22	9	7	0	0	15	133	28	3	14	83
2 0 0	7840	440	0	0	0	8	8	72	15	5	7	0	0	20	185	30	6	17	78
0 0 0	7600	608	0	0	0	0	0	72	7	2	2	0	0	7	85	17	1	3	96
^C																			
machin	e%																		

### **SunOS Memory Management** and Hardware Overview

The following is an overview discussion of Sun3 memory management using the 68020 processor. This serves as an introduction to memory management concepts including the free list appearing in the *vmstat*(8) output. See the list of references at the end of this article for further information.



An overview of Sun3 memory management necessarily involves hardware considerations and a review of the origins of early UNIX versions based on the limitations of PDP-11 architecture. Depending on the PDP-11 model, a process was limited to 64-128 Kbytes. Multiprogramming was provided by swapping the entire process to disk.

VAX architecture added larger process address spaces supported by virtual memory on a minicomputer. Berkeley redesigned the memory management portion of the UNIX kernel. Sun's virtual memory is essentially a port of the Berkeley VAX design to Sun's architecture.

Segments, Pages, Page Frames, and Swap Devices

The UNIX operating system defines the segment as the basic unit of memory management. Each process has three segments for code (or text), data, and the stack. All processes running the same program share a single code segment. The data and stack segments are private for each process.

The PDP-11 Memory Management Units (MMUs) support UNIX segments well, but do not support virtual memory. However, the variable size of segments complicates finding places for them when they are brought in from disk to memory. Therefore, Sun3 architecture defines the basic unit of memory management as a fixed-size page.

The Sun3 MMU consists of a context register, a segment map, and a page map. Virtual addresses from the processor are translated into intermediate addresses by the segment map. They are then translated into physical addresses by the page map. The Sun3 MMU uses an 8-Kbyte page size, a 128-Kbyte segment size, and maps contexts using a virtual address space of 256 Mbytes each. See the Sun-3 Architecture: A Sun Technical Report listed at the end of this article for details.

In contrast to the large virtual address spaces for processes, the Sun3 architecture defines a single *physical* address space of 16 Mbytes. Note that a particular workstation may have as little as two Mbytes of physical memory. This limited physical address space is conceptually divided into page-size units called page frames.

In many cases there may not be enough page frames to hold all of the pages of a large process or the pages for several small processes. In these circumstances many of the pages are likely to be held temporarily on a disk, known as the swap device. The kernel shuffles or swaps pages between the swap device and the page frames to use the limited memory as efficiently as possible among the processes competing for the limited physical memory resources. This swapping is transparent to the processes and, usually, to users as well.

Memory States

A process can be considered to be in one of three states, depending on where the pages for that process are located at a given moment. The three states are swapped, resident, and mapped.



Swapped

A swapped process has its pages residing completely on the swap device, typically a disk. The kernel keeps the disk address of the process's user page in a process table. The kernel can retrieve the page table and segments from information recorded in the process's user page.

Resident

A resident process is partially resident in memory and partially swapped to disk. In this case the process has its user page and page table in memory. Some of its segment pages reside in memory as well.

Mapped

A mapped process is resident, and in addition, its page table is loaded in the system page map. Only mapped processes are runnable. The page map is a cache of eight page tables that are likely to be needed in the near future. You can regard the mapped 'state' as an optimization of the resident state.

The kernel changes the memory states of processes as necessary. When there are few processes and little demand for physical memory, all processes are mapped and memory management overhead is minimal. The kernel unmaps the least active processes and changes the more active ones from the resident to the mapped state as the number of processes exceeds the number that can be mapped at the same time. The kernel moves some processes between the resident and swapped states in order to maintain good performance under conditions of high contention for physical memory.

Page Maps and Contexts

To maintain good performance, there are actually eight page maps, one for each of seven processes, or contexts, and one dedicated to the kernel. An MMU register, called the context register and maintained by the kernel, points to the page map belonging to the running process.

Because there are eight page maps, the processor can be switched among the eight mapped processes by simply changing the context register. The overhead of loading a process's page table into a page map is incurred only when the process to be run is not mapped. When the kernel *must* map a new process, it overwrites the page map of the least-recently-run process, first updating that process's page table entries.

To speed overall response to interrupts and system calls, there are actually two context registers. One register selects the running process's page map; the other selects the kernel's page map. Which of the two context registers is used as the page map pointer for a particular instruction depends on the processor state. The processor may be running in 'user state' or 'supervisory state'.



Normally the processor runs in user state. Interrupts and system calls switch the processor to the supervisor state. The corresponding return instructions switch the processor back to the user state. Thus, the kernel need never change the context register in response to an interrupt or a system call. The memory references to all instructions executed in the supervisor state are automatically mapped through the kernel's page map.

Paging, the Free List, and the Loop

The process of replacing the contents of page frames with different pages is called **paging**. In addition to the page tables, two structures are central to paging: the **free list** and the **loop**. The free list contains page frames that are eligible to be reused. Page frames are added to the *head* of the free list when they are no longer needed. Conversely, page frames are added to the *tail* of the free list when they may be needed again.

Page frames not on the free list are on the loop. This is a list that contains all allocated page frames, sequenced by physical address. Note that frames containing kernel code and data are on neither the loop nor the free list since the kernel is not subject to paging.

The pager is a system process that keeps the free list large enough to maintain good performance. It runs when the free list drops below a minimum-size threshold and continues running until it has built the free list back up to a maximum-size threshold. The pager's replacement policy is to release, on a system-wide basis, page frames containing pages that have not been recently accessed.

The pager and kernel work toward keeping frequently-accessed pages associated with page frames, while little-used pages tend to migrate to the swap device. The pager also moves not-recently-accessed page frames to the free list, so they can eventually be reallocated. Only the least-used pages get to the head of the free list and therefore have to be read from disk before they are used. The free list thus serves both as a source of available page frames and as a cache of recently-discarded pages that can be reclaimed quickly.

The kernel can predict neither the memory usage nor the reference patterns of an arbitrary group of mapped, running processes. Accordingly, physical memory can become over-committed. This over-commitment is indicated when the pager cannot keep the free list above its minimum-size threshold. To forestall the possibility of **thrashing**, excessively high paging activity, the kernel initiates a measure more drastic than paging: it swaps the entire process to disk.

Swapping out processes to the swap device frees page frames. In addition, the swapped process's segments, page table, and user page are eligible for swapping out as well. More importantly, swapping reduces short-term contention for page frames and therefore reduces the CPU cycles required to execute a process. With less contention, some resident processes should run to completion, bringing the demand for physical memory back into the range that can be managed effectively with paging and the free list.

Swapping



Swapping is the job of a kernel process called the **swapper**. It attempts to select for swapping those user processes whose progress will be least impeded by losing physical memory residency. A process that has been blocked for a long time is likely to remain so (often it is waiting for keyboard input). The swapper therefore selects the process that has been blocked the longest. If no resident process is blocked, it selects the process that has been resident the longest. This is an attempt to provide some measure of 'fairness' among processes and is an example of an artifact from timesharing.

A swapped process is swapped in when it becomes ready and enough memory is available. After swapping in a process, the kernel makes sure that the process makes some minimal progress before considering it again for swapping out.

### For Further Information

For a complete discussion of *vmstat(8)* usage and memory management, see the references listed below. The two books on MMU functions detail the internal bit fields contained in the mapping and paging tables. Examples of data structures and system calls associated with MMU activities are also included.

- □ vmstat(8) manual page
- The UNIX System: A Sun Technical Report
  Sun Microsystems, Inc., 1985
  Section 2.3, Memory Management
- Sun-3 Architecture: A Sun Technical Report Revised August 1986
   Sun Microsystems, Inc., 1986
   Section 1.4, Virtual Memory Architecture



## SunOS Sys4-3.2 Usage

Sys4-3.2 Usage Considerations

This article describes three important considerations for users of Sys4-3.2, the first Sun Operating System (SunOS) release being shipped with all Sun-4/200 series Scalable Processor ARChitecture (SPARC) workstation systems.

uuxqt

The uuxqt program called and executed by uucp does not function properly. A version of Sys4-3.2 that incorporates a fix to this problem is available from Sun Customer Service.

Kernel Configuration

The installation procedures included in the documentation do not properly describe how to configure a kernel for Sun-3 clients on a heterogeneous server. To build a Sun-3 kernel, the file permissions in /usr.MC68020/sys must first be modified so that a non-privileged user can write to the file. To change permissions, mount as read/write (rw) in the /etc/fstab file. When this has been done, perform the configuration and kernel build from the Sun-3 client. Install the kernel on the server as usual, then reboot the clients.

The extract\_release Shell Script Problem

When using 1/4" tape drives on a Sun-4, the shell script extract\_release incorrectly attempts to load from /dev/rst0. extract\_release should instead load from /dev/rst8.

The Workaround

To fix the problem, edit all occurrences of \${TAPE}0 in the file /usr/etc/extract\_release as follows.

Change: \${TAPE}0

To:

\${TAPE}8

Keep in mind that this change causes extract\_release to improperly function with 1/2" tape drives.



## **Asynchronous Modems**

# Asynchronous Modems Used on Sun Workstations

The following is a compilation of some of the asynchronous modems that have been successfully used with Sun workstations for various communications purposes.

This compilation includes the following information.

- Manufacturer's name and model number
- Baud rate at which modem has been tested and verified
- □ Modem connection: CPU port, ALM, MCP, ALM-2, terminal
- Modem use: tip, uucp, dial in and out
- □ Extra features, such as security callback, MNP, and so on (if applicable)
- Significant usage considerations

Please keep in mind that this list is not intended to be a recommendation or endorsement of these products over other products that are currently available.

Information on how to properly configure modems can be found in the section "Adding Hardware to Your System" in the System Administration for the Sun Workstation, part number 800-1323-03.

**USRobotics Courier 2400** 

Baud rate: 300, 1200, 2400

Modem connection: CPU port, ALM

Modem use: tip, uucp, dial in and out

Significant usage considerations:

Available as a rackmount or a standalone modem. The telephone number for customer service is located on the bottom of the modem. USRobotics modems have a two-year warranty, which may be extended an additional three years by mailing in the registration form.

This modem is Hayes-compatible.



Ven-Tel 1200-32

Baud rate: 300, 1200

Connected to: CPU port, ALM

Modem use: tip, uucp, dial in and out

#### Significant usage considerations:

There are at least two different versions of the Ven-Tel 1200-32 available. One model has four internal switches as well as four switches on the back of the modem. None of the switches are labeled. Another version of the Ven-Tel 1200-32 has ten internal switches only.

This modem is Hayes-compatible.

Hayes Smartmodem 2400

Baud rate: 300, 1200, 2400

Connected to: CPU port

Modem use: tip, uucp, dial in and out

Extra features: Can select either CCITT or Bell protocol at 1200 baud.

## Significant usage considerations:

The Hayes Smartmodem 2400 does not have any physical configuration switches. This modem is configured using commands that are stored in non-volatile memory.

If modem was last used at 1200 baud, it will not answer at 2400 baud. This can be avoided by configuring the modem to reset when DTR drops. However, this configuration produces the message init: getty failing, sleeping.

Ven-Tel MD212PLUS

Baud rate: 300, 1200

Modem connection: CPU port

Modem use: tip, uucp, dial in and out

### Significant usage considerations:

The Ven-Tel MD212PLUS modem uses a proprietary Ven-Tel auto-dial interface. This modem comes in several forms, such as rackmount, standalone, and with or without auto-dial capability.



This modem is not Hayes-compatible.

ADC MD1202

Baud rate: 300, 1200

Connected to: Terminal

Modem use: Terminal connection

Extra features: Automatic repeat dial when busy signal is detected.

Significant usage considerations:

This modem is Hayes-compatible.

DATACOMMunications Shuttle Volksmodem 12 Baud rate: 1200

Connected to: CPU port

Modem use: tip, uucp, dial in and out, ACSnet

Significant usage considerations:

This modem is available in Australia, and is Hayes-compatible.



## Sun Modula-2 Release 2.0

## Sun Modula-2 Release 2.0 Announcement

This article is a brief overview of Release 2.0 of the Sun Modula-2 programming language for the Sun-2 and Sun-3 workstation environment, including new features and installation/usage considerations. This release of Sun Modula-2 runs under Sun Operating System (SunOS) Release 3.4 and higher.

New Features and Improvements

The following highlights new features and performance improvements incorporated into Sun Modula-2 Release 2.0.

- Procedures can now return to any data type, including ARRAY,
   RECORD, and SET.
- SET types can now consist of up to 2048 elements.
- Set constructors can now include non-constant expressions.
- CASE statements can now contain long displacements.
- □ The Sun-3 version of m2c now supports the -ffpa option.
- The m2dep demo program, in source form, which can be used to build a dependency analyzer.
- BYTE and WORD types now allow comparison. Additionally, these types may also be used to cast numeric literals.
- Constant expressions now permit the built-in functions FLOAT and TRUNC.
- There is no longer a restriction on the number of modules being linked. This is an improvement over Sun Modula-2 Release 1.0, which was limited to a maximum of 128 linked modules.
- Calls to the Sun Modula-2 trap routine may be intercepted, allowing user-defined handling of error conditions to be implemented.

Sun Modula-2 performance has been enhanced as follows:

□ The Sun Modula-2 compiler now runs up to 25% faster. The speed improvement is especially apparent on substantial modules.



- The compiler generates much faster object code, as it now uses machine registers for heavily-used variables.
- The initialization of modules is now handled by generation of appropriate subroutine calls at link time, rather than by explicit test-and-set sequences at run time.
- Static data declared at the outer level is now placed in the "bss" section, rather than the "data" section, thus reducing object file size.
- Substantial improvements have been made in the performance of the String module.
- Subroutines are now used for range and bounds checking, rather than in-line code. This can dramatically reduce the code size of programs, with a modest degradation in execution speed.
- Coroutine variables are now implemented as pointers to a fixed location, rather than continuously varying during program execution. Thus, coroutine variables can now be safely assigned to one another.
- Stack corruption checking now occurs at each call to TRANSFER, thus providing improved security.
- ☐ The FOR loop code has been streamlined to execute faster.
- Certain alignment-related malfunctions which occurred when using type coercions in Release 1.0 have been fixed.

Specific details regarding some of the above features and improvements are described individually below.

ARRAY, RECORD, or Large SET Statements

Functions returning an ARRAY, RECORD, or large SET type do so by placing the value into a location allocated by the caller of the function, and passed by reference as a hidden parameter to the function. In the case where a function's return value is assigned directly to a variable, the location of the destination variable is passed. In other cases, a temporary location is allocated by the compiler, and the result is copied as necessary.

**CASE Statements** 

CASE statements can now contain long displacements. To access this feature, include the following as arguments to the m2c command:

-Qoption mfl -J



BYTE and WORD Type Comparisons

Comparisons are now allowed with BYTE and WORD types. Additionally, they may be used to cast numeric literals. Keep in mind that this is a non-standard language extension. Using these types will result in non-portable programs.

m2dep--Sun Modula-2 Dependency Analyzer Sun Modula-2 Release 2.0 contains a set of components that may be used to build a dependency analyzer. The analyzer program, called m2dep, accepts a series of options similar to those used by the m2c command. m2dep generates a PostScript® program, which can then be sent to an Apple Laser Writer<sup>TM</sup> printer. This program shows the import/export relationships among the modules, displayed in a tabular format. All or part of the program output can be modified to suit the user's individual needs.

In addition to displaying the import/export relationships, the m2dep program can be used as an example of the following:

- Partitioning a problem into interdependent modules
- Using a Makefile
- Organizing a project using SCCS
- Using pointers to arbitrarily-long character arrays
- Using procedure variables effectively
- Using various UNIX routines, such as qsort, strcmp, and strlen
- Using various UNIX calls, such as read, write, open, and close
- Getting program arguments
- Using the PostScript language

The m2dep demo program is provided in source form. If components of the program are used, keep in mind that it is the intellectual property of the author as well as the legal property of Sun Microsystems. Also keep in mind that no support will be provided by Sun Microsystems for the m2dep program or its components.

Installation and Usage Considerations

The set of Sun Modula-2 library modules, as well as their exported identifiers and meanings, remain unchanged from Release 1.0. There is no need to modify existing Sun Modula-2 programs to use the new release. Because the library version stamps are changed, as well as the linking and initialization mechanism, existing Sun Modula-2 programs will have to be recompiled and re-linked.



If Sun Modula-2 Release 1.0 exists on a Sun system, the new release will overwrite the old release. This release occupies approximately 1.2 Mbytes of disk space, or about 92% of the disk space required by Release 1.0.

Refer to the documents Software READ THIS FIRST Sun Modula-2, Sun Modula-2 Installation Guide, and Sun Modula-2 Release Notes for further information.



## C Hunt

C Hunt: Looking for Books on the C Programming Language?

Users who are new to the Sun workstation/UNIX operating system environment often look for introductory books discussing the C programming language. The following list includes some of the publications currently available in technical bookstores. The average retail price of each (in US dollars) is also included. The authors of these books assume that you are already familiar with the function and usage of basic programming concepts, such as variables, assignment statements, and loops.

Please keep in mind that this list is not intended to be a recommendation or endorsement of these books over other publications that are currently available.

- Learning to Program in C by Thomas Plum
   Englewood Cliffs, NJ: Prentice-Hall Incorporated, 1983.
   ISBN 0-911537-00-7 \$33.00
- C Primer Plus, revised edition, by Michael Waite, Stephen Prata, and Donald Martin
   Indianapolis, IN: Howard Sams & Company, 1987.
   ISBN 0-672-22582-4 \$24.95
- Date of the C Programmer's Handbook by Thom Hogan Bowie, MD: Brady Communications, Inc., 1984. ISBN 0-89303-365-0 \$19.95
- The C Primer, second edition, by Les Hancock and Morris Kreiger New York, NY: McGraw-Hill, Inc., 1986.
   ISBN 0-07-025995-X \$19.95
- C Programming Guide by Jack J. Purdum
   Indianapolis, IN: Que Corporation, 1983 and 1985.
   ISBN 0-88022-157-7 \$19.95
- A Book on C by Al Kelley and Ira Pohl
   Menlo Park, CA: The Benjamin/Cummings Publishing Company, 1984.
   ISBN 0-8053-6860-4 \$28.50
- The C Programming Tutor by Leon A. Wortman and Thomas O. Sidebottom
   New York, NY: Prentice-Hall, Inc., 1984.
   ISBN 0-89303-364-2 \$21.95



- The C Trainer by Alan R. Feuer
   Englewood Cliffs, NJ: Prentice-Hall, Inc., 1986.
   ISBN 0-13-109752-0 \$24.95
- □ The C Companion by Allen I. Hollub Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987. ISBN 0-13-109786-5 \$22.95
- C Made Easy by Hervert Schildt
   Berkeley, CA: Osborne McGraw-Hill, 1985.
   ISBN 0-07-881178-3 \$18.95
- Introduction to C by Paul M. Chirlian
   Beaverton, OR: Matrix Publishers, 1984.
   ISBN 0-916460-37-1 \$15.95

#### Standard C Reference Books

From these introductory texts, you should become familiar enough to proceed with confidence through the standard C reference books such as, *The C Programming Language* and its companion publication, the *The C Answer Book*, and *C: A Reference Manual*, listed below, as well as other advanced texts.

