sun microsystems

Software Technical Bulletin
September 1989

Technical Information Services

Part Number 812-8909-01
Issue 1989-09
September 1989

Software Technical Bulletin
September 1989

Technical Information Services

Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415-960-1300

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 2-318, 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.

Contents

Section 1 NOTES & COMMENTS 1125
Editor's Notes 1125
TOPS Ordering Information 1127
World Hotlines 1128
Reporting Bugs 1130
STB Duplication 1138
Errata 1139
Section 2 ARTICLES 1143
Sun386*i* SunOs 4.0.2 1143
Name Servers 1153
DOS Windows 1156
Section 3 STB SHORT SUBJECTS 1167
SunOS 4.0 Security 1167
SunOS 4.0.3 Upgrades 1168
Starting `suntools` 1169
SunView Variables 1170
`textedit` Memory 1171
`suntools` Segmentation 1172
Section 4 IN DEPTH 1175
Porting to SunGKS 3.0 1175
Section 5 HINTS AND TIPS 1197
`vi` Hints 1197
FORTRAN Tips 1199
Section 6 THE HACKERS' CORNER 1203
FORTRAN Referencing 1203

Section 7 HARDWARE, CONFIGURATIONS, & UPGRADES 1221
Software Release Levels 1221
Consulting Specials 1225
SCSI Hardware 1233
Type 4 Keyboards 1234
Section 8 CUMULATIVE INDEX: 1989 1239

1

NOTES & COMMENTS

NOTES & COMMENTS **1125**

Editor's Notes 1125
TOPS Ordering Information 1127
World Hotlines 1128
Reporting Bugs 1130
STB Duplication 1138
Errata 1139

1

NOTES & COMMENTS

Editor's Notes

Editor's Notes

The editor's notes for this September 1989 issue include the items of interest listed below.

- TOPS networking product and ordering information
- World hotlines for customer service calls
- World-wide bug reporting information
- Limited permission to duplicate your STB
- Hints and Tips: `vi` hints and FORTRAN optimization tips
- The Hackers' Corner: a FORTRAN Cross-Referencing tutorial
- Configurations: updated software release level tables, effective July 25, 1989

TOPS: Product and Ordering Information

See the note later in this Notes and Comments section containing the address and telephone number to use to get more information on TOPS networking products.

World Hotlines

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

Reporting Bugs World-Wide

A list of Sun service centers, addresses, email hotlines, and telephone hotlines appears. The information in this monthly note continues to be expanded as Sun software service centers are added world-wide.

STB Duplication Permission

This notice is published monthly, giving customers useful information regarding ordering and duplicating additional STB copies. This duplication permission is limited, as detailed in the note.

Hints and Tips

This month's hints and tips section contains two items of interest. The `vi` hints are for those running `vi` applications on a Sun386*i* machine with a 14" or 15" monitor.

The FORTRAN optimization tips offer improved performance for most FORTRAN programs. Examples are provided for SPARC and Sun-3 machines using FORTRAN 1.2, and Sun386*i* machines running FORTRAN 1.1.

The Hackers' Corner

This month's **Hackers' Corner** contains a tutorial in FORTRAN cross-referencing using `lex` and `awk` . An example usage and shell archive code is provided.

For those with email access and wishing an online copy of **Hackers' Corner** code samples, please email *sun!stb-editor* or *stb-editor@sun* with your request. Please include the program title, and the STB issue month and year with your request.

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.

Configurations: Current Sun Software Products and Release Level Tables

The seven tables showing current Sun software product release levels appear monthly. These tables show release levels for operating systems, communications products, unbundled languages, unbundled applications, unbundled graphics, other products, and TOPS networking products. The tables in this issue are updated through July 25, 1989.

Thanks.

The STB Editor

TOPS Ordering Information

TOPS Product and Ordering Information

TOPS networking products are used to link together IBM PCs or compatibles, Apple Macintoshes, and Sun workstations over an Ethernet or AppleTalk network or both.

For TOPS product and ordering information, contact the TOPS sales group directly at the address shown below.

TOPS, a Sun Microsystems Company
950 Marina Village Parkway
Alameda, CA 94501

(415) 769-8700

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.

The world hotlines are divided into those for Canada and the USA, CSD Europe, and Intercon. Intercon includes those countries outside the USA, Canada, Europe, and northern Africa.

Canada and the United States

Canada	Montreal	(514) 738-4885
	Ottawa	(613) 723-8112
	Toronto	(416) 475-6745
	Winnipeg	(204) 222-2333
	Edmonton	(403) 482-7264
	Calgary	(403) 262-6722
	Vancouver	(604) 684-4120
United States	All, including Puerto Rico	1-800-USA-4-SUN

CSD Europe

European Customer Service	Surrey Sun Microsystems Europe Inc.	(44) 276 51440
France	Paris Sun Microsystems France SA	(33) 1 4094 8080
Germany	Munich Sun Microsystems GmbH	(49) 089/46008-321
The Netherlands	Soest Sun Microsystems Nederland BV	(31) 2155 24888
Sweden	Solna Sun Microsystems AB	+46 8 764 78 10
Switzerland	Zurich Sun Microsystems (Schweiz) AG	(41) 1 828 9555
United Kingdom	Albany Park Sun Microsystems UK Ltd	(44) 0276 691052

Intercon

Australia	Sun Microsystems Australia	(011-61-2) 436-4699
Hong Kong	Sun Hong Kong	(011-852-5) 865-1688
Japan	C. Itoh Data Systems Nihon Sun	(011-81-3) 497-4676 (011-81-3) 221-7021
Countries Not Listed	All countries outside the USA, Canada, Europe, northern Africa, Australia, and Japan	(415) 496-6119

Reporting Bugs

Submitting Bugs and Email Service Calls

This article contains two sections for submitting bugs. The first section describes procedures to use within the United States. The second section describes Customer Service Division (CSD) Europe procedures.