- The C Programming Language by Brian Kernighan and Dennis Ritchie Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978. ISBN 0-13-110163-3 \$27.00
- The C Answer Book by Clovis C. Tondo and Scott E. Gimpell Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.
   ISBN 0-13-109877-2 \$20.95
- C: A Reference Manual, second edition, by Samuel P. Harbison and Guy L. Steele, Jr.
   Englewood Cliffs, NJ: Prentice-Hall, Inc. 1987.
   ISBN 0-13-109810-1 025 \$25.95



### NeWS Release 1.1

#### **NeWS 1.1 Announcement**

This article is a brief overview of NeWS Release 1.1, an enhanced release of Sun's Network Windowing System. NeWS Release 1.1 can be used with Sun-2, Sun-3, and Sun-4 hardware, and runs on SunOS Release 3.2 and greater (including Sys4-3.2).

New Features and Enhancements

NeWS Release 1.1 provides the following new features, changes, and enhancements.

- Support for the full PostScript® language font model, including definefont
- A new version of psview, as well as improved PostScript language previewing
- Dashed line support, using the standard setdash and currentdash
   PostScript language primitives
- Repeating keys
- Improved font library management
- Enhancements to psterm
- Support for Kanji font
- Improved communications with the server, using a new psio communications package in \$NEWSHOME/libcps.a
- A journalling package is now available to provide record/playback functions
- Reorganized NeWS root menu
- Enhanced flexibility in window and scrollbar creation
- Coexistence with SunView 1, allowing selections to be transferred between SunView 1 and NeWS
- New demos

In addition, fixes to bugs from NeWS Release 1.0 have been incorporated.



Full Font Model Support

definefont is now supported, allowing the ability to preview more kinds of PostScript language output without modifications. In addition, setcachedevice, and setcharwidth are supported. Note that charpath does not work, because all NeWS fonts are user-defined.

All fonts now use the Adobe collating sequence. This can be overridden by using definefont after changing the Encoding.

New Version of psview

A new version of psview, the PostScript language page previewer, looks for the PostScript language conventions %%EndProlog, %%Page and %%Trailer to determine where pages start. In addition, psview provides a slider to move to any page, as well as a menu to go to the first, previous, last, or next page.

Repeating Keys

The standard typing array of keys repeat at a default of 20 times per second after a .5 second threshold. When multiple keys are depressed, only the last key down is repeated. When the last key pressed is lifted, all repetition stops. Function keys and shift keys are not repeatable. The repeat function is implemented by the standalone package \$NEWSHOME/lib/NeWS/repeat.ps. This package is loaded as part of the Extended Input System started by init.ps.

Refer to "Assigning Function Keys" in the NeWS 1.1 Release Notes for further information.

Font Library Management Changes

A new font library management scheme allows all font filenames to be short. NeWS Release 1.1 is now completely insensitive to file font names. The changes are as follows.

- findfilefont, a new primitive which reads in a named family definition, and returns a unit high font that refers to it. Used by init.ps to initialize FontDictionary.
- enumeratefontdicts, which has been changed to push the font family filenames onto the stack, rather than particular font names. Called by init.ps and used to initialize FontDictionary.
- FontDictionary is now initialized and accessible in C, rather than accessible in the PostScript language only.
- findfont now looks in FontDictionary, rather than in its own private database.



New Version of psterm

The new version of psterm included with NeWS Release 1.1 has the following new features:

- □ -li # option to specify a number of lines
- □ -co # option to specify a number of columns
- $\neg xy xy$ , used with the  $\neg f$  fixed-size option to specify origin
- Rows and columns are extracted from termcap (5), and not the parent process of psterm.
- Editing characters are first determined by checking in WINDOW\_TTYPARMS, then at controlling terminal (if any), or defaulting to a standard set.
- The pattern matcher has been rewritten for better performance.
- Pseudo-ttys are initialized and handled better.
- /etc/utmp is handled properly.
- Page mode has been added.
- Automatic margin option has been added.
- A menu has been included to turn page mode and automatic margin on and off.
- A visual bell has been added.

New Communications Package for Communications with Server libcps.a in NeWS Release 1.1 incorporates a new psio communications package. The purpose of psio is to enhance portability between different environments. If PostScript or PostScriptInput global variables are referenced, the psio communications package must be used. In general, psio\_replaces the f prefix on calls such as feof and ferror. For calls such as fileno, simply prepend the psio\_prefix.

Be aware that failure to make this change will result in compile-time errors. Refer to Chapter 9, *Client Interface*, in the *NeWS Manual*, and the psio(3) manual page.



### Journalling

A new package has been added to support journalling, the ability to record and play back NeWS user input events. The file \$NEWSHOME/lib/NeWS/journal.ps implements the following three procedures:

- Begins replaying from the journalling file (default filename is /tmp/NeWS.journal)
- Starts a journalling session by opening the journalling file and logging user actions to it
- Ends a journalling session started by journalrecord, and closes the journalling file

The replay is at a very low level, so the system should be in exactly the same state at the beginning of the replay as it was at the start of the journalling session, including the same windows in the same screen positions, the same user running the system from the same directory, and so on. **journalplay** repositions the mouse automatically.

The journalling functions are accessed using a new pull-right menu that has been added to the root menu. From this menu, a user can start recording user input events, stop recording, play back the events, and remove journalling. A control panel is available with buttons that can be used to control the speed of playback, auto-repeat, select the journalling file to use, and so on. Refer to the journalling (1) manual page for further information.

#### Coexistence with SunView 1

SunView 1 binaries can be run while running NeWS Release 1.1, but the selection\_svc program must be running for SunView 1 programs to be able to use SunView 1's Selection Service to cut and paste between its windows. \$NEWSHOME/bin/ensure\_sel\_svc is a small program which looks to see if a SunView 1 Selection Service is available. If not, it starts one. The demo menu code (in \$NEWSHOME/lib/NeWS/demomenu.ps) calls ensure\_sel\_svc before running any of the SunView 1 applications that need the Selection Service. If the Selection Service is not available, selection\_svc is started. If SunView 1 programs are started up by the user, ensure\_sel\_svc can be used in a similar manner.

To copy selections between NeWS and SunView 1, the utility shell scripts news2sv\_put and sv2news\_put are used. These shell scripts are available from the SunView 1 Selection Transfer menu as 'NeWS to SunView 1 Shelf' and 'SunView 1 to NeWS Shelf'.



Both use the program news\_selection to get the NeWS selection and set the NeWS shelf. news2sv\_put uses the program set\_selection to set the SunView 1 shelf. sv2news\_put uses the standard get\_selection(1) utility to get the SunView 1 selection.

Installation and Usage Considerations

The following briefly describes some considerations to keep in mind when installing and using NeWS Release 1.1.

NeWS Release 1.1 Distribution Tape

To install NeWS Release 1.1, you must have 16MB of disk space free. Please note that this is 1MB greater than specified in the NeWS installation guide.

SunView 1 Binary Compatibility SunView 1 programs that have been designed to run from a SunView 1 terminal emulator window, such as bouncedemo in a shelltool, or jumpdemo in a Lgfxtool, do not work from NeWS terminal emulator windows, such as nterm or psterm. Running these SunView 1 programs will cause the workstation to appear to lock up and not accept further input.

NeWS Release 1.1 on Sun-4 Machines with Graphics Processors If NeWS Release 1.1 is to be used on a Sun-4 with a GP graphics accelerator running Sun Operating System (SunOS) Sys4-3.2, the FRAMEBUFFER environment variable must be explicitly set to /dev/cgtwo0 before starting the NeWS server. This is done from a C shell by entering the following:

% setenv FRAMEBUFFER /dev/cgtwo0

If this is not set, NeWS will not run, and will cite problems with /dev/fb.

The above only applies to Sun-4 machines with GP graphics accelerators. Thus, if NeWS is to be used on a Sun-3 with a GP graphics accellerator, the FRAMEBUFFER environment variable should not be set to /dev/cgtwo0 before starting the NeWS server.

case Semantics Corrected

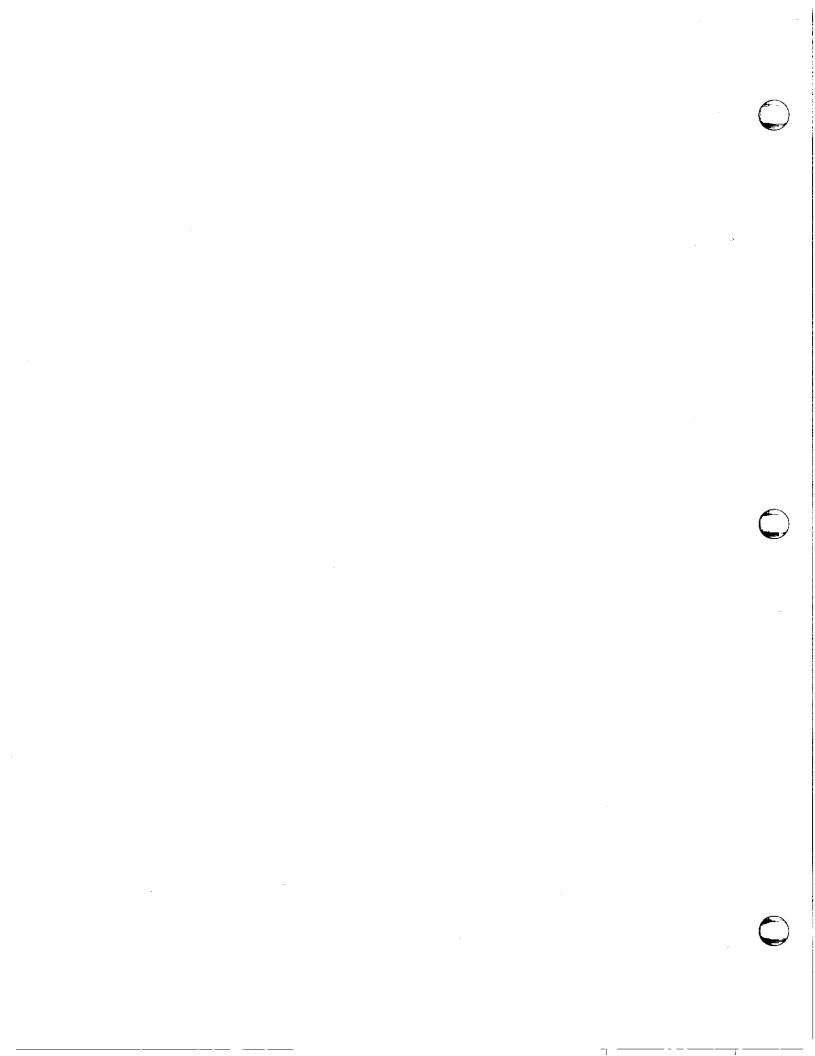
In NeWS Release 1.0, the case operator failed if the key was not matched, or if there was a match but no following executable array. In Release 1.1, case consumes its key in all situations, and uses /Default as the only default key.

Note that this bug fix will break programs using /Default {pop pop} to work around the bug.

nterm

The nterm source in \$NEWSHOME/clientsrc/client/nterm will not build under System V.





## STB SHORT SUBJECTS

STB SHORT SUBJECTS	643
SunOS 4.0 and Lisp 2.1	643
PCNFS.SYS Patch	644
PC-NFS 2.0 Installation	646
PC-NFS backup	647
Mixing Ethernets	648

•

## STB SHORT SUBJECTS

SunOS 4.0 and Lisp 2.1

Sun Common Lisp 2.1 Compatibility and SunOS Release 4.0 Sun Common Lisp 2.1 product will *not* be binary-compatible with SunOS release 4.0 on either Sun-3 or Sun-4 workstations.

Sun Common Lisp 2.1 operates normally on both Sun-3 workstations running SunOS releases 3.x, and Sun-4 workstations running Sys4-3.2.

PCNFS.SYS Patch

PCNFS.SYS Patch

The SunWindow bug caused by Microsoft can be fixed by applying the PCNFS.SYS patch discussed in this article. This patch allows correct SunWindow and PC-NFS operations, including printing.

Work Around

In applying the patch, change DEVICENUM in PDD. ASM from '16H' to '10H'.

Change the DEVICENUM in PDD. ASM as shown in the procedures below. Note the following conventions shown below.

represents a system prompt
 represents a debugging prompt

(bold) represents your input (italic) represents user notes

#### □ For PC-NFS Release 1.0

C> cd /nfs

C> copy pcnfs.sys pcnfs.sav

C> debug pcnfs.sys

- e 2A8

XXXX:02A8 16.10

(i.e. change the original value from 16H to 10H)

- M

Writing EB91 bytes

- q

C> comp pcnfs.sys pcnfs.sav

C:PCNFS.SYS and C:PCNFS.SAV

Compare error at OFFSET 1A8

File 1 = 10

File 2 = 16

Eof mark not found

Compare more files (Y/N)? n

C> reboot



#### For PC-NFS Release 2.0:

- C> cd /nfs
- C> copy pcnfs.sys pcnfs.sav
- C> debug pcnfs.sys
- e 297

XXXX:0297 16.10

(i.e. change the original value from 16H to 10H)

- w

Writing FCDB bytes

- q

C> comp pcnfs.sys pcnfs.sav

C:PCNFS.SYS and C:PCNFS.SAV

Compare error at OFFSET 197

File 1 = 10

File 2 = 16

Eof mark not found

Compare more files (Y/N)? n

C> reboot

Results

Now if you run SunWindows, you will see icons for drives up to S:. Printing, including SunWindows spooler printing, will work correctly.

Exceptions

The only feature that will not work is the 'trick' of listing the directory of T:, U:, or V: to check the print queue.



### **PC-NFS 2.0 Installation**

### **PC-NFS 2.0 Installation Aids**

The Enhanced Graphics Adapters (EGA) that support extended graphics modes beyond the normal EGA resolution of 640x350, specifically modes 640x480 and 752x410, may cause the PC to fail to display text during PC-NFS release 2.0 installation.

### **Problem Adapters**

The adapters that have caused this problem are the VEGA Deluxe from Video Seven, and the Micro Enhancer Deluxe from Everex Systems.

#### Work Around

To avoid this problem, you can set your graphics card to boot in standard 640x350 EGA mode. This can be done by configuring the adapters using the DIP switch located on the adapter's fastener bracket.

Set the adapter DIP switch as shown below.

#### □ For the VEGA Deluxe

switch 1 on switch 2 on switch 3 on switch 4 off

#### For the Micro Enhancer Deluxe

switch 1 off
switch 2 on
switch 3 on
switch 4 off
switch 5 off
switch 6 factory test
switch 7 on
switch 8 on

Your adapter is now in standard EGA mode. PC-NFS will perform the installation successfully.



PC-NFS backup

PC-NFS backup

PC-NFS is an interface between MS-DOS, UNIX, and other operating systems that allows PCs to benefit from the network's mass-storage resources across a network. One of PC-NFS's standard features is backing up the local PC hard disk to network mass-storage systems. This feature allows you to take advantage of the larger mass-storage capacity available on Sun workstations, adding extra security and reliability to your PC files.

MS-DOS Release 3.3 backup

An attempt to perform a backup on MS-DOS release 3.3 running PC-NFS release 2.0 will fail.

Workaround

To backup a file on MS-DOS release 3.3 running PC-NFS release 2.0, boot the PC with either MS-DOS release 3.1 or 3.2 and issue the backup command.



### **Mixing Ethernets**

## Mixing Thick and Thin Ethernet

Sun users periodically need to employ a combination of thick and thin Ethernet cable in their workstation systems. For example, it may be desirable to connect several Sun-3/60s on thin Ethernet cable to a Sun-3/280 server which does not have thin Ethernet connection capability. When thick and thin Ethernet cables are mixed in a network, the converter used often performs the thick-size to thin-size conversion mechanically, using a metal barrel connector. No impedance matching or adjusting for electrical properties is performed. This is functional if only one converter exists in a circuit, but if two or more are used, serious signal loss will occur.

The following offers two solutions to the above problem.

## The Cabletron MR-9000C Multiport Repeater

One solution to this problem is the MR-9000C Multiport Repeater, offered by Cabletron. The MR-9000C is a box with a normal AUI tranceiver port on one end, and eight thin Ethernet BNC connectors on the other end. The MR-9000C connects to a backbone through the AUI tranceiver port. This device provides a normal repeater-type function between the eight thin Ethernet cables and the backbone. Additionally, it provides segment isolation when excessive collisions are detected on one of the segments. Thus, if a bad node or length of cable causes a problem, the entire network will not come down.

## The Cabletron ST-500 Ethernet Tranceiver

Another solution to this problem, also offered by Cabletron, is the ST-500 Ethernet Tranceiver. The ST-500 is compatible with Ethernet versions 1.0 and 2.0, as well as 802.3 networks. The ST-500 offers three types of coaxial cable connections, as follows:

- Non-intrusive tap (similar to a vampire tap)
- Intrusive N-Series tap (an in-line tap between two N-Series connectors)
- BNC tap (an in-line thin Ethernet tap)

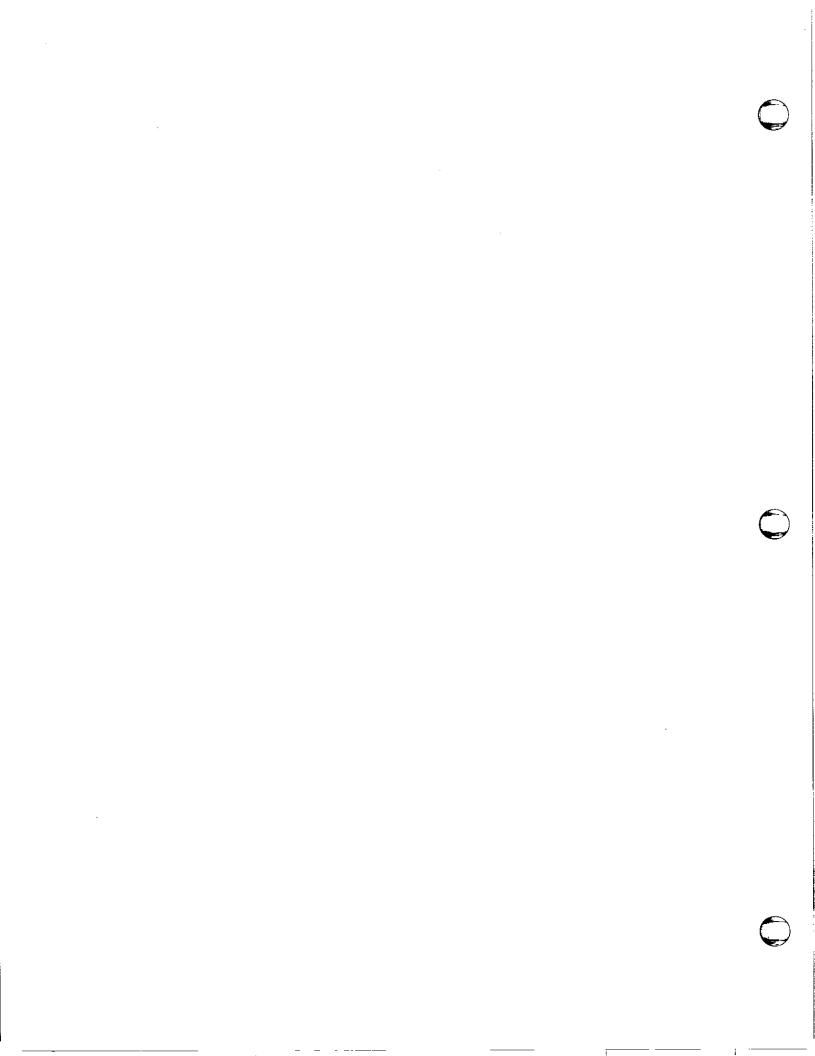
To use the Cabletron ST-500, a normal transceiver drop cable is first connected to the server, then plugged into the Cabletron tranceiver. The in-line thin Ethernet BNC connector is then used to connect to the thin Ethernet backbone, instead of a thick Ethernet connector or vampire tap. Thus, the thick-to-thin conversion is correct from both an electrical and a mechanical standpoint.



The Cabletron transceiver can be converted to use thick Ethernet cable at a later date, using a thick Ethernet connector module available from Cabletron. The thin BNC connector on the tranceiver is simply removed, and the thick Ethernet connector module is installed in its place.

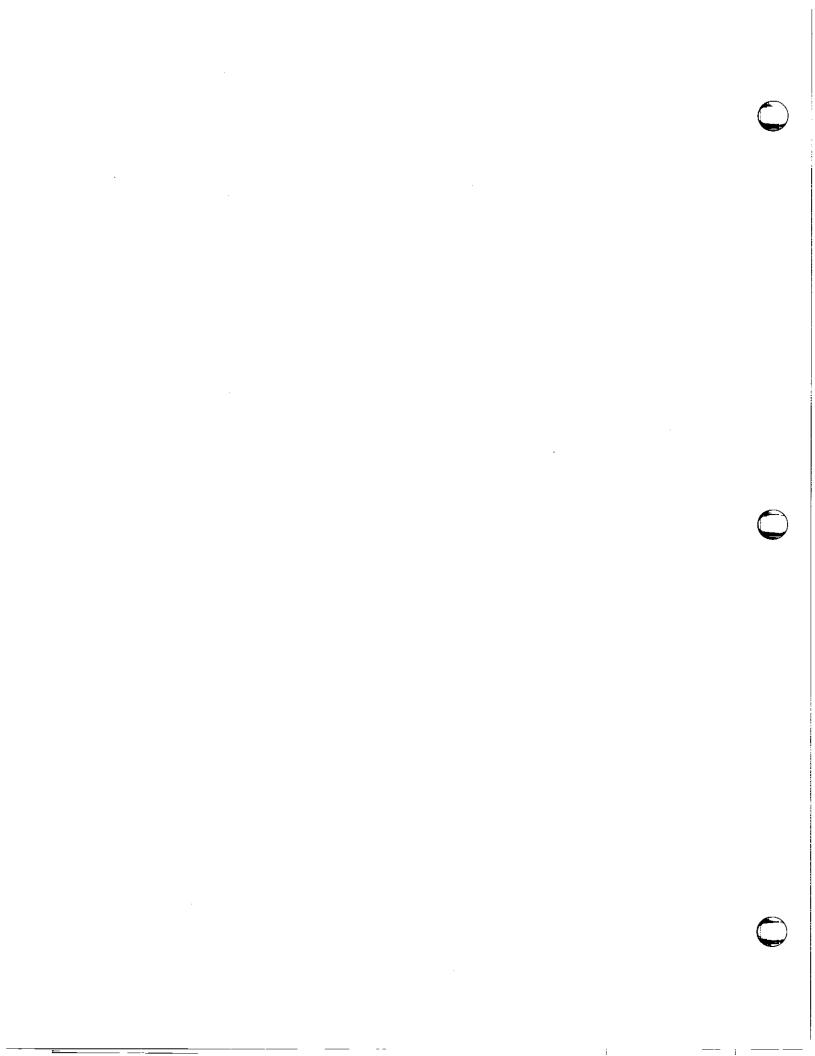
Please note that additional tranceiver products may be available from other suppliers; therefore, the above information is not intended to be a recommendation or endorsement over other products that are currently available.





# IN DEPTH

IN DEPTH	653
	000
SunOS 4.0 Overview	653
SunOS 4.0 Diagrams	666



## IN DEPTH

SunOS 4.0 Overview

SunOS 4.0 Release Report

An Overview of SunOS Release 4.0 Contents This in-depth feature describes the upcoming release of the Sun operating system, SunOS Release 4.0. The material contained in this feature is effective at the beginning of the beta-test phase of the product. Individual features and characteristics may change before the final release. Contact your Sun sales representative with questions regarding availability and requests for more information on any of the topics covered in this feature.

The major enhancements available in SunOS release 4.0 include the following features.

- Expanded, converged UNIX functions
- System administration improvements
- A foundation for hardware architectures planned for widespread use into the 1990s

Sun Microsystems has significantly redesigned the SunOS kernel, incorporating new technology and establishing a stable architectural platform for use during the 1990s. SunOS release 4.0 includes Virtual Memory (VM) management, shared libraries, and an improved OS security system.

SunOS release 4.0 is an example of the open-system approach that Sun brings to software products. Major emphasis remains on the themes listed below.

- Converging System V and BSD features into a single UNIX standard
- Supporting consistent computing across many heterogeneous environments



Platform Support

SunOS release 4.0 supports all Sun product lines, from diskless clients to servers, in the Sun-2, Sun-3, and Sun-4 series.

**New Architectural Features** 

New architectural features include new kernel architecture, a shared library facility, resizable swap areas for diskless clients, a lightweight process library, paged shared memory, and monitoring of Ethernet traffic.

#### □ New Kernel Architecture

Sun has restructured the kernel to accommodate a new virtual memory management schema that promotes system resource sharing and portability across different hardware platforms. Swap space requirements are reduced; system resource usage and caching of frequently-accessed data are more efficient. Files are treated as part of virtual memory, making access to large files more efficient.

The new VM management system accommodates page-by-page sharing, and employs a copy-on-write mechanism to create individual copies of pages when needed.

#### □ Shared Library Facility

Library-sharing reduces program size and swap space requirements, automatically incorporating and distributing newly-revised libraries throughout the system. The C library, libc, and user-built libraries can be shared.

Shared libraries use the new VM system and a revised link editor. The C compiler and assembler are enhanced to generate Position-Independent Code (PIC), used to build shared libraries.

Note that shared library usage is the default operation. However, a slight performance decrease may result from runtime linking.

#### Resizable Swap Areas for Diskless Clients

Resizing client swap space no longer requires taking a server and its clients offline or reinstalling the OS. Only clients whose swap spaces are being modified need to be halted. The resizing process is transparent to other clients. Standalone and server swap areas may also be increased online.

#### □ Lightweight Process Library

The lightweight process library is a user-level facility for managing multiple threads of control within a UNIX process. Its facilities include messages, monitors, exception handling, and flexible context switching. Both coroutine and preemptive scheduling may be used. Lightweight processes are available at the user application level only; they are not supported within the kernel.



#### Other Kernel Enhancements

Other enhancements include a System V shared memory that is now paged. Also, Network Interface Tap (NIT) protocol improvements include packet filtering, and integration with the STREAMS environment. Packet filtering provides an etherfind-like capability with which applications monitor Ethernet traffic. Ethernet packets from a particular host or with a specific field setting are two examples.

SunOS release 4.0 networking features include NFS servers, filesystem reorganization, kernel networking enhancements, automounting of remote filesystems, secure networking, and NFS performance enhancements.

#### NFS Servers

SunOS release 4.0 provides support for diskless client systems through the Network File System (NFS) rather than through the Network Disk (ND) mechanism. Clients no longer need their own partition on the disk since the swap file is now treated like any other file. OS installation and system administration of diskless clients and fileservers is streamlined.

The filesystem reorganization facilitates networking diskless clients using different CPU architectures. Support for heterogeneous environments is improved by offering implementations of the NFS server on non-Sun servers. Diskless Sun workstations will boot and swap from a non-Sun server.

The NFS server performance goal is to achieve SunOS release 3.2 ND performance levels. Eliminating ND does not affect the numbers of clients a server can support. Also, no PROM change is required for Sun-2 systems that do not support tftp booting since a user-level boot block server is provided.

#### □ Filesystem Reorganization

The SunOS release 4.0 filesystem is reorganized to simplify diskless client administration. Filesystem changes are transparent to most users. This reorganization separates host-dependent, nonshared files from architecture-dependent, shared files. This new filesystem layout is most useful for clients using different architectures working with a single server.

All files and directories that effectively define a machine's identity or are dedicated to that machine, such as config, are now in the directory /etc. These include the spool directories from /usr/spool and the adm files from /usr/adm which are now located in /etc/spool and /etc/adm, respectively.

The architecture-dependent files, including all executable files and libraries, have been moved to the /usr filesystem. The directory





/usr must now be mounted when booting into single-user mode. The filesystem /bin no longer exists since its contents are now located in /usr/bin. Similarly, the contents of /lib are now located in /usr/lib. The new layout of /usr is designed to be mounted read-only since it only contains executables.

Few executable files remain in the root directory. These include vmunix, init, sh, ifconfig, and mount which are needed for the initial machine boot. All executables in the root filesystem except vmunix are now located in the new directory /single.

#### Kernel Networking Enhancements

The native networking product set in SunOS release 4.0 is changed. Most changes are from 4.3BSD and should be transparent to applications. Important changes are listed below.

- 1. Improved Transmission Control Protocol (TCP) performance on wide-area networks
- 2. Full Internet Control Message Protocol (ICMP) support
- 3. Full Internet Protocol (IP) subnets
- 4. The Network Interface Tap (NIT) interface now uses the System V STREAMS mechanism instead of the sockets mechanism

#### □ Automounting of Remote Filesystems

The optional automount facility automatically mounts remote filesystems transparently. The automount command invokes a background daemon that intercepts directory references and mounts accessible remote filesystems when needed. Automatic unmounting occurs after a specified period of inactivity. Remote filesystem mounting uses Yellow Pages (YP) maps and local map files.

#### □ Secure Networking

Improvements in network security provide a more thorough authentication of user identification prior to allowing file access. Security measures include exchanging encryption keys and preventing superusers from using false userids to access otherwise secure filesystems across the network.

NFS uses the secure Remote Procedure Call (RPC) for optional server security. System users select secure operations by specifying the -secure option for individual /etc/exports entries. For secure operation, RPC uses a YP database of public and private encryption keys.



The secure RPC encryption mechanism is based upon the Data Encryption Standard (DES) algorithm. Note that this DES algorithm is not yet approved for export by the National Security Agency (NSA). Sun is therefore not licensed to *export* the secure networking features at this time. The SunOS release 4.0 export version does not contain the secure networking features. All other NFS and RPC features are unaffected, however.

#### NFS Performance Enhancements

The improved caching of the new VM system increases NFS performance.

SunOS release 4.0 standards reflect further progress in converging System V and BSD. Full System V Interface Definition (SVID) Release 3 Base System is supported. Mandatory record and file locking is not supported. New features are listed below.

- 1. All Base System calls are supported including chown, creat, fcntl, kill, mknod, open, and utime.
- 2. Complete System V STREAMS interface. STREAMS supports portable communication protocol modules and simplify writing device drivers.
- 3. Fully System V- and BSD-compatible tty interface using STREAMS. The tty driver supports all character sizes and parity settings.
- 4. System V-compatible archive utility: ar.
- 5. System V batch utility and job scheduler: at(1), batch(1), cron(1), and crontab(5).
- 6. Access to Sun value-added libraries from System V programs, e.g., SunView.

#### 4.3BSD Functions

SunOS release 4.0 provides most 4.3BSD functions, including the new upper limit of 64 open files per process. The existing limit is 30 open files. A fully compatible 4.3BSD subnet facility is also included. Generally, SunOS release 4.0 contains most of the 4.3BSD bug fixes and performance enhancements applicable to Sun systems.

Changes in SunView include a new text window menu, additional keyboard control in text windows, other text enhancements, alerts, a new mailtool, and display support for 8-bit characters.

Standards

SunView



#### New Text Window Menu

The SunView text menu has been reorganized and expanded, with industry-standard names replacing previous ones. For example, *Put*, *Get*, and *Delete* are now Copy, Paste, and Cut, respectively. All basic editing functions can be performed from the menu, with function keys, and with Meta-key sequences.

#### Keyboard Control in Text Windows

Keyboard keys can control caret movement within a text window. In general, <u>Control</u>-key sequences move the caret, and <u>Meta</u>-key sequences invoke menu commands such as editing, finding, and the like.

The Shift key generally acts to reverse the direction of these other key sequences.

#### More Text Window Enhancements

The 'Word Wrap' option automatically splits lines at word boundaries when they become too long for the window. Pressing Return starts a new paragraph. These automatic line splits have no impact on the way the file is actually saved. In the saved fine, the text up to Return is stored as one continuous line, regardless of screen appearance.

A Find and Replace pop-up frame can be invoked from the text menu. The user can search for a string and replace it with another, and can replace the current string, next string, or all occurrences of a designated string.

The user can select a delimiter such as "{" and choose 'Match Delimiter' from the text menu to extend the selection to the matching delimiter ("}" in this case). Text fields use a special pair of delimiters, shown below. By pressing Control-Tab, the caret jumps to the next field, and any typing replaces the selected field.

|>a field<|

#### □ Alerts

The Alerts package replaces a previous utility that displayed boxed error messages. Error messages, warnings, and queries now appear in pop-up alert windows. By pushing the appropriate button, the user can Continue after an error message, can choose to Confirm, discard edits or Cancel when quitting a document, and so forth.



#### The New mailtool

The new version of mailtool includes the features listed below.

- 1. A more compact control panel with fewer buttons.
- 2. A hierarchical Folders menu.
- Multiple pop-up Reply and Compose mail frames.

In order to provide downward compatibility and offer users both the old and new versions of mailtool, many of the new features are not visible until the user selects the new version, makes the appropriate modifications to mail settings in the *Mail* category of defaultsedit, and then restarts mailtool.

8-Bit Display Support for Data in Files

Text and TTY windows can display 8-bit characters.

For visual compatibility with previous versions of SunView, most of the above SunView enhancements can be "hidden" by setting preferences in the new *Compatibility* category in defaultsedit.

Changes in SunOS release 4.0 peripheral device installation includes the suninstall utility, online disk formatting, new mass storage systems, and removal of the Interphase 2180 driver.

The suninstall Utility

The new suninstall utility replaces the existing setup utility and improves system installation by making it more convenient and flexible. With the new installation utility, users can reuse configuration files for common configurations, edit existing configuration files to correct errors or to support minor configuration variations, and specify variants to Sun-supplied configuration alternatives.

The suninstall interface is tabular, much like the SunOS release 3.x terminal version of setup, and does not require bit-mapped screens.

The Online Disk Formatting Utility

The new disk formatting utility allows online formatting of disk drives. Formatting disks is much faster and no longer requires a dedicated system. Multiple disks can be formatted in parallel.





#### New Mass Storage Systems

SunOS release 4.0 supports current and upcoming mass storage systems, reflecting Sun's ongoing development of larger storage systems.

#### Removal of Interphase 2180 Driver

The removal of obsolete drivers is part of the ongoing program to unburden the system software from supporting an ever-growing number of devices. Systems with Interphase 2180 controllers include a few Sun-100Us and Sun-150Us which must be upgraded before they can run SunOS release 4.0.

#### Other Enhancements

Additional SunOS release 4.0 enhancements include optional secure system measures, compiler enhancements, internationalization of character sets, improved documentation, and an obsolescence mechanism.

#### □ Secure System Work

Optional security measures for tracking attempted security breaches invoke audit logs of failed attempts and successful attempts or both to use system resources, on a system-wide or individual-user basis. SunOS release 4.0 improves protection of the password database. Booting in single-user mode can be set up to require the root password for stronger system security.

SunOS release 4.0 meets C2-level functions for DoD uses, as specified in the National Computer Security Center (NSCS) *Trusted Computer System Evaluation Criteria* (Orange Book). SunOS 4.0 also provides the groundwork for future secure system products.

#### Compiler Enhancements

SunOS release 4.0 compiler technology includes Sun-4 code generators that take advantage of Sun's Reduced Instruction Set Computer (RISC) architecture, Sun's Scalable Processor ARChitecture (SPARC), and the Sun-4 processing power.

Extensive work has been done with code optimization, particularly for the C compiler. The Sun-4 global and peephole optimizers for C increase performance. Some compute-bound applications that take an hour to run unoptimized, run in about 17 minutes after recompiling with optimization.

Applications optimized for SunOS release 4.0 increase performance on Sun workstations by 20-25% over previous SunOS releases.



#### Internationalization of Character Sets

SunOS release 4.0 supports 8-bit non-ASCII characters, a step toward an international system that supports local character sets. The terminal driver now processes the input and output of 8-bit characters, both to and from terminals. The filesystem processes filenames containing 8-bit characters. The Bourne shell processes commands and their arguments containing 8-bit characters.

Neither the C shell nor the text editors ed and vi support 8-bit characters in this release. Note that textedit, shelltool, and cmdtool in SunView support 8-bit character display; however, they do not provide any mechanism for typing those characters into the system.

#### Improved Documentation

New, revised manuals include the System Services Overview, Security Features Guide, and a Global Index to all system manuals. Documents with major revisions include Installing UNIX and Programming Utilities.

Documentation repackaging offers more convenience when selecting specific manual sets, e.g. introduction to system use, system reference, system administration, and program development.

#### Obsolescence Mechanism

The directory /usr/old contains obsolete modules. Modules placed in this directory are subject to removal at major SunOS 4.x releases. This obsolescence mechanism notifies users about planned removals. The programs moved to /usr/old are listed below.

filemerge	Enhanced version provided with the Network Software Environment (NSE) product
sun3cvt	Needed only for transition to SunOS release 3.0
compact	Replaced by faster and more efficient, but incompatible compress program from 4.3BSD
eyacc	Used only to implement Pascal, removed in 4.3BSD
make	Pre-SunOS release 3.4 version of make is replaced
prmail	Replaced with mail -u in 4.3BSD
pti	Replaced with troff -a in 4.3BSD



## SunOS Release 4.0 Specifications

SunOS release 4.0 memory allocation, disk space requirements, and estimated performance for the various configurations will be fully characterized and documented by the formal release date.

System Resource Requirements and Performance Characteristics In the meantime, users can expect the following changes.