Submitting Software Bugs: United States and Canada

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

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

U.S. users holding support contracts can report bugs to USAC via the **(800) USA-4-SUN** phone hotline. Canadian users holding support contracts should call (800) 225-2615. The USAC phone hotline is the fastest way for a customer to find out if a problem is known and if a workaround exists. The status of previously-reported bugs can also be obtained in this way. The list of open software bugs is contained in the Customer Distributed BugsList (CDB).

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

- Whenever possible, customers should use the Online Bugs Database (OBD) described below before submitting bugs, to avoid resubmitting an already-known bug.

- Please note, however, that the alias *onlinebugs-db@sun.com* is **not** the appropriate avenue for submitting bugs.

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

Finally, customers not holding software support contracts may call the **(800) USA-4-SUN** phone hotline to report a problem and request support on a Time and Materials (T&M) basis. Canadian customers should call (800) 225-2615. In this case, please have a Purchase Order (PO) number for billing purposes.

The Online Bugs Database (OBD)

The OBD contains the same information as the Customer Distributed BugsList (CDB). The information available through the OBD is updated during the first week of each month. As a result, you receive the most timely information available on open known bugs and temporary workarounds for Sun software in an easily-accessible, online format.

The OBD service is initially available only within the United States. Future plans include worldwide introduction and distribution.

Information Provided by the OBD

The OBD provides you with rapid telephone access to the following information.

- Software Bug Reference Number--a unique identification number assigned to each valid software bug by Sun
- Online Bug Synopsis--a one-line summary of the software bug
- Bug Description--a brief description of the bug, with examples if available
- Software release(s) in which the bug was reported
- Affected configurations
- Temporary workarounds, where available

To use the OBD, simply dial the telephone number and enter the system password; both provided in the *Online Bugs Database Reference Manual*, part number 812-1001. This manual is automatically sent to the site contact of all Sun customers holding valid support contracts. The OBD is available at all hours, except for scheduled updates and preventive maintenance. System support is available during standard U.S. Answer Center business hours by calling the support numbers given above.

OBD Search Criteria

After logging in, you can quickly search the OBD by any one of the below parameters.

- Software Bug Reference Number
- Software Category (such as kernel, SunINGRES, or Datacomm)
- Software Subcategory (such as documentation related to a specific category)
- Software Release (such as 4.0, 3.5, 3.4, 3.2, 3.0)

Search capabilities can be enhanced by combining several of the primary search parameters. For example, all release 3.4 NFS bugs within the network category

can be searched. In most situations, you can locate a particular software bug and its related workaround within 30 seconds.

To ensure that your OBD use is as efficient as possible, a fast, easy-to-use Help facility is also provided. Help is available throughout your OBD session.

Summary: United States and Canada

For U.S. contract customers, **(800) USA-4-SUN** is the best method to report bugs. Canadian contract customers should report bugs to (800) 225-2615. The electronic mail address *sun!hotline* is available to submit materials that are difficult to relay over the phone. The OBD is available to research currently-known bugs.

For non-contract customers, the electronic mail address *sun!sunbugs* is available to report bugs.

To help us serve you better, please include the following information with all electronic mail reports.

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing
- Please do *not* submit bugs to *sun!onlinebugs-db*

Submitting Software Bugs: CSD Europe

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

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

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

Summary: CSD Europe

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

- Your name

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

Detailed information for European Customer Service and individual countries follows.

European Customer Service

The European Customer Service office is located at the address shown below.

Sun Microsystems Europe, Inc.
Bagshot Manor
Green Lane
BAGSHOT
Surrey GU19 5NL
United Kingdom

Telephone: (44) 276 51440

Telefax: (44) 276 51287

Telex: 859017

France

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

Service "HOT LINE"
SUN Microsystems France
La Boursidiere
R.N. 186
92357 Le Plessis Robinson Cedex

Hotline Telephone: (33) 1 4094 8080

Telefax: 0276 691774

Special Dispatch Arrangements:

Please provide Dispatch with the following items:

System serial number *or* Contract number

Arrangements for Non-Contract Customers:

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

Germany

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

Hotline
Sun Microsystems Gmbh
Stoerungsannahme
Am Hochacker 3
D-8011 Grasbrunn 1
West-Germany

Hotline Telephone: (49) 089/46008-321

Telex: 5 218 197 sun

Telefax: 089/46008-400

Email Address: *{sunuk,unido}!sunmuc!hotline*

Arrangements for Non-Contract Customers:

> Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

The Netherlands

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

Sun Microsystems Nederland BV
Birkstraat 95-97
3768 HD SOEST
The Netherlands

Hotline Telephone: (31) 2155 24888

Arrangements for Non-Contract Customers:

> Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Sweden

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

Sun Microsystems AB
Hemvarnsgatan 9
S 171 54 Solna
Sweden

Hotline Telephone: +46 8 764 78 10

Email Address: *hotline@sunswe.se* or *sunswe!hotline*

Arrangements for Non-Contract Customers:

> Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Switzerland

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

Sun Microsystems (Schweiz) AG
Postfach
Rohrstrasse 36/38
CH-8152 GLATTBRUGG
Switzerland

Hotline Telephone: (41) 1 828 9555

Email Address: *sunuk!sunswis!hotline*

Special Dispatch Arrangements:

Provide Dispatch with the following item:

Contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

United Kingdom

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

Hotline
Sun Microsystems (UK) Ltd
Technical Centre
Unit 3D
Albany Park
Frimley
Surrey
GU15 2PL

Hotline Telephone: (44) 0276 691052

Telefax: 0276 691774

Special Dispatch Arrangements:

Please provide Dispatch with the following items:

System serial number *or* contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Submitting Software Bugs: Intercon

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

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

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

Summary: Intercon

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

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

Detailed information for individual countries follows.

Australia

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

Hotline
Sun Microsystems Australia Pty Ltd
PO Box 320
Artarmon
NSW 2064

Hotline Telephone: (011-61-2) 436-4699

Telefax: 02 436 1084

Special Dispatch Arrangements:

Please provide Dispatch with the following items:

System serial number *or* contract number

Arrangements for Non-Contract Customers:

Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

STB Duplication

Duplicating the STB

Your company's software support contract includes a monthly issue of the STB. Each month, the copy of your STB is mailed to your company's primary contact person or department. 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

Please make the below corrections to articles appearing in three past STB issues.