- Improved memory usage due to shared libraries and more efficient buffer caching
- Sun-3 and Sun-4 performance at approximately SunOS 3.x levels, even with added functions. Programs compiled with pre-SunOS 4.0 releases may not perform as well when run with SunOS release 4.0, since they lack the new enhancements such as shared libraries. Recompiling with SunOS 4.0 will correct problems of this nature.
- Simplified system administration for system installation and for managing diskless clients
- Installing the entire SunOS release 4.0 requires more disk space. System installers should remove unnecessary modules from their systems to optimize disk usage.
- The SunOS release 4.0 kernel must be reconfigured to realize SunOS 3.x performance levels. A reconfigured kernel optimizes memory usage compared to the GENERIC kernel. Detailed documentation and several typical configuration files are provided.
- Binary code compatibility with SunOS 3.2 and later releases in most cases. Source code changes may be required for modules using changed or obsoleted library and system calls. Consult the Change Notes for details.

## Compatibility Issues and Procedural Changes

Key compatibility issues and areas subject to visible impact on administrative or program-building procedures include those described below.

New Architectural Features

New SunOS release 4.0 architectural feature compatibility issues and changes are described below.

□ New Kernel Architecture

Programs that depend on the format of kernel data structures may require revisions.

□ Shared Library Facility

Existing programs will require rebuilding to benefit from shared libraries. Programs requiring a non-shared version of a library must explicitly specify this in their build procedures.



#### □ NIT Improvements

Programs that use NIT will require source changes and recompilation.

Networking

Networking changes required for running SunOS release 4.0 are summarized below.

#### Improved Support for Diskless Clients

Administrative procedures are easier; adding new clients has no affect on active diskless clients.

#### □ Filesystem Reorganization

Simplified procedures; applications referencing relocated files without symbolic links should be updated.

#### □ Kernel Networking Extensions

The ifnet structure offers more generality. For example, a single interface can be used by different address families. All SunLink products are affected and new releases supporting SunOS release 4.0 will be issued.

The kernel uses a new mbuf convention to process sockets. Some customer network drivers may require modification.

The interpretation of TCP-urgent data is closer to the official specification. Note that rlogin from SunOS releases 3.3 or earlier may not negotiate terminal modes correctly with SunOS release 4.0.

#### Secure Networking

Secure authentication prohibits access to setuid programs that lack authorized access.

Compatibility issues and procedural changes for standards include System V enhancements, System V shared memory, SunView, peripherals, and other enhancements.

#### System V Enhancements

**SVID Compliance.** Programs built in the System V environment with fcntl, open, and utime will require recompilation if fully SVID-compliant behavior is desired.

System V/BSD tty. Programs using System V termio ioctl calls will require recompilation if fully SVID-compliant behavior is desired.





Batch Utility/Job Scheduler. Slightly affects system administration procedures.

#### System V Shared Memory

Programs using the System V shared memory shmdt library call that were linked prior to SunOS release 3.4 should be recompiled to avoid a spurious message to the console or window.

#### SunView

shift\_mask. SunView in SunOS 4.0 is virtually source code compatible with SunView in any SunOS release 3.x. One known exception is in programs testing the shift\_mask. This exception applies to programs that test the entire shift\_mask rather than individual bits within the shift\_mask.

defaultsedit. A new defaultsedit category, *Compatibility*, overrides the new features in SunOS release 4.0 and restores the 'look and feel' of SunView as it was when running SunOS releases 3.x.

#### □ Peripherals

suninstall Utility. This utility now includes more convenient installation procedures.

Online Disk Formatting Utility. The utility has faster and easier administrative procedures.

Removal of Interphase 2180 Driver. Systems using the Interphase 2180 must be upgraded.

#### Other Enhancements

Secure System Work. Modules making direct use of the encrypted password field in /etc/passwd or /etc/group will require modification.

#### SunCGI and SunCore

With SunOS release 4.0, current plans call for SunCore and SunCGI product development to be frozen. New Sun platforms will not be supported by SunCGI and SunCore. The products are still supported on existing systems, but no new functions will be added. Reported problems will still be resolved in accordance with Sun's normal support procedures. Sun is sensitive to the issue of obsoleting software and is designing migration strategies to provide smooth transitions to new graphics technologies.

## Sun FORTRAN and Sun Pascal

Sun FORTRAN 1.1 and Sun Pascal 1.1 are separate, value-added products that will be shipped concurrently with SunOS release 4.0. Due to recent legal developments, £77 and pc, the UNIX FORTRAN and Pascal compilers, will be removed from SunOS release 4.0 tapes. Customers with support contracts



will receive SunOS 4.0 versions of Sun FORTRAN and Sun Pascal free of charge. Other customers may purchase these as seperate products.

Sun FORTRAN has many new features including VMS extensions. Sun Pascal is ISO Pascal with separate compilation, variable-length strings, and global optimization.

filemerge

filemerge has been replaced by fileresolve, a new, enhanced version that is provided with the NSE and is critical for large development projects that take advantage of parallel development. fileresolve contains many new features including an improved user interface. Through tight integration with NSE, fileresolve tracks the version history of files and can automatically retrieve ancestor versions. Its automerge capability facilitates three-way merges, automatically resolving nonconflicting changes between successive versions of files.

SunOS Release 4.0 Installation

SunOS release 4.0 is a major release introducing extensive architectural changes and requires a full system installation. However, with the suninstall utility, system installation and configuration are more user-friendly and offer more flexibilty.

Availability and Distribution

SunOS release 4.0 will start shipping in the second quarter of calendar year 1988.

Customers with Sun software support contracts will receive SunOS release 4.0 as part of their support program. In the U.S., those customers should call the United States AnswerCenter (USAC) at 800-USA-4SUN for installation assistance. Customers holding support contracts and who are outside the U.S. should call their local support group. See the note World Hotlines appearing in the STB Notes and Comments Section I for software customer service numbers worldwide. Contact your sales representatives for more information about Sun's software support services.

Workstations that are not covered under a Software Support agreement with Sun need individual licenses for each SunOS release 4.0 upgrade. Upgrade licenses, media, and documentation will be available through your Sun sales office when SunOS release 4.0 begins shipping.



**Diagrams** 

#### **SunOS 4.0 Diagrams**

# SunOS 4.0 Compatibility This

This article provides diagrams comparing SunOS release 4.0 with established systems and standards. Diagrams appearing on the following pages are listed below.

- □ Figure 1: SunOS 4.0 and SunOS 3.2 Functional Comparison
- Figure 2: SunOS 4.0 and SVID OS Service and General Library Routines
- Figure 3: SunOS 4.0 and SVID OS Service Routines and Utilities Extensions
- □ Figure 4: SunOS 4.0 and SVID Systems and Software Development Routines
- □ Figure 5: SunOS 4.0 and SVID Terminal and Network Interface Routines
- □ Figure 6: SunOS 4.0 and SVID Networking, Header, and Shared Resource Utilities
- □ Figure 7: SunOS 4.0 and 4.3BSD System Calls
- □ Figure 8: SunOS 4.0 and 4.3BSD Library Routines
- □ Figure 9: SunOS 4.0 and 4.3BSD Commands



#### SunOS 4.0

## NEW ARCHITECTURAL FEATURES

State-of-the-art VM system Shared library facility Lightweight process library

#### **NEW NETWORK SERVICES**

SunOS 3.2

or later

NFS support for swapping and paging diskless clients Resizable swap area for diskless clients

Filesystem reorganization for efficient support of multiple architectures

Full Sun-2, Sun-3, and Sun-4 client and server support Automounting of remote files

#### NEW SYSTEM V FACILITIES

Full base level SVID Release 3 interface STREAMS interface to kernel STREAMS-based System V tty driver Archive utility and job scheduler

#### **NEW SUNVIEW FEATURES**

Industry-standard function names Keyboard control of menus Menus provide all basic editing functions Enhanced mailtool

#### NEW PERIPHERAL SUPPORT

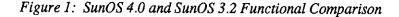
Reusable and editable configuration files On-line disk formatting New mass-storage systems

#### SECURITY MEASURES

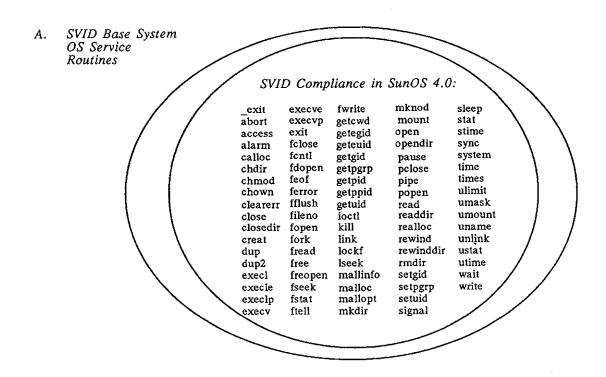
RPC authentication Secure NFS Audit trails

#### **OTHER FEATURES**

Support for 8-bit characters C global and peephole optimizers Up to 64 open files per process Obsolescence mechanism Dedicated documentation sets