August 1989 STB

Please disregard the short subject entitled 'SunOS 4.0.3 and SunLink' appearing on page 1054. A corrected short subject entitled 'SunOS Upgrades' appears in this September 1989 STB issue.

June 1989 STB

In the article entitled 'SunOS 4.0.3 Announcement' on page 750, remove 'Sun-4/330' shown as running the `sun4c` kernel architecture. It is among the other Sun-4 machines running the `sun4` kernel architecture.

March 1989 STB

In the article entitled 'LaserWriter II Fonts' on page 374, the code for method 4 shows four backslashes in the `LowAscii` and `HiAschii` strings at the bottom of the page. For the code to run, you must remove the backslashes and use each string as one continuous line.

On page 375, change one line of code as shown below.

Old Line:

```
1 inch 10.5 inch moveto GivenFont cvx str cvs show (  point)) show
```

New Line:

```
1 inch 10.5 inch moveto GivenFont cvx str cvs show ( \(24 point\)) show
```

2

ARTICLES

ARTICLES **1143**

Sun386*i* SunOs 4.0.2 1143
Name Servers 1153
DOS Windows 1156

2

ARTICLES

Sun386*i* SunOs 4.0.2

Sun386*i* SunOS 4.0.2 Announcement

Sun386*i* SunOS 4.0.2 is now available. This article contains an overview of changes included in this new SunOS upgrade, including lists of the bugs fixed.

The main highlights of Sun386*i* SunOS 4.0.2, plus new documentation are listed below.

- Over 400 bug fixes for the entire SunOS, including utilities, DOS, and Ease-of-Use tools
- Faster DOS keyboard, mouse, and AT bus interrupt throughput
- New Owners' Set with fewer release notes
- New 'Cookbook' for Sun386*i* system and network administration

A New Quick-Ship Upgrade Program

Sun is evaluating a quick-ship upgrade program for some customers in addition to the normal Customer Service Division (CSD) channels. This new program is managed by a third-party fulfillment house and consists of a tape or floppy upgrade including the new Owners' Set.

For United States customers, orders are placed using a toll-free number available from your Sales Representative and to be announced in an upcoming STB issue. Payment is done via VISA or Master Card and a Purchase Order Number. Delivery is second day FEDEX.

Due to export and licensing restrictions, European and Intercon customers may obtain their upgrades through normal CSD channels.

Sun386*i* Upgrade Features and Details

The following paragraphs summarize major features and improvements in Sun386*i* SunOS 4.0.2.

- *DOS Keyboard Response*

  Interactive DOS keyboard response is improved by an order of magnitude. Speed of cursor movement in DOS word-processing software exceeds that of users. This improved keyboard response time is seen as screen improvements for DOS text-based applications. For example, scrolling response time for Lotus and WordPerfect applications are significantly improved.

- *AT Bus DOS Interrupts*

  The processing speed for DOS interrupts for AT bus cards is increased. For example, PC LAN file transfer throughput is now three to five times faster. Also, more cards with higher interrupt rates are now supported.

- *New Sun386i Owner's Set*

  The new Owner's Set now includes the *Sun386i Owner's Set Index*, a guide to all Sun386*i* documentation and a complete index of the four manuals in the set. In addition, each of the four manuals in the set contains new information, corrections, and new illustrations.

- *GXi Graphics Accelerator Support*

  Sun386*i* SunOS 4.0.2 contains the microcode and `pixrects` support needed for the new GXi graphics accelerator.

- *International Layered Utilities*

  Sun386*i* SunOS 4.0.2 is the base-level release for the upcoming International Layered Utilities release. Look to future STB issues for details on these utilities.

- *Backup Media and Clusters*

  Because Sun386*i* SunOS 4.0.2 is an upgrade and not a full release, some customers with Sun386*i* SunOS 4.0.1 backup media will not be able to load clusters onto Sun386*i* SunOS 4.0.2 hard disks. Customers receiving new systems have all clusters pre-loaded. Note that all customers must obtain Sun386*i* SunOS 4.0.2 upgrade media for each site.

The New Cookbook

The *Sun386i System and Network Administration Cookbook* is now available as well. The Cookbook is an Sun-internal document and is not part of the standard Sun386*i* Owner's Set, and can be distributed to customers as required. Portions of the cookbook may be included in future revisions of Sun386*i* customer documentation.

The three purposes for this new publication are listed below.

- Inform system administrators and support personnel of the similarities and differences among Sun386*i*, Sun-3, and Sun-4 system and network administration
- Provide instructions to complete specific tasks manually
- Provide 'pointers' to details contained in the other manuals where appropriate.

The book serves both to enhance the understanding and perception of Sun386*i* systems and to offer procedures to make inter-operability among other Sun systems easier.

A sample of some of the topics found in the *Sun386i System and Network Administration Cookbook* include the following:

- Peripheral Administration
- File System Activities
- Adding systems to established Sun-3 and Sun-4 networks
- Disabling Yellow Pages
- Running with multiple YP Domains
- SNAP and ASI: Under the Hood

Bugs Fixed in Sun386*i* SunOS 4.0.2

The following sections include information on bugs fixed in Sun386*i* SunOS 4.0.2, manual page changes, and selected cross references in the case of duplicate bug reference id numbers. The SDR and BugTraq bug reference id numbers and bug synopses are provided when available.

Bugs Fixed

Bugs fixed in Sun386*i* SunOS 4.0.2 are listed on the following pages.

SDR#	product	BugTrac#	Synopsis
1553			
1842	sysex		floppy test failed too frequently
2235			
2393	admobj/ezdb	1009792	does not seem to start when master dead
3054	DOS	1012173	speaker may hand DOS
3055	DOS	1012174	sound quality poor
3227	rpc.pnpd	1012062	first install is the only chance
3329	DOS	1012182	movsw opcode not supported in emulator
3383	DOS	1012183	dos -w hangs terminal
3465	organizer	1012143	drop not active after map
3508	libadmobj	1012104	reports same error for all diskless agnt
3666	rpc.pnpd	1015659	does not verify net address
3842			
3875	/bin/mail	1015672	assumes homedir in the automounted /home
4031			
4123	DOS	1015808	killer serial bugs
4177	DOS	1015810	must support international keyboards
4178	DOS		ISO clipboard
4179	DOS		ISO text file conversation
4180	DOS		ISO filename support
4281	DOS	1015813	drive B attach/detach
4284			
4286	DOS	1015814	physical/virtual drive B
4391			
4435	kernel	1016053	hung mouse
4497	dos	1015822	scrolling displays garbage on 24th line
4533	snap/libadmobj	1015693	lets you add users to diskless systems
4605	admobj/ezdb	1015695	secondary groups not displayed
4608	organizer	1015886	buttons remain active inappr
4610	organizer	1015889	map mode show inappropriate open
4614	organizer	1015893	should indicate no match
4730	libadmobj	1015705	drops original error report
4764	DOS	1015835	PCPAINT/solitaire cursor/mouse misbehave
4805	snap/admobj	1015719	typo error in message
4820	snap/libadmobj	1015721	does not fully remove systms from YP maps
4831	snap	1015729	no directory, home =/ -- chg user group
4833	snap	1015730	restore functions does not return
4838	kernel	015770	GENERIC config file needs extra line
4856	drarpd	1015733	darpd should ignore its own packets
4859	logintool/ezdb	1015735	does not wait for map pushes
4860	bar	1015736	does not open floppy exclusively
4861	kernel	1015313	panic when DOS exits
4880	DOS		cntl/alt keys stuck
4891	DOS	1015772	TTYSW pty wrt fail opr wld blk (Lentec)
4895	sanp/restore	1015375	setuid and setgid not reset (no preserve)
4904	organizer	1015902	copy over child msg needs rew
4911	shutdown		diskless client root cannot ex shutdown
4938	kernel	1015774	wrong error code for bad sector
4940	snap/admobj	1015749	tty modes not restored

4943	snap/libadmobj	1015750	crashes with too many characters
4953	DOS		BIOS equipment flags not updated
4972	drarpd	1015488	cannot have multiple 386i YP domains
4973	Utilities	1015784	replicated mounts
4974	dos		DOS x-devel tools need to be moved to 3rdtools
5004	organizer	1015648	Bkup hangs on '.' files
5013	uid_allocd	1016419	multi YP domains - need uid/gid ranges
5020	snap	1016422	large menues don't display (lg networks)
5022	DOS	1016424	loss of characters (MSWORD 4.0)
5025	zoneinfo	1016426	central Europe timezone != CET (Unkwn)
5041	newfs/instboot		newfs does not install a boot block
5042	newfs		newfs does not set proper disk defaults
5058	EZ	1017298	back-up
5059	EZ	1017299	back-up
5060	EZ	1017300	back-up
5065	organizer		drop button does not respond(*)
5071	kernel	1016505	NMI-SYSEX
5091	kernel	1016507	mv/cp panic across lpback mnts (Docupro)
5091	kernel	1016578	mv/cp fails across loopback mounted filesytems
5095	DOS	1016587	CMOS date/time incorrect
5105	DOS	1016622	bad setup.pc
5106	DOS	1016623	bad setup.pc
5108	kernel	1016640	HP plotter on ttya hangs on last 500 chars
5108	kernel	1016640	plotting
5109	4.0	1015159	panic:dirremove
5114	organizer		doesn't start find in rite di
5115	kernel	1017301	cap lock toggle
5115	kernel	1017301	caps lock does not toggle right (see5703)
5119	DOS		extend has help for /share options
5120	kernel	1016721	corrupted heap (Docupro)
5122	DOS	1016724	16-bit I/O to AT bus
5136	DOS		joystick should be shared
5149	kernel	1016803	4mb will not run
5161	DOS		allow ^Z to background DOS
5162	newkey	1016867	/usr/etc/newkey command is missing from 4.0.1
5162	newkey	1016867	command /usr/etc/newkey missing
5165	dump		change default dump device rmt8 -> rst8
5167	DOS	1016916	DOS Window Crash
5170	kernel	1016919	swiss bank - clk ticks dropped by 82380
5171	ypsync	1016928	tests wrong for YP servers (dupe)
5172	DOS		smtimes gt KIOCSLED: Bad file number MSG
5174	sunvga		MsWindows resizes screen to CGA from VGA lrgscrn
5176	kernel	1016937	hang on file create (Docupro)
5178	DOS		better address decoding
5181	DOS		conirm reboot on attach/detach
5187	Utilities	1016711	ypbind binding on host
5188	Utilities	1016786	yppasswd can be changed
5217	organizer	1017085	move broken in new windows
5226	DOS	1017131	LIM 4 broken w/win 286
5241	kernel		panic when DMA wraps
5249	Utilities	1017302	yp rebind fails or is slow
5255	DOS	1017180	MS Works mouse position wrong
5261	ex/vi		cannot decrypt encrypted text