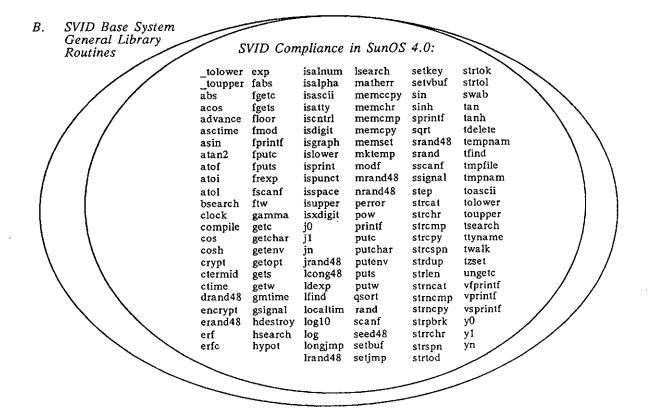


Figure 2: SunOS 4.0 and SVID OS Service and General Library Routines



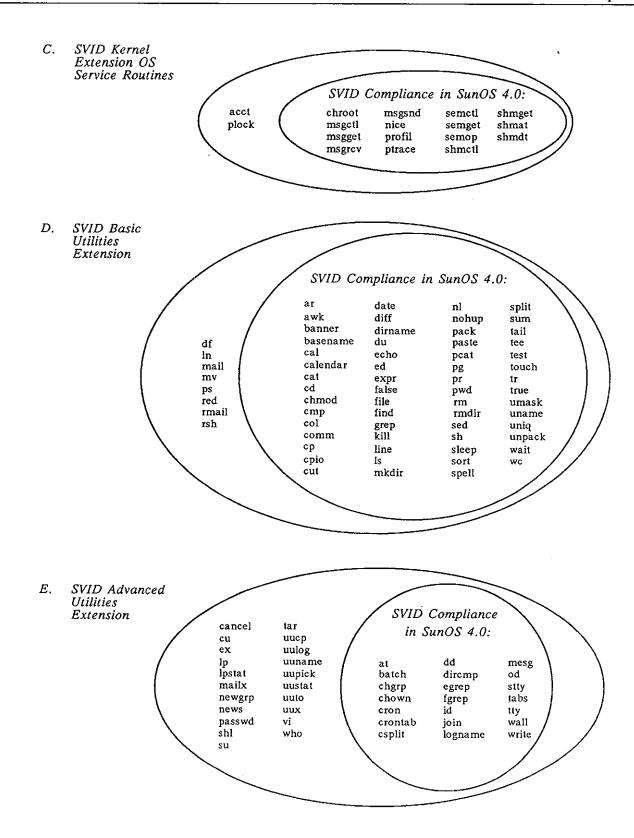


Figure 3: SunOS 4.0 and SVID OS Service Routines and Utilities Extensions



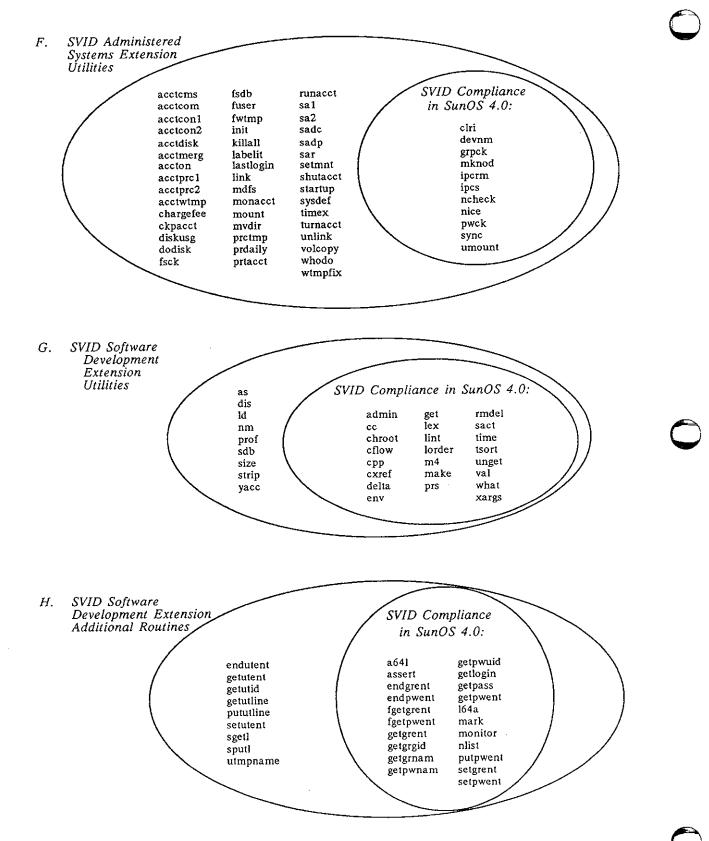


Figure 4: SunOS 4.0 and SVID Systems and Software Development Routines



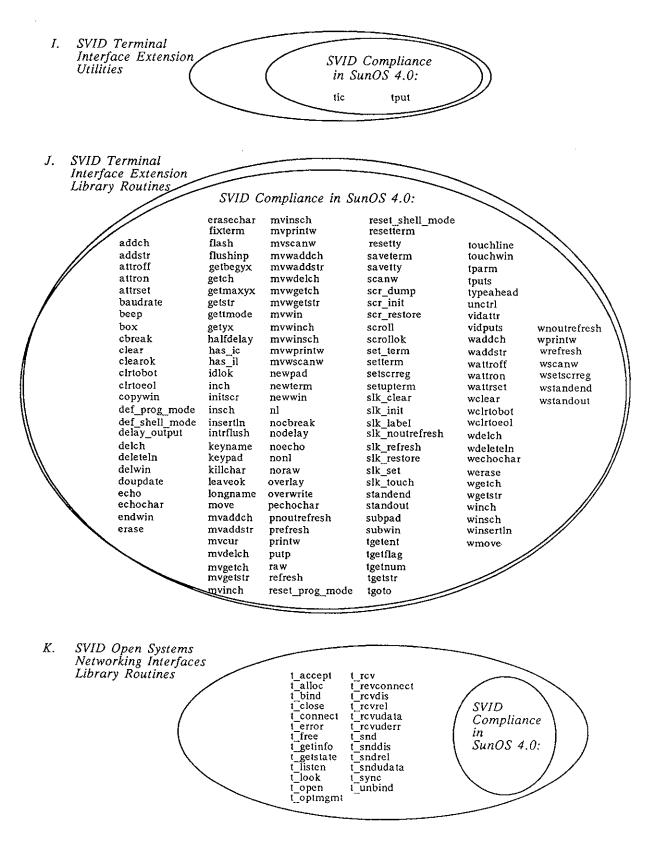


Figure 5: SunOS 4.0 and SVID Terminal and Network Interface Routines



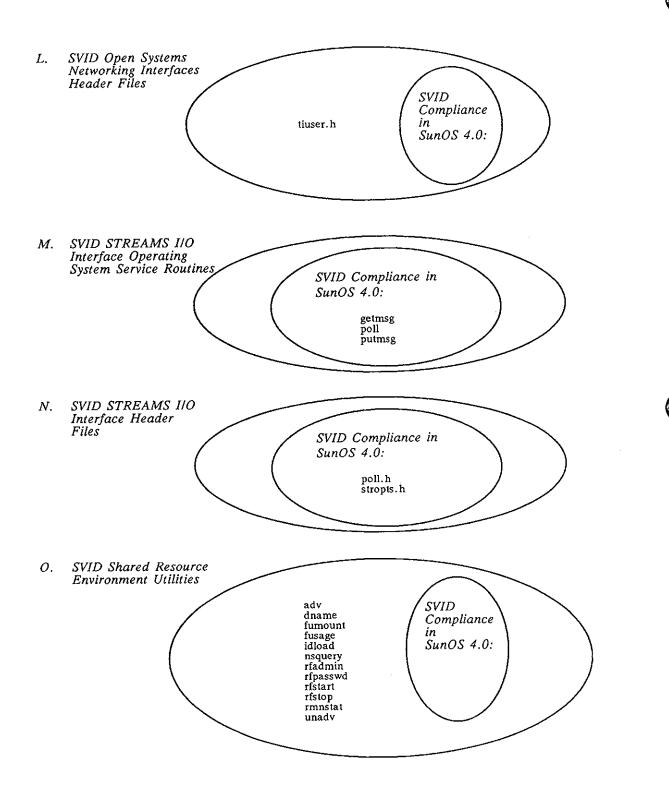


Figure 6: SunOS 4.0 and SVID Networking, Header, and Shared Resource Utility



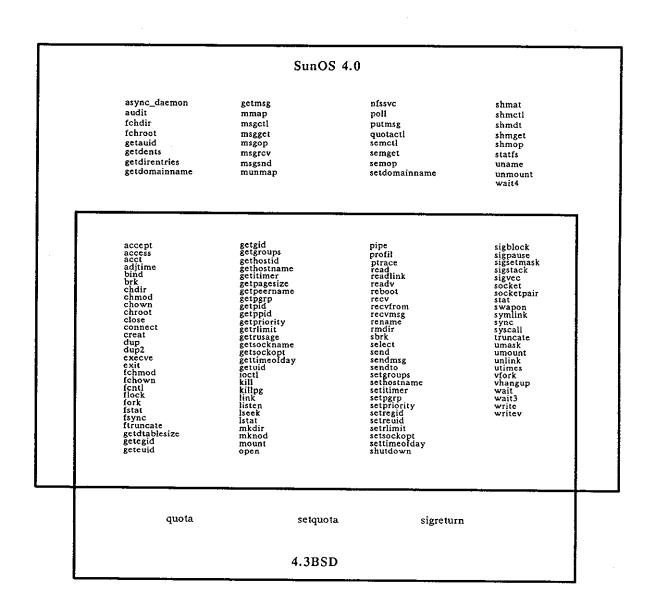


Figure 7: SunOS 4.0 and 4.3BSD System Calls



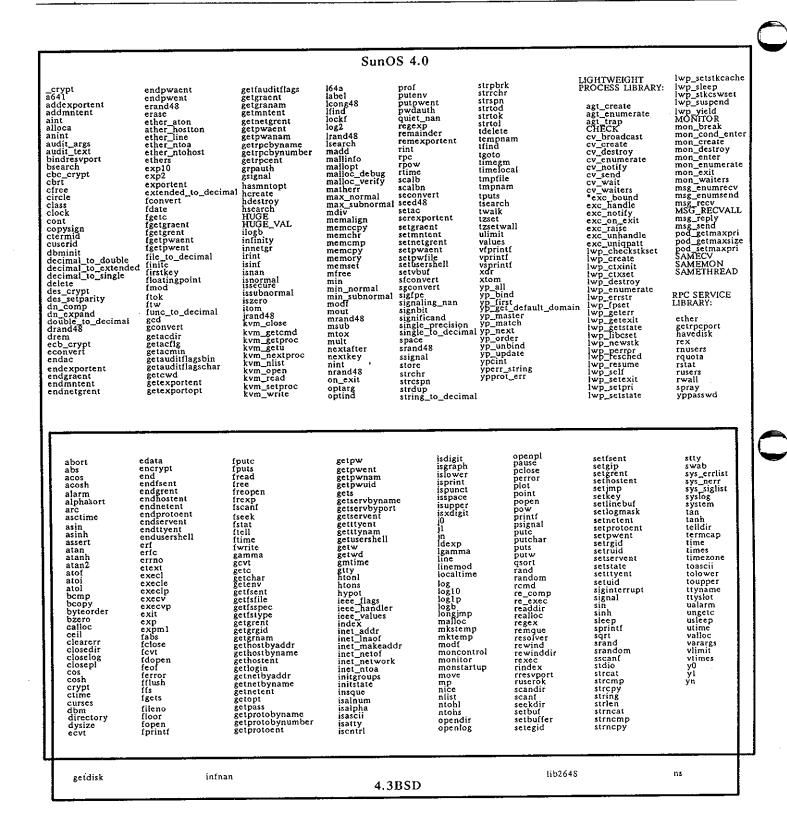


Figure 8: SunOS 4.0 and 4.3BSD Library Routines



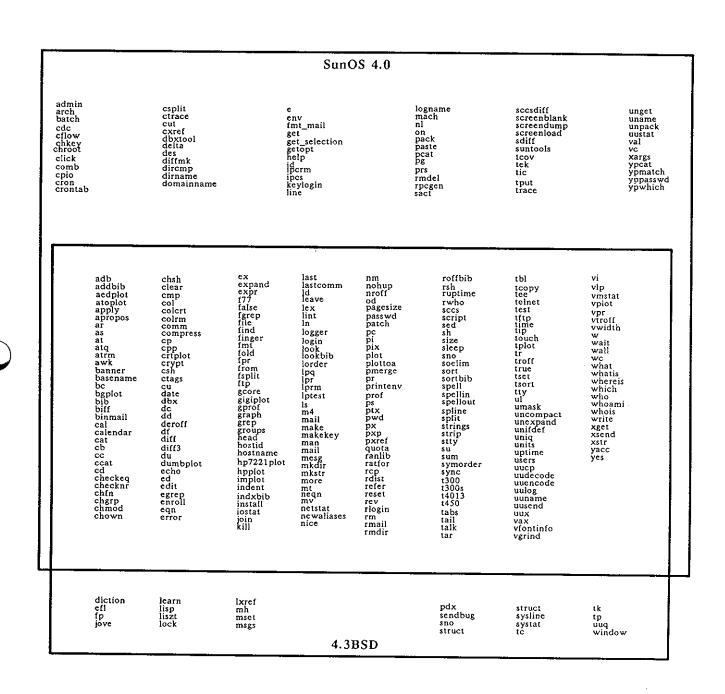
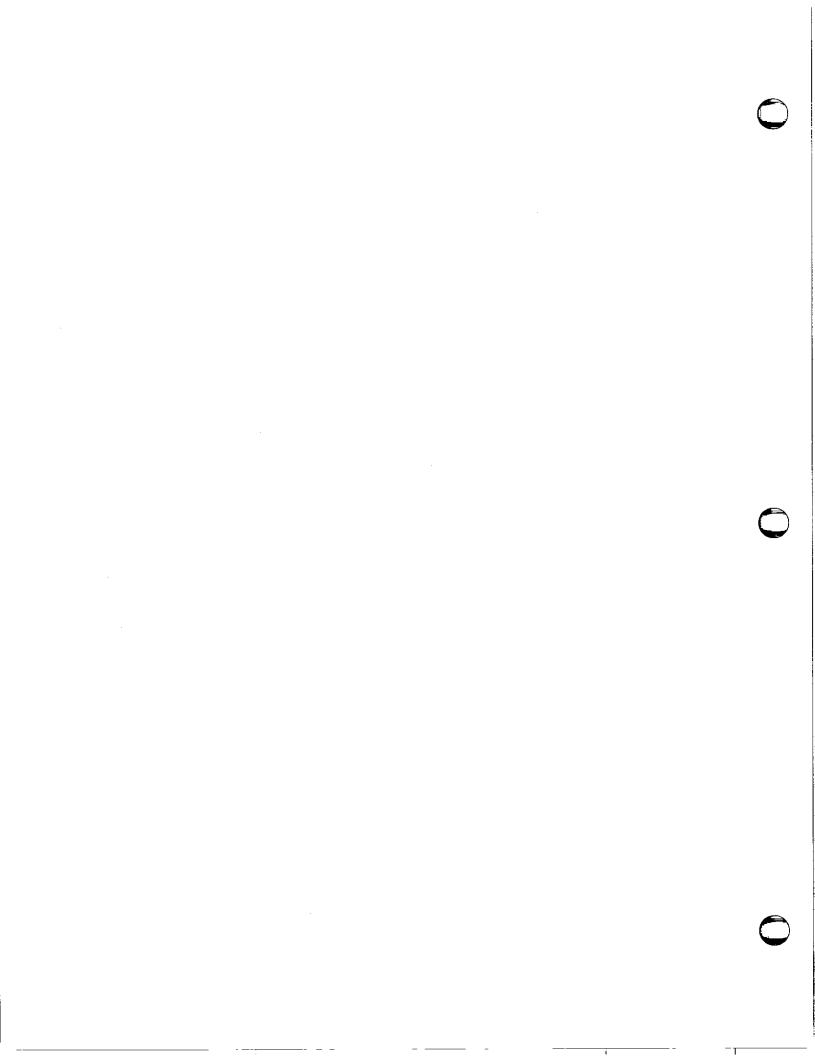


Figure 9: SunOS 4.0 and 4.3BSD Commands





# QUESTIONS, ANSWERS, HINTS, AND TIPS

QUESTIONS, ANSWERS, HINTS, AND TIPS			
Q&A, and Tip of the Month	679		

. .

## QUESTIONS, ANSWERS, HINTS, AND TIPS

Q&A, and Tip of the Month

Hints & Tips #12

This is the twelfth in a continuing series of this column which I have created for two purposes. <sup>12</sup> First, some questions are asked regularly on the AnswerLine. I feel everyone can benefit from distributing discussions of these problems as widely as possible. Second, a large and constantly growing body of information, hints, and tips are not documented anywhere.

I will collect and distribute these information nuggets in this continuing column so that we can all learn from them. I will cover unusual topics, but this column should not be used as an alternative to contacting your support center or using the AnswerLine.

If you have a question that you would like answered in this column, please mail your question to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Avenue, M/S 2-312, Mountain View, CA 94043. You can also send in your question by electronic mail to *sun!stb-editor*. U. S. customers can call Sun Customer Software Services AnswerLine at 800 USA-4-SUN for technical questions on this column or any other article in this bulletin. I look forward to hearing from you!

Local and Remote Files

Is the file you are using on a local disk or mounted via NFS? Normally, you do not care, but there are situations where you might want to know. The program filetype.c, appearing on page 309 of the March 1988 STB, shows you how to find out. The code is repeated in this article for your convenience.

Why might you want this? If you are working on a program that does a lot of random access in a temporary file that would finally be renamed to some permanent name at the end of processing, it would make sense to put the temporary file in its final home if that directory is local. It might be better to move it to /tmp on a local disk if the directory is NFS-mounted. You would

<sup>12</sup> This continuing column is submitted by Chuq Von Rospach, Customer Software Services.



then copy the data across the network only at the end of the processing. If the majority of the file accesses are done during the random access, this would save a large amount of network activity and improve performance.

These kind of performance concerns are not issues that the user needs to decide. Use the program shown on the next page to figure out what method is best to use. This program, filetype.c, shows how to tell whether the file is on a local disk or mounted via the network. Note that for the purposes of this example, ND partitions are considered local.



```
* filetype.c -- local or NFS mounted file?
 * chuq von rospach Sun Software Technical support
 */
#include <stdio.h>
#include <mntent.h>
#include <sys/types.h>
#include <sys/stat.h>
main(argc, argv)
    int
                    argc;
    char
                   *argv[];
{
    struct stat
                    s, s2;
   FILE
                   *mntent;
   struct mntent *mp;
   if (argc != 2) {
        fprintf(stderr, "Usage: %s <filename> \n", argv[0]);
       exit(1);
   if (stat(argv[1], &s) < 0) {
       fprintf(stderr, "%s: can't stat %s\n", argv[0], argv[1]);
       exit(-1);
   if ((mntent = setmntent(MOUNTED, "r")) == 0) {
       fprintf(stderr, "%s: can't setmntent %s\n", argv[0], MOUNTED);
       exit(-1);
   while ((mp = getmntent(mntent)) != 0) {
       if (strcmp(mp->mnt_fsname, argv[1]) == 0) {
            endmntent (mntent);
           printf("%s\n", mp->mnt_type);
           exit(0);
       if (stat(mp->mnt_dir, &s2) < 0) {</pre>
           fprintf(stderr, "%s: can't stat %s\n", argv[0], mp->mnt_fsname);
           exit(-1);
       if (s.st_dev == s2.st_dev) {
           endmntent(mntent);
           printf("%s\n", mp->mnt_type);
           exit(0);
       }
   fprintf(stderr, "%s: couldn't find mount point for %s\n", argv[0], argv[1]);
   exit(1);
```



#### Tip of the Month

Are you one of those people who tends to accidentally hit the <Caps> key in the lower, left corner of your Sun3 keyboard while reaching for the left shift key? Are you one of those people who gets frustrated about the fifth time you do this? I am, and so I have found this program, capset.c by Richard Morin, a real joy. It allows you to turn off the offending key when it gets in the way. I simply put it in my .login file so the key is always disabled.

Besides this use, this program is a good example of how to work with the keyboard mask in general, so you could use this technique to, for instance, create a Dvorak keyboard. Some users feel that the alternate key mapping in the Dvorak keyboard allows faster typing from the standard 'qwerty' keyboard mapping.

Using the capset.c Program

To use the program to turn < Caps> on, follow the procedure shown below.

machine% make capset machine% capset on

To use the program to turn < Caps> off, use either of the two commands shown below.

machine% capset off
machine% capset

The Program Code

The capset.c program appears on the following pages.



```
|| capset.c - Control activities of "Caps" key on SUN3 keyboard.
|| Usage: capset [on|off]
+1
[ ]
        turn "Caps" key off
|| off turn "Caps" key off
        turn "Caps" key on
11
|| Freely redistributable, so long as this notice is retained.
|| Richard Morin, proprietor
                                    {hoptoad, leadsv, lll-lcc}!cfcl!rdm
|| Canta Forda Computer Lab.
                                    +1 415 994 6860
|| Post Office Box 1488
                                    Full spectrum consulting services
|| Pacifica, CA 94044 USA
                                   for science and engineering.
*/
#include <stdio.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sundev/kbd.h>
#include <sundev/kbio.h>
#define A SHIFTKEYS+CAPSLOCK
#define I NOP
main(argc, argv)
    int
                    argc;
    char
                  **argv;
{
    int
                    fd, index, type;
    static int
                    maps[] = {A, A, A, A, I},
                    masks[] = {0, CAPSMASK, SHIFTMASK, CTRLMASK, UPMASK},
                    mode = -1;
   struct kiockey key;
   switch (argc) {
    case 1:
       mode = 0;
       break;
   case 2:
       if (strcmp(argv[1], "off") == 0)
            mode = 0;
       else if (strcmp(argv[1], "on") == 0)
           mode = 1;
       break;
   }
```

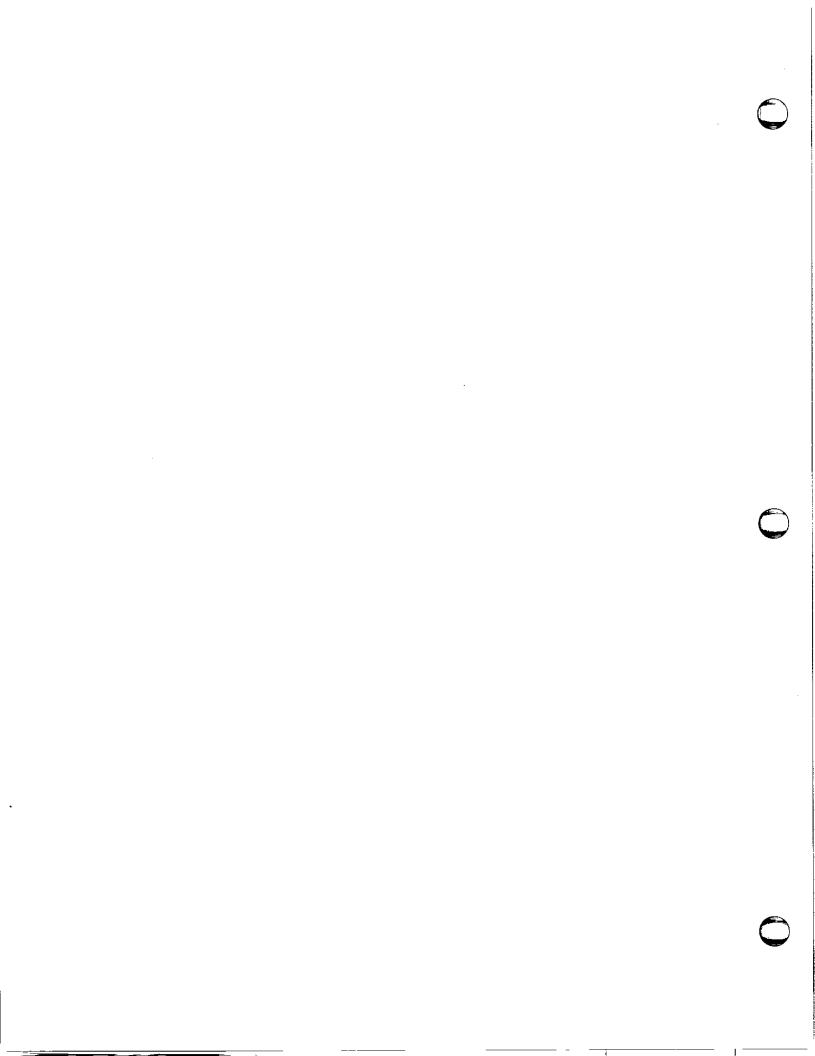


}

```
if (mode < 0) {
    fprintf(stderr, "Usage: capset [on|off]\n");
    exit(1);
if ((fd = open("/dev/kbd", O_RDWR)) == -1) {
    perror ("capset");
    exit(2);
if (ioctl(fd, KIOCTYPE, &type) == -1) {
    perror("capset");
    exit(3);
if (type != KB_SUN3) {
    fprintf(stderr, "capset: Not SUN3 kbd (%d)\n", type);
    exit(4);
key.kio_station = 0x77; /* "Caps" key */
                        /* Kill it */
key.kio_entry = NOP;
for (index = 0; index < 5; index++) {
    if (mode == 1)
        key.kio_entry = maps[index];/* Restore it */
    key.kio_tablemask = masks[index];
    if (ioctl(fd, KIOCSETKEY, &key) == -1) {
        perror("capset");
        exit(5);
    }
}
```

## THE HACKERS' CORNER

THE HACKERS'	CORNER	 687
VMS Tape Backup	*************************	 687



### THE HACKERS' CORNER

#### VMS Tape Backup

#### Reading a VMS Backup Tape

This month's **Hackers' Corner** contains a program and documentation to read a VMS-generated backup tape, convert the files to the UNIX format, and write the files to disk. The program was written by John Douglas Carey, and Sven-Ove Westberg from Lulea University of Technology, S-951 87 Lulea, Sweden.

Please consult your local shell script or programming expert regarding any script or code problems. The example programs are not offered as a supported Sun product, but as items of interest to enthusiasts wanting to try out something for themselves. Note that **Hackers' Corner** code may not work in all cases, and may not be compatible with future SunOS releases.

#### The vmsbackup Program

vmsbackup has the default operation of going through an entire VMS-generated backup tape, extracting every file, and writing it to disk. Program operation may be modified by the options described in the following paragraphs.

Synopsis

The synopsis for vmsbackup is shown below.

vmsbackup -{tx}[cdevw][s setnumber][f tapefile] [ name ... ]

**Program Options** 

The nine programs options are described below.

□ C

Complete. Use complete filenames, including the version number. A colon and the octal version number will be appended to all filenames. A colon, rather than a semicolon, is used since the UNIX shell uses the semicolon as the line separator. Using a colon prevents the user from having to escape the semicolon when referencing the filename. This option is useful only when multiple versions of the same file are on a single tape or when a file of the same name already exists in the destination directory. The default is to ignore version numbers.



Note: The filename match uses the complete VMS file names.

□ **d** 

Directory. Use the directory structure from VMS. The default value is off

.

Extensions. Process all filename extensions. Since this program is mainly intended to move source code and possibly data from a DEC system to a UNIX system, the default is to ignore all files whose filename extension specifies system-dependent data. The file types which will be ignored, unless the **e** option is specified, are listed below.

exe VMS executable file

lib VMS object library file

obj RSX object file

odl RSX overlay description file

olb RSX object library file

pmd RSX post mortem dump file

stb RSX task symbol table file

sys RSX bootable system file

tsk RSX executable task file

o **f** 

Tape Device. Use the next argument in the command line as the tape device to be used, rather than the default.

If vmsbackup is compiled with the remote tape option and the file name has the form

system [. user ]:/dev/???,

the program will use the tape drive /dev/??? on the remote system via rsh(1) and rmt(8). The optional user portion of the pathname specifies the login name to use on the remote system. If it is not supplied, the current user's login name will be used.

In all the cases, the user must have the appropriate permissions on the remote machine, in order to use this facility.

Saveset. Process only the given saveset number.

□ t

Table of Contents. Produce a table of contents (a directory listing) on the standard output, of the files on the tape.



- v

Verbose Output. Normally vmsbackup works silently. The verbose option will cause the filenames of the files being read from tape to disk to be output on the standard output.

. 1

Wait. The program prints the action to be taken followed by file name, then waits for user confirmation. If a word beginning with 'y' is given, the action is done. Any other input means do not do it.

□ **x** 

**Extract.** The program extracts the named files from the tape. The optional *name* argument specifies one or more filenames to be searched for specifically on the tape and only those files are to be processed. The name may contain the usual sh(1) meta-characters \*?![] \nnn.