5264	ex/vi	1017230	ed-x does not decrypt encrypted files
5269	kernel	1017222	byte swap problem in arp table
5293	kernel	1017400	panic:scb overwritten (dbx on "maker")
5298	snap/backup	1017404	spot help on type of backup is wrong
5307	kernel	1017457	general protection fault
5309	DOS		hi-res monitr needs 2x wndow sz (SunFed)
5317	DOS	1017458	killer serial bugs
5324	DOS	1017268	MS Codeview gives 2x mouse response
5340	DOS	1017539	file space available
5343	kernel	1017969	erroneous error messages
5350	kernel		typos in 2 modload / unload error msgs
5352	kernel		signal handler args are wrong (SIGSEGV)
5353	modload		flags option ignored - 9 unit # valid
5360	/bin/mail		requires that YP is running
5367			
5367	kernel		ld.so bug in shared library
5369	DOS	1017660	killer serial Bugs
5370	DOS	1017664	conflicting board memory range
5378	organizer		key 8-bit chars
5396	DOS	1017838	DOS time-of-day wrong (Chem Abstracts)
5398	ftpd	1015111	security enhancement (Xerox)
5400	DOS	1017456	BofA comm.pgn does not rec. COM1
5405	chfn	1017877	chfn creates bogus su entry /etc/passwd
5406	snap/admobj		invald credentials (automatic publickey)
5417	DOS	1017968	timer not monotonic
5427	DOS		dos -w has slow keyboard responsiveness
5441	bar		cannot follow link to root
5446	DOS	1018209	cannot print from vs appl (Fraser Dingman)
5451	kernel	1018235	parallel port BUSY line not monitored
5452	help		crashes goin tween frame and interleaf files
5457	DOS		DMA channel 5 doesn't work
5466	automounter		cd /net/roadman == /etc not / on roadman
5478	organizer	1018355	delete fails on dir
5488	kernel		ptrace crashes the system.
5527	kernel		DOS serial - out of timeslots
5528	kernel		locked nfs /read-a-head (Thorne EMI)
5531	organizer		bad filename hangs system
5532	spot help	1018681	wrong default help path
5534	kernel	1018689	timeout tbl ovrflo,panic(Fraser Dingman)
5538	ASI scripts	1018706	need better existing net msgs (Unkwn)
5539	bar		bar issues debug messages w/Moption
5541	DOS	1018726	bus mouse does not work in msword
5542	modunload		panic rmfree with modunload
5543	kernel		csh unlimit to stack hangs system
5550	DOS		boards.pc file needs fixing/reorganizing
5555	organizer		Sun3/4 fix /usr/bin/syswait: no file/dir
5561	DOS	1018876	redirector does not supp long paths
5562	DOS		syswait fails if SHELL != /bin/csh
5563	in.fingerd	1018877	new version of virus fix patch (unknown)
5567	DOS	1017692	Procomm 2.42 bombs (unknown)
5580	DOS	1017648	VTerm connection lost (unknown)
5586	kernel		keyboard performance for DOS
5586	kernel		notification too slow for keyboard events

5590	DOS		kernel trap handler too slow
5592	login		login -n fjhfgjhfg -p == security enchancement
5594			
5605	sysex		colorfb failure leaves dead mouse
5606			
5606	sysex		more extensive sysex testing
5606	sysex		needs more extensive testing
5607	sysex		cannot do much in expert mode (system hangs)
5608	lint libs		many /usr/lib/lint libs missing
5611	DOS	1017433	long make loses files (unknown) 5306
5612	DOS	1019101	Intel Aboveboard does not boot correctly
5614	DOS	1019124	No VGA support
5626	DOS		Unix2dos/dos2unix bad error msg src/trgt
5627	snap		snap default modem == 2400 baud
5630	snap		crashes on /etc/hosts hostname entry
5641	crontabs		crontab looks for all .nfs files on net
5641	uli		crontab looks for all .nfs files on /net
5643	bar		if bar spans vols with less than 1 block
5648	DOS		falcon hangs DOS window (rpt arrow keys)
5651	DOS		Paradox2 displays a file locking error
5659	organizer		crashes on multiwindow fileselect/move
5660	DOS		BofA prog does not connect to mainframe
5661	DOS		DOS windows move slower
5662	DOS		serial port quickpc settings wrong
5667	DOS		MSword - cannot pick menu cmds w/mouse
5668	DOS		mv kybd cd fm timer.c to kbd.c and kbd_sun.c
5670	DOS		DOS filesharing compatability mode bad
5684	kernel		dos hangs on OLD type 4 keyboards
5688	DOS		no cursor on external monitor cards
5694	pixrect		SV apps using pr_region dsply incorrect
5695	DOS		8 bit filenames can't be read by DOS
5696	dos		paste to dos changes char (Itl keyboard)
5699	kernel		mouse tracking poor on external VGA
5701	terminfo	1019870	termcap and terminfo entries different
5702	DOS		cannot install portuguese keybd
5703	kernel		(dupe5115) caps lck key does not toggle correctly
5710	organizer		if no YP hosts.byname map, /net hangs
5730	DOS		cannot type on some keyboards
5730	dos		cannot type on some keyboards
5731	DOS		MS Flight Simulator runs very slow
5741	organizer		crashes when renaming long file name
5745	dos		Swedish keyboard -- incorrect characters display
5746	DOS		autorepeat ON forever with accent chars
5753	pixrect		screen display diff from normal FBs
5759			
5760	kernel		win sys debug msg on console
5765	format		format dumps core on NON-RR disks
5765	uli		format core dumps formatting nonCDC SCSI disks
5769	organizer		find crashes if started from /home dir
5777	snap		does not grey right stuff if no YP running
5795	rpc.pnpd		YP update fails w/o diag msg in syslog
5796	/usr/etc/client		coredumps on authdes_create
5800	dos		ger kybd - wrong chars and strange autorepeat

```
5806    kernel                      ws_dispense.c w/pid=0 crashes
5808                      1020406   sh Bourne shell can corrupt mem
5810    dos                         vi_div0: too many opcode prfxs (FALCON hangs)
5828    dos                         dos 7 window smoke test crashes windows
5835    dos               1020396   (dupe4497)GW_BASIC scrolls screen 1 wrong
5851    org                         bl10e organizer gags on filenames w/spaces
5857    dos                         (dupe5810) strting flight sim in VGA window dies
5899    dos                         invoking w/batch file on invalid drv should exit
5905    uli                         mail delverd /var/spool instead of  /mail/inbox
5907    uli                         cd /net/thorin -- permission denied
5914    dos                         remove 'type conflict' warning msg in screed.c
5915    org                         no organizer frame spot help
5921    bar                         bar sees zero bytes read/written @ eov on tape
5922    bar                         does not use $AUTO_FIXNAMES
5923
5927
5933
5939    dos                         remove call to set_cmos_ega (performance)
5966
5967 5975 5984 5993 6007 6010 6011 6014 6018 6030 6034 6039 6040 6048 6053 6056
6060 6066 6072 6075 6087 6088 6089 6090 6092 6094 6096 6098 6099 6101 6101 6107
6108 6109 6113 6117 6118 6119 6122 6125 6131 6133 6134 6139 6139 6141 6142 6144
6144 6148 6148 6149 6151 6152 6156 6170 6170 6173 6173 6175 6176 6176 6177 6186
6188 6191 6192 6196 6198 6202 6203 6205 6208 6212 6220 6223 6241 6244 6248 6252
6253 6257 6264 6268 6274 6275 6283 6287 6288
```

Bugs Fixed: Manual Pages

The changes shown on the following pages have been made to the manual pages. Fixes are listed by SDR numbers.

SDR#	Manual Pages Changed
2449	mknod.2, open.2
3061	mem.4S
3065	screenblank.1
3313	dos2unix.1, unix2dos.1
3371	getty.8
3747	boards.pc.5, setup.pc.5
3769	catman.8, man.1, whatis.1
3821	user_agentd.8, ypbatchupd.8C
4299	ethers.5, hosts.5
4561	load(1) needs updating
4575	kb(4) incinsistant arg naming
4615	screenblank.1
4848	unload.1
4925	drand48.3
4941	sysex.1
5035	getdoinname.2
5077	dd.1
5084	nm.1
5179	mmap.2
5328	pwd.1, getwd.3, automount.8
5344	calendar.1
5354	setttyent.3
5490	dos2unix.1, unix2dos.1
5616	ftpd.8C
5669	sysex.1
5801	ls.1
5850	getpriority.2
5852	nice.1
5853	csh.1
5854	diffmk.1
5855	foption.1
5856	renice.8
5858	change_login.8
5859	copy_home.8
5860	ext_ports.5
5861	ypaliases.5
5862	yppasswd.5
5863	ypgroup.5
5864	ypprintcap.5
5865	nice.3C
5866	user_agentd.8
5879	systems.5
5880	login.1
5881	screendump.1
5882	troff.1
5883	vprintf.3s
5884	backup.5
5885	ttytab.5
5886	dump.8

```
5887            fsck.8
```

Duplicate Bug Reference ID Number Cross-References

A list of known duplicate bug reference ID numbers with cross-references appears below.

```
4305 is a duplication of 4116 and 4689
4316 is a duplication of 4116 and 4689
4692 is a duplication of 1985
4856 is a duplication of 4972
5051 is a duplication of 4435
5089 is a duplication of 4859
5310 is a duplication of 4929
5322 is a duplication of 4116 and 4689
5331 is a duplication of 4116 and 4689
5385 is a duplication of 4833
5524 is a duplication of 4907
5640 is a duplication of 5592
5642 is a duplication of 4833
5703 is a duplication of 5115
5711 is a duplication of 5451
5835 is a duplication of 4497
5857 is a duplication of 5810
```

Name Servers

Name Servers Defined

This articles contains two sections. The first is an overview and general description of name servers, their uses, and advantages. The second section is a series of frequently asked questions about name servers and the answers.

Name Servers: An Overview

A name server is a network service that enables clients to name resources or objects and share this information with other objects in the network. This is, in effect, a distributed database system for objects in a computer network. It is used primarily to forward, resolve, and delegate host name lookup throughout interconnected domains on the INTERNET.

The advantage of using a name server over the host table lookup for host name resolution is to avoid the need for a centralized clearing house for all names. Through use of the name service, the authority for this information can be delegated to different responsible organizations on the network. For example, our Sun machine connected to the INTERNET is responsible for resolving all inquiries to our *sun.com* domain. It is also responsible for communicating *sun.com*–generated name queries for other, outward domains to the responsible name server for that domain path.

Sun's names service implementation was not intented for full implementation and support until SunOS 4.0 and subsequent releases. The Sun Network Software Development group, however, has made appropriate files available for SunOS 3.x implementation via an available 'Name Server Kit'. However, as noted in the kit's `README`, use of the Name Server Kit for SunOS 3.x implementation is considered a do-it-yourself operation and is not supported or maintained. The Name Server Kit is available via the Sun customer support channels. We recommend the customer upgrade to SunOS 4.x for full support.

See pages 537 through 552, 'Name Server Operations', chapter 22 of the *System & Network Administration* manual, part number 800-1733, of the SunOS 4.0 documentation set. It is recommended that the customer read this chapter before attempting to install and operate the name service.

Name Servers: Questions and Answers

The following comments apply to those sunning SunOS 4.x on Sun-2, Sun-3, and Sun-4 machines.

1. What are the host lookup options available under SunOS?

   Host lookups under SunOS may be configured in one of two ways. Either the Domain Name Service can be used for all hostname and address lookups, or YP can be used to look up hostnames and addresses initially with DNS lookup when the YP lookup failed. To configure hostname and address lookups to occur through the DNS exclusively you must follow the following steps:

A. All clients must have their `libc.so` file replaced with the equivalent version that contains the resolver versions of `gethostbyname()` and `gethostbyaddr()`.

B. Replace the statically linked executables with their resolver based equivalents.

C. The file `/etc/resolv.conf` must be set up on all machines to point at a system that is running a name server.

2. How do I link YP to the domain name resolver?

To configure the host and address lookups to occur through YP and then through DNS (the preferred method) the following steps must be followed:

A. The two YP maps `hosts.byname` and `hosts.byaddr` must have the `YP_INTERDOMAIN` key set in them. This is done by using the `-b` flag with `makedbm` when creating them. You will need to modify your `yp Makefile` to do this.

B. The YP master server must be a SunOS 4.x system so that it will preserve this flag.

C. All SunOS 3.x slave ypservers must be running with the `-i` flag set with `ypserv`.

D. There should be an `/etc/resolv.conf` file that points to a valid nameserver on all YP servers.

Also, under SunOS 4.0 and 4.0.1 the resolver link will not work even after rebuilding the host maps. This is due to recent bugs discovered (bug reference id 1011577) in both `ypserv` and `ypxfr` . Fixed versions of `ypserv` and `ypxfr` are available from your local service center and are included in SunOS 4.0.3.

3. How do I link YP and name lookup to a remote machine running the name server?

Set up an `/etc/resolv.conf` file. See the *resolve.conf(5)* manual page for details. Note that the correct file name is `resolv.conf` with no 'e'. You will specify the Internet address of the remote server in the `resolv.conf` file.