The vmsbackup Program Code

The code for the vmsbackup program appears on the following pages.



```
#!/bin/sh
# to extract, remove the header and type "sh filename"
This progam reads a VMS backuptape.
The tape program is orginally written by John Douglas Carey and
the pattern matching routine by some unknown on the net.
The remote tape option use the rmtlib from mod.sources.
A good way to archive remotetape access for users with only
a local account is to create a "netwide" user tar and let
the remote tape programs do suid to user tar.
The program is tested on vax and sun.
Sven-Ove Westberg
Lulea University of Technology
S-951 87 Lulea, Sweden
UUCP: sow@luthcad.UUCP
UUCP: {decvax, philabs, seismo}!mcvax!enea!luthcad!sow
REMOTE = - DREMOTE
                    # -DREMOTE use remote tape
                    # -DSWAP swap bytes
SWAP=
CFLAGS=$(SWAP)
LFLAGS=
                    # -lrmt remote magtape library
LIBS=
                    # user for remote tape access
OWNER=tar
MODE=4755
BINDIR=/usr/local/bin
MANSEC=1
MANDIR=/usr/man/man$ (MANSEC)
vmsbackup: vmsbackup.o match.o
    cc $(LFLAGS) -o vmsbackup vmsbackup.o match.o $(LIBS)
install:
    install -m $(MODE) -o $(OWNER) -s vmsbackup $(BINDIR)
    cp vmsbackup.1 $(MANDIR)/vmsbackup.$(MANSEC)
clean:
    rm -f vmsbackup *.o core
#include <stdio.h>
#include <sys/types.h>
                            /* The '*' metacharacter */
#define ASTERISK '*'
#define QUESTION '?'
                            /* The '?' metacharacter */
#define LEFT_BRACKET '['
                            /* The '[' metacharacter */
                            /* The ']' metacharacter */
#define RIGHT BRACKET ']'
```



```
#define IS_OCTAL(ch) (ch >= '0' && ch <= '7')
typedef int BOOLEAN;
#define VOID void
#define TRUE 1
#define FALSE 0
#define EOS '\000'
static BOOLEAN do_list ();
static char nextch ();
static VOID list_parse ();
   FUNCTION
   match
            test string for wildcard match
   SYNOPSIS
  BOOLEAN match (string, pattern)
   register char *string;
   register char *pattern;
   DESCRIPTION
   Test string for match using pattern. The pattern may
   contain the normal shell metacharacters for pattern
   matching. The '*' character matches any string,
   including the null string. The '?' character matches
   any single character. A list of characters enclosed
   in '[' and ']' matches any character in the list.
* If the first character following the beginning '['
   is a '!' then any character not in the list is matched.
*/
   PSEUDO CODE
   Begin match
       Switch on type of pattern character
*
       Case ASTERISK:
*
           Attempt to match asterisk
           Break
       Case QUESTION MARK:
           Attempt to match question mark
       Case EOS:
           Match is result of EOS on string test
           Break
       Case default:
```



```
If explicit match then
            Match is result of submatch
            Else
            Match is FALSE
            End if
            Break
        End switch
        Return result of match test
    End match
 */
BOOLEAN match (string, pattern)
register char *string;
register char *pattern;
{
    register BOOLEAN ismatch;
    ismatch = FALSE;
    switch (*pattern) {
    case ASTERISK:
        pattern++;
        do {
        ismatch = match (string, pattern);
        } while (!ismatch && *string++ != EOS);
        break;
    case QUESTION:
        if (*string != EOS) {
        ismatch = match (++string, ++pattern);
        break;
    case EOS:
        if (*string == EOS) {
        ismatch = TRUE;
        break:
    case LEFT_BRACKET:
        if (*string != EOS) {
        ismatch = do_list (string, pattern);
        break;
    default:
        if (*string++ == *pattern++) {
        ismatch = match (string, pattern);
        } else {
        ismatch = FALSE;
        }
        break;
    return (ismatch);
}
```



```
FUNCTION
   do_list
              process a list and following substring
   SYNOPSIS
   static BOOLEAN do_list (string, pattern)
   register char *string;
   register char *pattern;
   DESCRIPTION
  Called when a list is found in the pattern. Returns
   TRUE if the current character matches the list and
   the remaining substring matches the remaining pattern.
   Returns FALSE if either the current character fails to
   match the list or the list matches but the remaining
   substring and subpattern's don't.
   RESTRICTIONS
   The mechanism used to match characters in an inclusive
   pair (I.E. [a-d]) may not be portable to machines
   in which the native character set is not ASCII.
   The rules implemented here are:
       (1) The backslash character may be
           used to quote any special character.
           I.E. "\]" and "\-" anywhere in list,
           or "\!" at start of list.
       (2) The sequence \nnn becomes the character
           given by nnn (in octal).
       (3) Any non-escaped ']' marks the end of list.
       (4) A list beginning with the special character
           '!' matches any character NOT in list.
           The '!' character is only special if it
           is the first character in the list.
*/
  PSEUDO CODE
  Begin do_list
      Default result is no match
```



Skip over the opening left bracket

```
If the next pattern character is a '!' then
       List match gives FALSE
       Skip over the '!' character
       Else
       List match gives TRUE
       End if
       While not at closing bracket or EOS
*
       Get lower and upper bounds
       If character in bounds then
           Result is same as sense flag.
            Skip over rest of list
       End if
       End while
        If match found then
        If not at end of pattern then
            Call match with rest of pattern
       End if
        End if
        Return match result
   End do_list
 */
static BOOLEAN do_list (string, pattern)
register char *string;
char *pattern;
{
    register BOOLEAN ismatch;
    register BOOLEAN if_found;
    register BOOLEAN if_not_found;
    auto char lower;
    auto char upper;
    pattern++;
    if (*pattern == '!') {
    if_found = FALSE;
    if not found = TRUE;
    pattern++;
    } else {
    if found = TRUE;
    if_not_found = FALSE;
    ismatch = if not found;
    while (*pattern != ']' && *pattern != EOS) {
    list_parse (&pattern, &lower, &upper);
    if (*string >= lower && *string <= upper) {
        ismatch = if_found;
        while (*pattern != ']' && *pattern != EOS) {pattern++;}
    }
    if (*pattern++ != ']') {
    fprintf (stderr, "warning - character class error\n");
    } else {
```



```
if (ismatch) {
         ismatch = match (++string, pattern);
    }
    }
    return (ismatch);
}
/*
 *
    FUNCTION
    list parse
                   parse part of list into lower and upper bounds
    SYNOPSIS
   static VOID list_parse (patp, lowp, highp)
    char **patp;
    char *lowp;
    char *highp;
    DESCRIPTION
    Given pointer to a pattern pointer (patp), pointer to
    a place to store lower bound (lowp), and pointer to a
    place to store upper bound (highp), parses part of
    the list, updating the pattern pointer in the process.
    For list characters which are not part of a range,
    the lower and upper bounds are set to that character.
 */
static VOID list_parse (patp, lowp, highp)
char **patp;
char *lowp;
char *highp;
    *lowp = nextch (patp);
    if (**patp == '-') {
    (*patp)++;
    *highp = nextch (patp);
    } else {
    *highp = *lowp;
}
   FUNCTION
              determine next character in a pattern
   nextch
   SYNOPSIS
```



```
static char nextch (patp)
   char **patp;
   DESCRIPTION
   Given pointer to a pointer to a pattern, uses the pattern
   pointer to determine the next character in the pattern,
   subject to translation of backslash-char and backslash-octal
   sequences.
   The character pointer is updated to point at the next pattern
   character to be processed.
 */
static char nextch (patp)
char **patp;
{
    register char ch;
    register char chsum;
    register int count;
    ch = *(*patp)++;
    if (ch == '\\') {
    ch = *(*patp)++;
    if (IS OCTAL (ch)) {
        chsum = 0;
        for (count = 0; count < 3 && IS_OCTAL (ch); count++) {
        chsum *= 8;
        chsum += ch - '0';
        ch = *(*patp)++;
        (*patp) --;
        ch = chsum;
    }
    return (ch);
}
/*
    Title:
    Backup
    Decription:
    Program to read VMS backup tape
    Author:
    John Douglas CAREY.
                          (version 3.0)
    Sven-Ove Westberg
    Net-addess:
```



```
john%monu1.oz@seismo.ARPA
   luthcad!sow@enea.UUCP
   History:
   Version 1.0 - September 1984
       Can only read variable length records
   Version 1.1
       Cleaned up the program from the original hack
       Can now read stream files
   Version 1.2
       Now convert filename from VMS to UNIX
           and creates sub-directories
   Version 1.3
       Works on the Pyramid if SWAP is defined
   Version 1.4
       Reads files spanning multiple tape blocks
   Version 1.5
       Always reset reclen = 0 on file open
       Now output fixed length records
       Version 2.0 - July 1985
       VMS Version 4.0 causes a rethink !!
       Now use mtio operations instead of opening and closing file
       Blocksize now grabed from the label
   Version 2.1 - September 1985
       Handle variable length records of zero length.
   Version 2.2 - July 1986
       Handle FORTRAN records of zero length.
       Inserted exit(0) at end of program.
      Distributed program in aus.sources
  Version 2.3 - August 1986
       Handle FORTRAN records with record length fields
       at the end of a block
       Put debug output to a file.
      Distributed program in net.sources
  Version 3.0 - December 1986
*
      Handle multiple saveset
      Remote tape
      Interactive mode
      File name selection with meta-characters
      Convert ; to : in VMS filenames
      Flag for usage of VMS directory structure
      Flag for "useless" files eg. *.exe
      Flag for use VMS version in file names
      Flag for verbose mode
      Flag to list the contents of the tape
      Distributed to mod.sources
```



```
Installation:
   Computer Centre
   Monash University
   Wellington Road
   Clayton
   Victoria
                3168
   AUSTRALIA
 */
#include
            <stdio.h>
#include
            <ctype.h>
#include
            <sys/ioctl.h>
#include
            <sys/types.h>
#ifdef REMOTE
            <local/rmt.h>
#include
#include
            <sys/stat.h>
#endif
#include
            <sys/mtio.h>
            <sys/file.h>
#include
#ifdef pyr
#define SWAP
#endif pyr
#ifdef sun
#define SWAP
#endif
struct bbh {
            bbh_dol_w_size;
    short
    short
            bbh_dol_w_opsys;
            bbh_dol_w_subsys;
    short
    short
            bbh_dol_w_applic;
    long
            bbh dol 1 number;
    char
            bbh_dol_t_spare_1[20];
    short
            bbh_dol_w_struclev;
            bbh_dol_w_volnum;
    short
    long
            bbh dol 1 crc;
            bbh_dol_l_blocksize;
    long
            bbh dol l flags;
    long
    char
            bbh_dol_t_ssname[32];
            bbh dol w fid[3];
    short
            bbh_dol_w_did[3];
    short
            bbh dol t filename[128];
    char
            bbh dol b rtype;
    char
    char
            bbh dol b_rattrib;
            bbh_dol_w_rsize;
    short
            bbh_dol_b_bktsize;
    char
    char
            bbh_dol_b_vfcsize;
            bbh_dol_w_maxrec;
    short
            bbh_dol_l_filesize;
    long
```



```
bbh_dol_t_spare_2[22];
    short
            bbh_dol_w_checksum;
} *block_header;
struct brh {
    short
            brh_dol_w rsize;
            brh_dol_w_rtype;
    short
    long
            brh_dol_1_flags;
    long
            brh dol l address;
    long
            brh dol 1 spare;
} *record header;
/* define record types */
#define brh_dol_k_null
#define brh_dol_k_summary
#define brh dol k volume
#define brh_dol k file
#define brh_dol k vbn
#define brh_dol_k_physvol
                             5
#define brh dol k lbn
#define brh_dol_k fid
struct bsa {
    short
            bsa_dol_w_size;
    short
            bsa_dol_w_type;
    char
            bsa_dol_t_text[1];
} *data_item;
#ifdef
        STREAM
char
        *def_tapefile = "/dev/rts8";
#else
char
        *def_tapefile = "/dev/rmt8";
#endif
char
        *tapefile;
char
        filename[128];
int filesize;
char
        recfmt;
                    /* record format */
#define FAB dol C UDF
                        0 /* undefined */
#define FAB_dol_C FIX
                        1 /* fixed-length record */
#define FAB_dol_C_VAR
                        2 /* variable-length record */
#define FAB_dol_C VFC
                        3 /* variable-length with fixed-length control record */
#define FAB_dol_C STM
                        4 /* RMS-11 stream record (valid only for sequential org) */
#define FAB_dol_C_STMLF 5 /* stream record delimited by LF (sequential org only) */
#define FAB_dol_C_STMCR 6 /* stream record delimited by CR (sequential org only) */
#define FAB_dol_C_MAXRFM 6 /* maximum rfm supported */
char
        recatt;
                    /* record attributes */
#define
                FAB dol V FTN
                                0 /* FORTRAN carriage control character */
```



```
1 /* line feed - record -carriage return */
#define
                FAB dol V CR
#define
                FAB_dol_V_PRN
                                2 /* print-file carriage control */
                                3 /* records don't cross block boundaries */
                FAB_dol_V_BLK
#define
#define FANO
#ifdef pyr
                bsa *file_table[FANO];
static struct
#else
struct bsa *file table[FANO];
#endif
FILE
        *f = NULL;
int file_count;
short
        reclen;
short
        fix;
short recsize;
int vfcsize;
#ifdef NEWD
        *1f;
FILE
#endif NEWD
            /* tape file descriptor */
int fd;
        cflag, dflag, eflag, sflag, tflag, vflag, wflag, xflag;
int
int setnr;
char
        **gargv;
int
        goptind, garge;
#define LABEL_SIZE 80
        label[LABEL SIZE];
char
        *block;
char
int blocksize;
struct mtop
                op;
FILE *
openfile (fn)
        *fn;
char
{
            ufn[256];
    char
            ans[80];
    char
    char
            *p, *q, s, *ext;
    int procf;
    procf = 1;
    /* copy fn to ufn and convert to lower case */
    p = fn;
    q = ufn;
    while (*p) {
        if (isupper(*p))
            *q = *p - 'A' + 'a';
```



```
else
             *q = *p;
         p++;
         q++;
    *q = ' \ 0';
    /* convert the VMS to UNIX and make the directory path */
    p = ufn;
    q = ++p;
    while (*q) {
         if (*q == '.' || *q == ']') {
             s = *q;
             *q = ' \setminus 0';
             if (procf && dflag) mkdir(p, 0777);
             *q = '/';
             if (s == ']')
                 break;
         *q++;
    }
    *q++;
    if(!dflag) p=q;
    /* strip off the version number */
    while (*q && *q != ';') {
        if( *q == '.') ext = q;
        q++;
    }
    if (cflag) {
        *q = ':';
    }
    else {
        *q = ' \setminus 0';
    if(!eflag && procf) procf = typecmp(++ext);
    if(procf && wflag) {
        printf("extract %s [ny]", filename);
        fflush(stdout);
        gets (ans);
        if(*ans != 'y') procf = NULL;
    if (procf)
        /* open the file for writing */
        return(fopen(p, "w"));
    else
        return (NULL);
typecmp(str)
                 /* Compare the filename type in str with our list
                    of file type to be ignored. Return 0 if the
                    file is to be ignored, return 1 if the
                    file is not in our list and should not be ignored. */
                 *str;
register char
```