4. How can I test the name server link?

You can test the link in two ways. After setting up the `/etc/resolv.conf` file appropriately do either of the following.

A. Use *nslookup(8C)* to test the link. The `nslookup` utility is linked directly to the resolver and does not use the YP access. A successful

name resolve using `nslookup` will verify that the `resolv.conf` file and the resolver communication to the name server are working. See the example shown below.

```
% nslookup rice.edu          #should return address for "rice.edu"
```

B. After making the YP hosts maps with the `makedbm -b` command *and* installing the appropriate `ypserv` and `ypxfr` fixes (or SunOS 4.0.3), you should be able to start YP, verify you are bound correctly to a YP domain server, and run a host name lookup using `ypmatch` . An example follows.

```
% ypmatch rice.edu hosts     #should also return address for "rice.edu"
```

5. How should I specify a domain name in a network which is running named, and in which the global domain name for mail is not included in the YP domain name?

   Under SunOS 4.0, domain name specification in its default setting is controlled by the *domainname(1)* system command. This results in starting at the first 'dot' in the `sendmail.cf` and `resolv.conf` files.

   For example, if the local host name is *aurelius* and the domain name is set as *usac.sun.com*, then fully qualified it becomes *aurelius.sun.com*. Note that *usac* is removed. If the domain name is set to *.usac.sun.com*, the fully qualified path becomes *aurelius.usac.sun.com* .

   Note that a domain name starting with a 'dot' (.) or a plus sign (+) causes the first component to be retained. If you wish this domain name setting (used by YP) to be different from the domain name used in mail headers and with the domain resolver, you can specify the desired domain name with the `Dm` macro in the `sendmail.cf` file. See the comments in that file. You can also use the `domain` option in the `/etc/resolv.conf` file.

6. Why are some of our machines responding without domain names while others respond with double domain names?

   Check domain name settings on these machines. For subsidiary mail machines ensure that their `sendmail.cf` files have the setting shown below.

   ```
   # my official hostname
   Dj$w.$m
   ```

   Note that `$m` is the appended default domain name. It is set by the `Dm` macro in this same file if the `Dm` macro is used.

DOS Windows

DOS Windows Release 1.0 Announcement

This article announces the availability of DOS Windows release 1.0 for Sun-3 and Sun-4 systems, running Sun Operating System (SunOS) release 4.0, and with a minimum of 8 Mb of memory.

Introduction to DOS Windows

DOS Windows allows users to run IBM PC/AT application software on a Sun workstation. The best way to think of DOS Windows is as a platform for running IBM PC/AT software. Below the platform, the SunOS system software performs its normal functions of operating the workstation. Above the platform, IBM PC/AT application software runs as it would on a personal computer. Between these software layers, DOS Windows maps resources that SunOS and the Sun workstation provide to what the IBM PC/AT application expects. With DOS Windows, IBM PC/AT productivity software like spreadsheets, database managers, and word processors can be installed and run on the Sun workstation. DOS Windows thus allows the user to combine the benefits of multitasking, networking, and flexibility of the Sun workstation with the variety of familiar IBM PC/AT application software.

Related Sun Products

The following Sun products can be used to integrate Sun workstations with the DOS environment:

- The *SunIPC* is a VME-based hardware coprocessor that provides a SunView1 window for emulating an IBM-PC. The SunView1-based user interface is similar to DOS Windows.
- The Sun386*i* is a Sun workstation based on the Intel 80386 CPU. It runs standard SunOS software including SunView1-based applications like *mailtool(1)* and *textedit(1)*, as well as a few applications unique to the Sun386*i*, like an on-line help viewer and a file organizer. As with the SunIPC, the SunView1-based user interface of DOS Windows for the Sun386*i* is similar to DOS Windows for the Sun-3 and Sun-4.
- *PC-NFS* is an implementation of Sun's Network File System (NFS) for MS-DOS machines connected to an Ethernet local area network.
- *TOPS* is a local area network for sharing files between IBM PCs and compatibles, Apple Macintoshes, and Sun workstations on an AppleTalk network.

Using DOS Windows

Once DOS Windows is installed on the Sun workstation, it can be started like any other SunView1 program; that is, the user can either type **dos** from the command line, or set up a line in the `~/.rootmenu` file.

The DOS Window window is organized like most SunView1 applications. The top of the frame is a name stripe that indicates the name of the application and the names of available DOS devices (for example, `drive C:`, `COM1:`, and so on). Two different menus are available with DOS Windows: the `Frame` menu and the DOS Windows menu. The `Frame` menu has the same selections as the SunOS 4.0 and higher menus; that is, `Close`, `Move`, `Resize`, `Front`, `Back`, `Props`, `Redisplay`, and `Quit`. If `Close` is selected, DOS Windows will close into an icon on the desktop. This also causes DOS Windows to pause execution of any task it may be performing at the time the window is closed.

When DOS Windows starts up, the user sees the familiar DOS prompt. At this point, DOS commands such as `type` and `dir` can be entered.

Communicating with DOS Windows

Interaction with DOS Windows can be considered from the perspective of either a Sun workstation user or a PC user, summarized as follows.

As a Sun Workstation User

If you are a Sun workstation user, working with DOS Windows is much like working with any other SunView1 window. The Sun mouse can be used to pop up the `Frame` menu to open, close, or quit DOS Windows. In addition, you can pop up the DOS Windows menu to perform common DOS Windows operations, such as the following:

- Copy and paste
- Change display adapter emulation
- Attach and detach the PC mouse
- Send data to a printer
- Use the DOS Windows keyboard speaker
- Control access to emulated DOS devises
- Exit DOS Windows

As a PC User

If you are a PC user, DOS Windows provides the following features:

- (Logical) hard disk
- Microsoft bus mouse and driver emulation
- IBM PC/AT keyboard emulation
- Multiple display adapter emulation
- Access to PC peripherals through serial ports

- Lotus-Intel-Microsoft (LIM) expanded memory standard (EMS)
- Direct access to the SunOS file system from MS-DOS

Disk Drives

DOS Windows works with three kinds of disk drives: a floppy disk drive, a logical hard disk drive, and a redirected disk drive. If an optional disk drive is installed on your Sun workstation, DOS Windows will access that disk drive as drive `A:`. Logical hard disk drives are SunOS files that emulate a hard drive on an IBM PC. They are useful for holding MS-DOS system files and applications whose file protection schemes require drive `C:`. When DOS Windows is first installed, each user has a single logical disk `C:` for holding MS-DOS system files. Redirected disk drives are SunOS directories that are available to DOS Windows as separate drives. DOS Windows manages the connections between the SunOS file system and the MS-DOS file system.