}

```
{
        static char *type[] = {
                                 /* vms executable image */
                "exe",
                                /* vms object library */
                "lib",
                                 /* rsx object file */
                "obj",
                                 /* rsx overlay description file */
                "odl",
                                /* rsx object library */
                "olb",
                                /* rsx post mortem dump */
                "pmd",
                                 /* rsx symbol table */
                "stb",
                                /* rsx bootable system image */
                "sys",
                                /* rsx executable image */
                "tsk",
        "dir",
        "upd",
        "tlo",
        "tlb",
                                 /* null string terminates list */
        };
        register int
                        i;
        i = -1;
        while (*type[++i])
                if (strncmp(str, type[i], 3) == 0)
                                         /* found a match, file to be ignored */
                         return(0);
                                         /* no match found */
        return(1);
}
process file (buffer)
        *buffer;
char
    int i, n;
    char
            *p, *q;
            dsize, nblk, lnch;
    short
    int c;
    short
            *s;
    int
            procf;
    s = (short *) buffer;
    /* check the header word */
    if (*s != 257) {
        printf("Snark: invalid data header\n");
        exit(1);
    }
    c = 2;
    for (i = 0; i < FANO; i++) {
        file table[i] = (struct bsa *) &buffer[c];
#ifndef SWAP
        dsize = file_table[i]->bsa_dol_w_size;
#else
        swap(&file_table[i]->bsa_dol_w_size, &dsize, sizeof(short));
```



```
#endif
        c += dsize + 4;
    /* extract file name */
#ifndef SWAP
    dsize = file_table[0]->bsa_dol_w_size;
#else
    swap(&file_table[0]->bsa_dol_w_size, &dsize, sizeof(short));
#endif
    p = file_table[0]->bsa_dol_t_text;
    q = filename;
    for (i = 0; i < dsize; i++)
        *q++ = *p++;
    *q = ' \setminus 0';
    /* extract file's record attributes */
#ifndef SWAP
    dsize = file_table[5]->bsa_dol_w_size;
#else
    swap(&file_table[5]->bsa dol w size, &dsize, sizeof(short));
#endif
    p = file_table[5]->bsa_dol_t_text;
    recfmt = p[0];
    recatt = p[1];
#ifndef SWAP
    bcopy(&p[2], &recsize, sizeof(short));
#else
    swap(&p[2], &recsize, sizeof(short));
#endif
    vfcsize = p[15];
    if (vfcsize == 0)
        vfcsize = 2;
#ifdef DEBUG
   printf("recfmt = %d\n", recfmt);
   printf("recatt = %d\n", recatt);
   printf("reclen = %d\n", recsize);
   printf("vfcsize = %d\n", vfcsize);
#endif
#ifndef SWAP
   bcopy(&p[10], &nblk, sizeof(short));
   bcopy(&p[12], &lnch, sizeof(short));
#else
    swap(&p[10], &nblk, sizeof(short));
    swap(&p[12], &lnch, sizeof(short));
#endif
    filesize = (nblk-1)*512 + lnch;
#ifdef DEBUG
   printf("nbk = %d, lnch = %d\n", nblk, lnch);
   printf("filesize = 0x%x\n", filesize);
#endif
    /* open the file */
```



```
if (f != NULL) {
        fclose(f);
        file count = 0;
        reclen = 0;
    procf = 0;
    if (goptind < garge)</pre>
        for(i=goptind; i < gargc; i++) {</pre>
            procf |= match(filename,gargv[i]);
        }
    else
        procf = 1;
    if (tflag && procf)
        printf( " %-35s %8d \n", filename, filesize);
    if (xflag && procf) {
        /* open file */
        f = openfile(filename);
        if(f != NULL && vflag) printf("extracting %s\n", filename);
    }
}
    process a virtual block record (file record)
 */
process_vbn(buffer, rsize)
            *buffer;
unsigned short rsize;
    int c, i;
    if (f == NULL) {
        return;
    i = 0;
    while (file_count+i < filesize && i < rsize) {
        switch (recfmt) {
        case FAB_dol_C_FIX:
            if (reclen == 0) {
                 reclen = recsize;
            fputc(buffer[i], f);
            i++;
            reclen--;
            break;
        case FAB_dol_C_VAR:
        case FAB_dol_C_VFC:
            if (reclen == 0) {
                 reclen = *((short *) &buffer[i]);
#ifdef SWAP
                 swap(&reclen, &reclen, sizeof(short));
#endif
```



```
#ifdef NEWD
                fprintf(lf, "---\n");
                fprintf(lf, "reclen = %d\n", reclen);
                fprintf(lf, "i = %d\n", i);
                fprintf(lf, "rsize = %d\n", rsize);
#endif NEWD
                fix = reclen;
                i += 2;
                if (recfmt == FAB_dol_C_VFC) {
                     i += vfcsize;
                     reclen -= vfcsize;
            } else if (reclen == fix
                     && recatt == (1 << FAB_dol_V_FTN)) {
                     /****
                     if (buffer[i] == '0')
                         fputc('\n', f);
                     else if (buffer[i] == '1')
                        fputc('\f', f);
                     *** sow ***/
                     fputc(buffer[i],f); /** sow **/
                     i++;
                     reclen--;
            } else {
                fputc(buffer[i], f);
                i++;
                reclen--;
            if (reclen == 0) {
                fputc('\n', f);
                if (i & 1)
                    i++;
            }
            break;
        case FAB_dol_C_STM:
        case FAB_dol C STMLF:
            if (reclen < 0) {
                printf("SCREAM\n");
            if (reclen == 0) {
                reclen = 512;
            c = buffer[i++];
            reclen--;
            if (c == ' \setminus n') {
                reclen = 0;
            }
           fputc(c, f);
           break;
       case FAB_dol C STMCR:
           c = buffer[i++];
```



```
if (c == '\r')
                fputc(' \n', f);
           else
                fputc(c, f);
           break;
       default:
            fclose(f);
            unlink(filename);
            fprintf(stderr, "Invalid record format = %d\n", recfmt);
       }
   file_count += i;
#ifdef SWAP
/*
   do swapping for Motorola type architectures
*/
swap(from, to, nbytes)
char
       *from, *to;
int nbytes;
   int i, j;
   char temp[100];
   for (i = 0; i < nbytes; i++)
        temp[i] = from[i];
    for (i = 0, j = nbytes-1; i < nbytes; i++, j--)
        to[i] = temp[j];
#endif
/*
 *
   process a backup block
 */
process_block(block, blocksize)
       *block;
char
int blocksize;
    unsigned short bhsize, rsize, rtype;
    unsigned long
                    bsize, i;
    i = 0;
    /* read the backup block header */
    block header = (struct bbh *) &block[i];
    i += sizeof(struct bbh);
```



```
bhsize = block_header->bbh_dol_w_size;
    bsize = block header->bbh dol 1 blocksize;
        SWAP
    swap(&bhsize, &bhsize, sizeof(short));
    swap(&bsize, &bsize, sizeof(long));
#endif
    /* check the validity of the header block */
    if (bhsize != sizeof(struct bbh)) {
        fprintf(stderr, "Snark: Invalid header block size\n");
        exit(1);
    if (bsize != 0 && bsize != blocksize) {
        fprintf(stderr, "Snark: Invalid block size\n");
        exit(1);
#ifdef DEBUG
    printf("new block: i = %d, bsize = %d\n", i, bsize);
#endif
    /* read the records */
    while (i < bsize) {
        /* read the backup record header */
        record_header = (struct brh *) &block[i];
        i += sizeof(struct brh);
        rtype = record header->brh dol w rtype;
        rsize = record_header->brh_dol_w_rsize;
#ifdef
        swap(&rtype, &rtype, sizeof(short));
        swap(&rsize, &rsize, sizeof(short));
#endif
#ifdef DEBUG
        printf("rtype = %d\n", rtype);
        printf("rsize = %d\n", rsize);
        printf("flags = 0x%x\n", record_header->brh_dol_1_flags);
        printf("addr = 0x%x\n", record_header->brh_dol 1 address);
        printf("i = %d\n", i);
#endif
        switch (rtype) {
        case brh_dol k_null:
#ifdef
       DEBUG
            printf("rtype = null\n");
#endif
            break;
        case brh_dol_k summary:
#ifdef
       DEBUG
            printf("rtype = summary\n");
#endif
           break;
```



```
case brh_dol_k_file:
#ifdef
        DEBUG
            printf("rtype = file\n");
#endif
            process_file(&block[i]);
            break;
        case brh dol_k_vbn:
#ifdef
        DEBUG
            printf("rtype = vbn\n");
#endif
            process_vbn(&block[i], rsize);
            break;
        case brh_dol_k_physvol:
#ifdef DEBUG
            printf("rtype = physvol\n");
#endif
            break;
        case brh_dol_k_lbn:
#ifdef
        DEBUG
            printf("rtype = lbn\n");
#endif
            break;
        case brh_dol_k_fid:
#ifdef
            printf("rtype = fid\n");
#endif
            break;
        default:
            fprintf(stderr, " Snark: invalid record type\n");
            fprintf(stderr, " record type = %d\n", rtype);
            exit(1);
#ifdef pyr
        i = i + rsize;
#else
        i += rsize;
#endif
    }
rdhead()
    int i, nfound;
    char name[80];
    nfound = 1;
    /* read the tape label - 4 records of 80 bytes */
    while ((i = read(fd, label, LABEL_SIZE)) != 0) {
        if (i != LABEL_SIZE) {
```



```
fprintf(stderr, "Snark: bad label record\n");
             exit(1);
         }
        if (strncmp(label, "VOL1",4) == 0) {
             sscanf(label+4, "%14s", name);
             if(vflag || tflag) printf("Volume: %s\n", name);
        if (strncmp(label, "HDR1", 4) == 0) {
             sscanf(label+4, "%14s", name);
            sscanf(label+31, "%4d", &setnr);
        /* get the block size */
        if (strncmp(label, "HDR2", 4) == 0) {
            nfound = 0;
             sscanf(label+5, "%5d", &blocksize);
#ifdef DEBUG
            printf("blocksize = %d\n", blocksize);
#endif
        }
    }
    if((vflag || tflag) && !nfound)
        printf("Saveset name: %s
                                  number: %d\n", name, setnr);
    /* get the block buffer */
    block = (char *) malloc(blocksize);
    if (block == (char *) 0) {
        fprintf(stderr, "memory allocation for block failed\n");
        exit(1);
    return (nfound);
}
rdtail()
    int i;
    char name[80];
    /* read the tape label - 4 records of 80 bytes */
    while ((i = read(fd, label, LABEL_SIZE)) != 0) {
        if (i != LABEL_SIZE) {
            fprintf(stderr, "Snark: bad label record\n");
            exit(1);
        if (strncmp(label, "EOF1",4) == 0) {
            sscanf(label+4, "%14s", name);
            if(vflag || tflag)
                printf("End of saveset: %s\n\n\n",name);
        }
    }
}
usage (progname)
char
        *progname;
   fprintf(stderr,
```



```
710
```

}

{

```
"Usage: %s -{tx}[cdevw][-s setnumber][-f tapefile]\n",progname);
main(argc, argv)
int argc;
char
        *argv[];
    char *progname;
    int c, i, eoffl;
    int selset;
    extern int optind;
    extern char *optarg;
    progname = argv[0];
    if(argc < 2){
        usage (progname);
        exit(1);
    }
    gargv = argv;
    gargc = argc;
    tapefile = def tapefile;
    cflag=dflag=eflag=sflag=tflag=vflag=wflag=xflag=0;
    while((c=getopt(argc,argv,"cdef:s:tvwx")) != EOF)
        switch(c){
        case 'c':
            cflag++;
            break;
        case 'd':
            dflag++;
            break;
        case 'e':
            eflag++;
            break;
        case 'f':
            tapefile = optarg;
            break;
        case 's':
             sflag++;
             sscanf (optarg, "%d", &selset);
            break;
        case 't':
             tflag++;
            break;
        case 'v':
             vflag++;
            break;
        case 'w':
             wflag++;
             break;
         case 'x':
             xflag++;
             break;
         case '?':
```



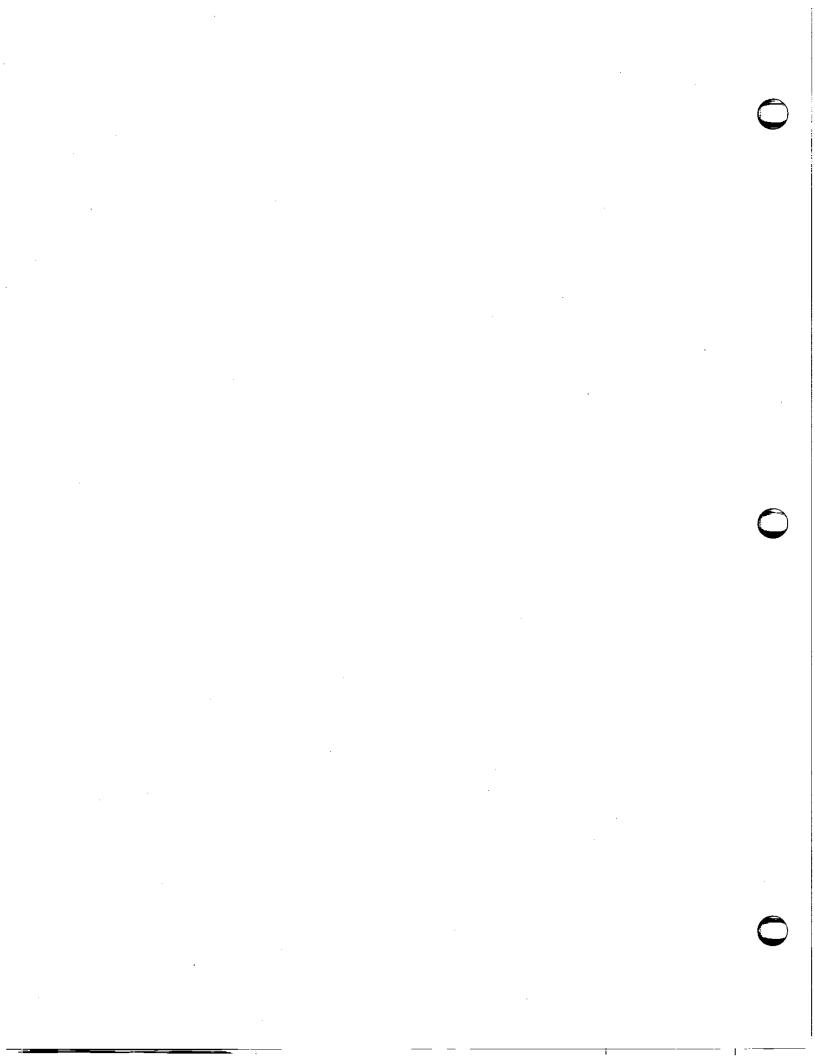
```
usage (progname);
            exit(1);
            break;
        };
    if(!tflag && !xflag) {
        usage (progname);
        exit(1);
    goptind = optind;
#ifdef NEWD
    /* open debug file */
    lf = fopen("log", "w");
    if (lf == NULL) {
       perror("log");
        exit(1);
#endif
    /* open the tape file */
    fd = open(tapefile, O_RDONLY);
    if (fd < 0) {
        perror(tapefile);
        exit(1);
    }
    /* rewind the tape */
   op.mt_op = MTREW;
   op.mt_count = 1;
   i = ioctl(fd, MTIOCTOP, &op);
   if (i < 0) {
       perror(tapefile);
       exit(1);
   }
   eoffl = rdhead();
   /* read the backup tape blocks until end of tape */
   while (!eoffl) {
       if(sflag && setnr != selset) {
           op.mt_op = MTFSF;
           op.mt_count = 1;
           i = ioctl(fd, MTIOCTOP, &op);
           if (i < 0) {
               perror(tapefile);
                exit(1);
           i = 0;
       }
       else
           i = read(fd, block, blocksize);
       if(i == 0) {
           rdtail();
           eoffl=rdhead();
```



```
else if (i != blocksize) {
           fprintf(stderr, "bad block read i = %d\n", i);
           exit(1);
       }
       else{
           eoffl = 0;
           process_block(block, blocksize);
       }
   if(vflag || tflag) printf("End of tape\n");
    /* close the tape */
   close(fd);
#ifdef NEWD
    /* close debug file */
   fclose(lf);
#endif NEWD
    /* exit cleanly */
    exit(0);
```

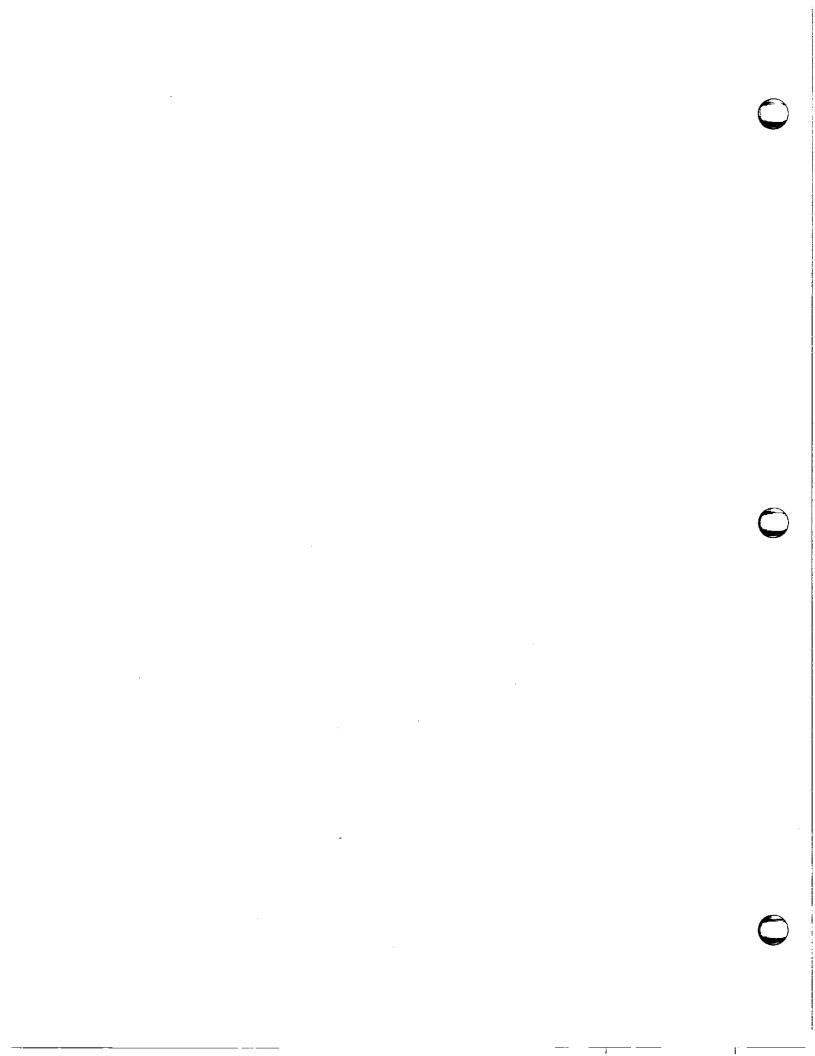
## **CUMULATIVE INDEX: 1988**

CUMULATIVE INDEX: 1988		715
------------------------	--	-----



**CUMULATIVE INDEX: 1988** 





## Index

1 1-800-USA-4-SUN device driver calls, 51
4 4.3BSD diagrams, 666
8 800 USA-4-SUN use of, 12
A address device drivers, 48 address mask, 67 addresses classes of, 107 Internet, 107 alias used with history, 78 aliases creating your own, 209 distribution lists, 210 mail, 155 receiving mail, 212 announcement NeWS 1.1, 635
Sun Modula-2, 629 AnswerLine, 9, 155, 255, 335, 679 device driver calls, 51 architecture Prism, 151 ARP, 109 assembler bugs
compilers, 348 <b>B</b>
back-to-back packets, 79 backup copy daemon Hackers' Corner, 261
beginner's guides documentation bugs, 428 bind
port numbers, 75 block size using plot(1G), 617

books C reference, 633 boot from PROM monitor, 73 booting specific kernel, 76 Bourne shell bugs bugs, 514 Bridge box, 81 broadcasting subnets, 107 brouchure Sun Education, 26 BSC3270 bugs, 400 **BSCRJE** bugs, 401 **BSD** 4.3 diagrams, 666 buffer Ethernet, 79 buffers color frame, 140 frame, 37 bug reporting, 13 bugs assembler, 348 beginner's guides documentation, 428 Bourne shell, 514 BSC3270 bugs, 400 BSCRJE bugs, 401 C compiler, 349 C shell, 514 CGI, 472 CGI documentation, 429 channel bugs, 402 compiler general, 396 compiler library, 383 compiler utilities, 397 compilers, 348 cross compilers, 399 datacomm, 400 datacomm documentation bugs, 405 DDN bugs, 403

debugger documentation, 432

debuggers, 355

1	bugs, continued	
bugs, continued	SunINGRES library, 495	
diagnostics, 418	SunINGRES program, 497	
DNI bugs, 403	SunPro, 522	
documentation, 419	SunSimplify, 523	
driver, 479	S.m.T. 505	
editor documentation, 434	SunUNIFY, 526	
editor utility, 562	SunView, 536	
f77SunCore Bugs, 474	SunView general, 545	
formatter utility, 563	SunView library, 536	
FORTRAN, 455	SunView program, 546	
FORTRAN compilers, 455	SunWindows, 551	
FORTRAN converter, 468	syscall, 485	
FORTRAN Cross Compiler, 399	system administration, 555	
FORTRAN documentation, 434, 468	system administration documentation, 443	
FORTRAN library, 469	transcript, 558	
general, 481	utilities, 562	
general utility, 568	uucp, 577	
GKS documentation, 435	vt100tool bugs, 413	
GP, 478	windows documentation, 419	
GP documentation, 437	X.25 bugs, 413	
graphics, 472	bulletin board	
hardware documentation, 437	Sun Education, 26	
installation, 555	Sun Education, 20	
kernel, 479	C	
library utility, 566		
lint, 388	C	
local 3270 bugs, 406	reference books, 633	
mail utility, 566	C compiler bugs	
manual pages documentation, 450	compilers, 349	
MCP bugs, 407	C shell bugs	
Modula2, 499	bugs, 514	
network, 502	C2	
network configuration, 502	SunOS 4.0 security, 660	
network documentation, 438	canvas	
network general, 506	colormaps, 146	
network library, 502	carrier sense, 114	
network NFS, 502	CGI	
network program, 508	documentation bugs, 429	
network protocol, 511	CGI bugs	
network yellow pages, 513	graphics, 472	
optimizer, 390	CGI documentation	
OSI bugs, 407	bugs, 429	
Pascal documentation, 438	channel	
Pixrects, 478	bugs, 402	
printer utility, 571 program utilities documentation, 440	DDN bugs, 403	
program utility, 572	checksum	
release notes documentation, 443	Ethernet, 96	
sendmail, 576	client	
setup, 556	sample programs, 130	
shell, 514	stream socket, 130	
SNA 3270 bugs, 410		
SunAlis, 518	clock setting leap year, 305	
SunAlis database, 518	± •	
SunAlis documentation, 518	collisions	
SunAlis general, 518	detection of, 115	
SunAlis spreadsheet, 519	color, 139	
SunCore, 474	maps, 140	
SunGKS, 521	colormaps, 36	
SunGKS, 321 SunGKS general, 521	compilers, 348	
SunGKS general, 321 SunGKS library, 521	assembler bugs, 348	
	bugs, 348	
SunINGRES, 491 SunINGRES documentation, 491	C compiler bugs, 349	
SunINGRES documentation, 491 SunINGRES general, 495	debugger bugs, 355	
Sallitores Rousian, 420		

compilers, continued	datacomm, 400
general bugs, 396	BSCRJE bugs, 401
library bugs, 383	bugs, 400
lint bugs, 388	channel bugs, 402
optimizer bugs, 390	DDN bugs, 403
utility bugs, 397	DNI bugs, 403
configuration bugs	documentation bugs, 405
network configuration, 502	local 3270 bugs, 406
configurations	MCP bugs, 407
controllers, 59	OSI bugs, 407
disks, 59	SNA 3270 bugs, 410
Sun-2, 62	vt100tool bugs, 413
Sun-3, 60	X.25, 413
CONSULT-HSPEED	datagrams
high-speed disciplines, 52	fragmentation of, 109
CONSULT-PLOCK	reassembly of, 109
lock process text, 52	daylight savings time
consulting	kernel, 30
device drivers, 51	dbxWorks, 615
Consulting	debugger bugs
real-time specials, 302	compilers, 355
consulting	debugger documentation
specials, 51, 615	bugs, 432
context	debuggers
switching, 622	documentation bugs, 432
controller	demultiplexing
Ethernet, 79	TCP/IP, 93
controllers	dependency tables, 598
combinations with disks, 61, 62	device drivers
disk configurations, 59	
SunOS installation, 63	Consulting Services, 47
conversion	courses, 56 device addresses, 48
color to monochrome, 37	phone support, 51
copy	references, 57
backup daemon, 261	third party, 53
courses	device names
device drivers, 56	SunOS installation, 63
Sun Education, 26	devices
cross compilers, 399	ones present, 163
2.0 announcement, 214	diagnostics, 418
applications, 214	bugs, 418
bugs, 399	-
compatibility, 216	diagrams SunOS 4.0, 666
disk requirements, 216	disk
FORTRAN Cross Compiler bugs, 399	
CSD Consulting	combinations with controllers, 61, 62 determining configurations, 59
device drivers, 51	enlarging procedure, 39
specials, 51	enlarging SunIPC, 39
Customer Software Services, 9	dispatching
customer-training@sun.com	procedures, 15
Sun Education, 26	DMA, 47
	DNI
D	
daemon	bugs, 403
backup copy, 261	documentation, 419
DARPA, 66	beginner's guides documentation bugs, 428
data communications	bugs, 405, 419
BSC3270 bugs, 400	CGI documentation bugs, 429 debugger documentation bugs, 432
data type	editor documentation bugs, 432
void, 315	FORTRAN documentation bugs, 434
	GKS documentation bugs, 435
database bugs	GP documentation bugs, 437

documentation, continued	filesystems, continued
hardware documentation bugs, 437	local, 308
manual pages documentation bugs, 450	NFS-mounted, 308
network documentation bugs, 438	SunOS 4.0 reorganization, 655
Pascal documentation bugs, 438	filetype
program utilities documentation bugs, 440	Hackers' Corner, 679
release notes documentation bugs, 443	formatter utility
system administration documentation bugs, 443	bugs, 563
windows documentation bugs, 419	FORTRAN, 455
documentation bugs	bugs, 455
SunAlis documentation, 518	FORTRAN compiler bugs, 455
SunINGRES, 491	FORTRAN converter bugs, 468
DoD, 66	FORTRAN documentation bugs, 468
domain system	FORTRAN library bugs, 469
Internet, 103	FORTRAN compilers
	bugs, 455
dot dot releases	FORTRAN converter
SunOS 3.5.1, 609	bugs, 468
driver	
bugs, 479	FORTRAN Cross Compiler bugs
drivers	cross compilers, 399
courses, 56	FORTRAN documentation
references, 57	bugs, 434, 468
third party, 53	FORTRAN library
DST, 30	bugs, 469
Australia, 30	fragmentation
Europe, 30	datagrams, 109
rules table, 31	frame buffers
DVMA, 47	with screendump, 37
_	free list, 623
E	ftime, 30
editor documentation	FTP, 86
bugs, 434	•
editor utility	G
bugs, 562	gateway, 66
education	
courses, 26	gateways, 106
SunOS courses, 65	general
Educational Services	bugs, 481
courses, 26	general bugs
EGA	compilers, 396
PC-NFS, 646	network general, 506
	SunGKS, 521
email	SunINGRES, 495
Sun Education, 26	SunView, 545
errata, 605	general utility
Ethernet, 96	bugs, 568
back-to-back packets, 79	gettimeofday,30
buffer, 79	GKS documentation
controller, 79	bugs, 435
header, 96	GMT, 30
mixing thick and thin, 648	GP bugs
throughput, 80, 81	graphics, 478
experiment	GP documentation
devices present, 163	bugs, 437
T7	graphics, 472
F	bugs, 472
f77SunCore Bugs	CGI bugs, 472
graphics, 474	f77SunCore Bugs, 474
filemerge, 665	GP bugs, 478
files	Pixrects bugs, 478
after power failures, 77	SunCore bugs, 474
filesystems, 316	Cattoore case, 1, 1
determining type, 308	

Н	${f L}$
Hackers' Corner	labels
devices present, 163	pedestal, 59
filetype, 679	layering
manual pages, 341	mail, 91
SunView, 261, 341	leap year
VMS tapes, 687	clock setting, 305
vmsbackup, 687	level 1
hardware	network hardware, 123
color frame buffers, 140	level 2
dependency tables, 598	network hardware, 123
hardware documentation bugs, 437	library
headers	shared, 654
IP, 95	library bugs
octets, 91	compilers, 383
overview, 93	network library, 502
history	SunGKS, 521
use of, 78	SunINGRES, 495
hotline	SunView, 536
procedures, 15	library utility
use of, 11	bugs, 566
hotline@sun.COM	lint bugs
reporting bugs, 13	compilers, 388
hotlines	local 3270
world, 7, 198, 290, 603	bugs, 406
<b>.</b>	localtime, 31
I	loop
I/O	memory management, 623
sockets, 126	M
ICMP, 102	mail, 87
images	aliases, 155
converting to monochrome, 37	formats, 157
incompatibility	layering, 91
SunOS 4.0 and Lisp 2.1, 643	pitfalls, 157
installation	routing, 105
SunOS, 63	mail utility_
installation bugs	bugs, 566
system administration, 555 Intercon	make bugs
hotline, 7, 198, 290, 603	SunPro, 522
Internet	make bugs
addresses, 107	bugs, 567
domain system, 103	manual pages
protocols, 85	Hackers' Corner, 341
IP, 85	manual pages documentation
headers, 95	bugs, 450
	manuals
K	proprietary, 50
kernel, 479	maps color, 140
booting specific, 76	YP, 34
bugs, 479	mask
daylight savings time, 30	address, 67
driver bugs, 479	MCP
general bugs, 481	bugs, 407
multi-threaded, 301	memory
single-threaded, 301	enough swap space, 335
syscall bugs, 485 time zones, 29	states, 621
mile 201165, 27	memory management
	and vmstat(8), 620
	MMU

	P
MMU, continued	packets, 96
PDP-11 architecture, 621	back-to-back, 79
registers, 622 Sun3 architecture, 621	pager, 623
	Pascal documentation
modems asynchronous, 626	bugs, 438
modula	pathnames
Sun Modula-2, 629	finding, 316
Modula2, 499	PC-NFS
bugs, 499	backup, 647
monitors	installation, 646
high-resolution, 37	MS-DOS 3.3 and backup, 647
MS-DOS, 39	PCNFS.SYS
3.3 and PC-NFS 2.0, 647	patch, 644
backup, 647	pedestal
***	information, 59
N	Personal AnswerLine, 9
network, 502	Pixrects bugs
bugs, 502	graphics, 478
configuration bugs, 502	plot(IG)
general bugs, 506	using, 617
library bugs, 502 NFS bugs, 502	port number
program bugs, 508	assignment of, 75
protocol bugs, 511	power failures diskless workstations, 77
yellow pages bugs, 513	
network documentation	printer utility bugs, 571
bugs, 438	printing
networks	images, 36
carrier sense, 114	Prism
collision detection, 115	windows, 151
Ethernet theory, 114	procedure
hardware problems, 122	enlarging SunIPC disk, 39
performance of, 118	hotline, 15
Q & A, 124 SunOS 4.0, 655	process priority, 300
SunOS 4.0, 655 SunOS 4.0 security, 656	products
thin Ethernet, 122	release levels, 6, 197, 289, 597
NeWS 1.1, 635	program bugs
NFS, 88	network program, 508
NFS bugs	SunINGRES, 497
network NFS, 502	SunView, 546
NFS servers	program utilities documentation
SunOS 4.0, 655	bugs, 440 program utility
NFS-mounted	bugs, 572
filesystems, 308	PROM monitor
NSCS	using boot, 73
SunOS 4.0 security, 660	proprietary manuals, 50
	protocol bugs
О	network protocol, 511
obsolescence	notivora protecting 5-2-
SunOS 4.0, 661	$\mathbf{R}$
octets	Read This First
TCP/IP headers, 91	purpose, 18
optimizer bugs	real time
compilers, 390	specials, 615
orange book, 660	real-time
OSI	OS requirements for, 299
bugs, 407	SunOS processing, 297
out-of-band data	reassembly
sockets, 126	datagrams, 109

reference	SPARC
C books, 633	with Sys4-3.2, 205
references	specials
device drivers, 57	CSD Consulting, 51
release level	device drivers, 51
SunOS, 17	real-time, 302
release notes documentation	Sun Consulting, 615
	<del>_</del>
bugs, 443	specific kernel
releases	booting, 76
software products, 6, 197, 289, 597	spreadsheet bugs
reporting bugs, 13	SunAlis spreadsheet, 519
routing	STB
mail, 105	duplication of, 8, 199, 291, 604
RTF	subnets
purpose, 18	address mask, 67
Rutgers University, 85	broadcasting, 107
Ratgers Chiversity, 65	definition, 66
C C	enabling, 69
S	Exterior Gateway Protocol, 66
savecore	· · · · · · · · · · · · · · · · · · ·
with swap space, 337	limitations, 68
screendump, 36	subnetting, 66
color windows, 152	Sun Common Lisp 2.1
screenload, 37	SunOS 4.0 incompatibility, 643
	Sun Consulting
security	specials, 615
SunOS 4.0, 660	Sun Education
SunOS 4.0 networking, 656	device driver course, 56
sendmail	
bugs, 576	SunOS courses, 65
server	Sun Modula-2
stream socket, 127	release 2.0, 629
servers	sun!hotline
	reporting bugs, 13
NFS and SunOS 4.0, 655	use of, 11
setup bugs	sun!stb-editor, 8, 155, 199, 255, 291, 335, 604, 679
system administration, 556	sun!sunbugs
shared library, 654	<b>4</b>
shell, 514	reporting bugs, 13
Bourne shell bugs, 514	suncustomer-training
bugs, 514	Sun Education, 26
C shell bugs, 514	SunAlis, 518
shoebox	bugs, 518
disk labels, 60	database bugs, 518
•	documentation bugs, 518
SIGIO, 126	general bugs, 518
signal handlers	spreadsheet bugs, 519
SunOS 4.0, 315	sunbugs@sun.COM
SIGPIPE	reporting bugs, 13
server, 127	SunCGI, 144
SIGQUIT	·
server, 127	SunCore, 146
SIGURG, 126	printing images, 36
	SunCore bugs
SMTP	graphics, 474
application example, 100	SunGKS, 521
SNA 3270	bugs, 521
bugs, 410	general bugs, 521
sockets	library bugs, 521
example programs, 127	
out-of-band data, 126, 133	SunINGRES, 491
programming examples, 126	bugs, 491
servers, 127	documentation bugs, 491
	general bugs, 495
well-known, 97	library bugs, 495
software	program bugs, 497
dependency tables, 598	suninstall, 665

suninstall, continued	system administration, continued
SunOS 4.0, 659	installation bugs, 555
SunIPC	setup bugs, 556
enlarging disk, 39	system administration documentation
SunIPC installation	bugs, 443
SunOS 3.5, 310	System V
SunOS	and SunOS 4.0, 663
and System V, 663	diagrams, 666
determining release of, 17	T
installation, 63	tables
real-time processing, 297	hardware/software dependencies, 598
release 3.5.1, 609	software release levels, 6, 197, 289, 59
release 4.0 architecture, 654	tape drives
release 4.0 diagrams, 666	SunOS installation, 63
release 4.0 filesystems, 655	tapes
release 4.0 report, 653 release 4.0 security, 656, 660	backing up VMS, 687
release Sys4-3.2, 625	TCP, 85
SunOS 3.5	sockets, 130
SunIPC installation, 310	TCP/IP
SunOS 4.0	demultiplexing, 93
Lisp 2.1 incompatibility, 643	references, 110
signal handlers, 315	TELNET, 86
SunPro	textedit
bugs, 522	Hackers' Corner, 341
make bugs, 522	thick Ethernet
SunSimplify, 523	mixing with thin, 648
bugs, 523	thin Ethernet
suntools	mixing with thick, 648
frame buffers, 141	specification, 122
SunTrac, 525	throughput
bugs, 525	Ethernet, 80, 81
SunUNIFY, 526	time zones
bugs, 526	TZ, 29
SunView, 536	uucico, 30
bugs, 536	training
color frame buffers, 142	Sun Education, 26
general bugs, 545	transcript, 558
Hackers' Corner, 261, 341	bugs, 558
library bugs, 536	TZ, 29
program bugs, 546	DST rules table, 31
SunWindow bugs, 551	<b>200</b>
under Sys4-3.2, 205	${f U}$
SunWindow bugs	UDP, 102
SunView, 551	update, 77
SVID	USA-4-SUN
diagrams, 666	use of, 12, 15
swap space	USAC
enlarging, 336	feedback, 10
large memory, 335	utilities, 562
swapping, 623	bugs, 562
switcher(1)	editor bugs, 562
colormaps, 151	formatter bugs, 563
Sys4-3.2, 625	general bugs, 568
announcement, 205	library bugs, 566
binary compatibility, 207	mail bugs, 566
hardware support, 206	make bugs, 567
software configurations, 207	printer bugs, 571
syscall	program bugs, 572
bugs, 485	sendmail bugs, 576
system administration, 555	uucp bugs, 577
bugs, 555	yellow pages, 34

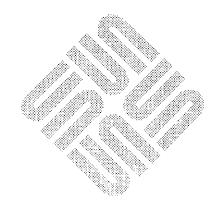
utility bugs compilers, 397 uucico time zones, 30 uucp bugs, 577 VMS tapes Hackers' Corner, 687 vmsbackup Hackers' Corner, 687 vmstat(8) and memory management, 620 void data type, 315 vt100tool bugs, 413 W well-known sockets, 97 windows, 140 color frame buffers, 141 manual pages, 341 Prism, 151 windows documentation bugs, 419 world hotlines, 7, 198, 290, 603  $\mathbf{X}$ X.25 bugs, 413  $\mathbf{Y}$ yellow pages, 32 installation, 33 mail aliases, 155 utilities list, 34 yellow pages bugs network yellow pages, 513 YP, 32 clients, 32 domains, 33 installation, 33 maps, 34 master server, 32 rpc, 34 server maps, 32

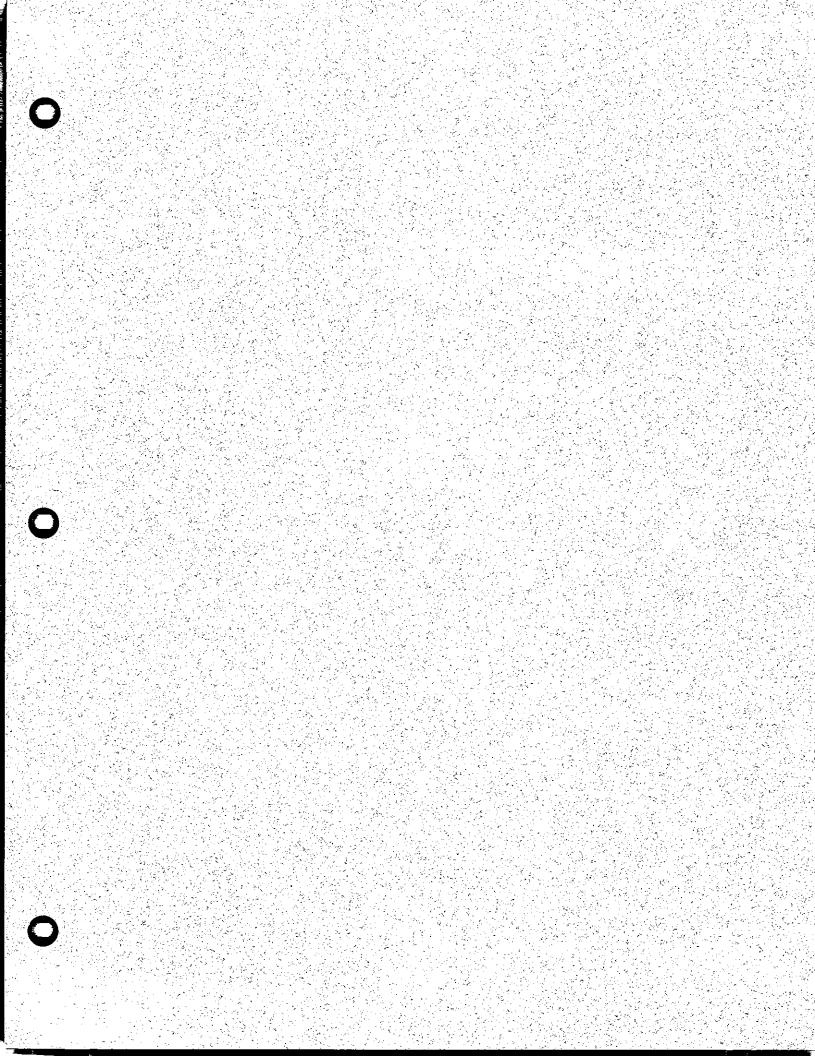
slave servers, 32 utilities list, 34

ypbind, 32 ypserv, 32

## **Revision History**

Revision	Date	Comments
FINAL	April 1988	Fourth issue of the 1988 Software Technical Bulletin, developed by Software Information Services (SIS), Customer Services Division (CSD).





Bulk Rate U.S. Postage PAID Permit No. 515 Mountain View, CA

Corporate Headquarters Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, CA 94043 415 960-1300 TLX 287815

For U.S. Sales Office locations, call: 800 821-4643 In CA: 800 821-4642 European Headquarters Sun Microsystems Europe, Inc. Sun House 31-41 Pembroke Broadway Camberley Surrey GU15 3XD England 0276 62111 TLX 859017 Australia: 61-2-436-4699 Canada: 416 477-6745 France: (1) 46 30 23 24 Germany: (089) 95094-0 Japan: (03) 221-7021 The Netherlands: 02155 24888

UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters: 0276 62111.

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