Redirector

One of the key features of DOS Windows is the *File Redirector*. This is a facility that allows DOS Windows to access the SunOS file system as if it were an MS-DOS file system. The redirector allows SunOS directories to be extended or assigned to virtual DOS drives. By using the redirector with Sun NFS, the user can gain access to a virtually unlimited number of files from within PC applications on DOS Windows.

The redirector software consists of two programs, as follows:

- `redir` is a small memory-resident program that initializes the file redirection software, and is run from `autoexec.bat`. It takes no arguments, and must be run before `extend`.
- `extend` is the user program for the redirector. It handles the management of redirection for specific directories.

The default `autoexec.bat` file for DOS Windows redirects four SunOS directory, as follows:

- `R:` is set to the root directory
- `H:` is set to the user's home directory
- `N:` is set to `$DOSDIR` (by default `/etc/dos`)
- `E:` is rooted at the root, and starts from the directory where DOS Windows is started

Logical Hard Disks

DOS Windows can have as many as two logical hard disks. A logical hard disk is a SunOS file with a file format that internally emulates the structure of an MS-DOS disk drive. You can use MS-DOS commands on a logical hard disk to perform standard operations, such as checking the available disk space and copying files from a floppy disk.

The size of this logical hard disk can be reset after DOS Windows is installed. A second logical hard disk can be created as drive `D:` if additional storage is desired. It is recommended, however, that the redirector facility be used to store DOS Windows files in SunOS directories, and to only use a logical hard disk to hold system files and applications that require it.

The logical hard disk format used by DOS Windows for the Sun-3 and Sun-4 is compatible with those used by the Sun386*i* and the SunIPC. Users can read and write files between logical hard disks on these different systems.

Floppy Disk Drives

DOS Windows supports optional floppy disk drives as drive `A:` and drive `B:`. These are available from third party suppliers. Keep in mind that floppy drive support may require the installation of a device driver in the SunOS kernel.

Peripherals

DOS Windows emulates several PC peripherals, such as the mouse, the AT keyboard, and the serial ports, as described below.

MS-DOS Mouse Emulation

For applications requiring a mouse, DOS Windows emulates a Microsoft Bus Mouse, and also has an MS-DOS driver that provides a Microsoft-compatible interface. This allows users to run PC applications that use a mouse. By default, the Sun mouse is reserved for SunView1, and any mouse movement or mouse button clicks are interpreted by SunView1. The `Mouse` submenu on the DOS Windows menu allows the user to toggle between having the mouse dedicated to SunView1 and MS-DOS emulation. When the mouse is in DOS mode, the cursor will not be visible unless an application is running that uses it. The right mouse button will always display the DOS Windows menu and allow the user to return the mouse to SunView1. Quitting DOS Windows will also return the mouse to SunView1.

IBM PC/AT Emulation

When working with DOS Windows, the Sun keyboard emulates an IBM PC/AT 84-key keyboard. This is possible through keyboard mapping, described in in detail in the *DOS Windows Sun-3 and Sun-4 User's Guide*, part number 800-3249-10.

Multiple Display Adapters

DOS Windows emulates three types of display adapters used by PC applications, as follows:

- IBM monochrome adapter: 80 x 25 characters
- IBM color graphics adapter: 80 x 25 characters; 640 x 200 pixels
- Hercules adapter: 80 x 25 characters; 720 x 348 pixels

PC Peripherals

The Sun workstation serial and parallel ports can be used to connect PC peripherals for use with DOS Windows.

EMS Memory

DOS Windows supports the EMS memory expansion standard (EMS), which means that DOS Windows can handle large spreadsheets, such as might be used

with Lotus 1-2-3 applications. Through this memory expansion standard, DOS Windows can provide up to 8 Mb of additional memory for PC applications. DOS Windows supports both common versions of the EMS specification. `pemm.exe` supports the new EMS 4.0 specification, and is upwardly compatible with EMS 3.2.

Differences Between DOS Windows and the IBM PC/AT

DOS Windows differs from the IBM PC/AT in a few ways. Specifically, DOS Windows does not do the following:

- Support peripherals over an AT bus. DOS Windows does support the use of peripherals through serial ports on a Sun workstation, and physical printers associated with the logical device names `LPT1`, `LPT2`, and `LPT3`.
- Support for all IBM PC/AT display adapters. DOS Windows only supports the monochrome, CGA, and Hercules display adapters. Application software that depends on the presence of any other display adapter cannot run on DOS Windows.
- No multi-tone speaker. DOS Windows uses the keyboard speaker in a Sun workstation to emulate the IBM PC/AT speaker. However, since the Sun keyboard speaker can only emit a single tone, DOS Windows cannot play tunes or duplicate the varied frequencies produced by some PC applications. In these cases, the Sun keyboard speaker plays a single tone.

Performance Issues

DOS Windows consumes much of the memory and CPU resources in a Sun workstation. Therefore, when DOS Windows is not in use, the user should select the `Pause` item from the `System` menu, or close the window into an icon. To use DOS Windows again, select `Run` from the `System` menu.

This does not cause too much of a problem in practice, because DOS Windows usually detects when it it is not being used, and stops consuming system resources until either the mouse or keyboard are used. Thus, DOS Windows will have a brief delay while it restarts.

Differences Between DOS Windows and Other SunView1 Windows

DOS Windows differs from most other SunView1 windows in one principal way: users cannot scale the text and graphics in the window used by DOS Windows by resizing. DOS Windows displays an amount of text and graphics that is determined by the display adapter that DOS Windows is emulating. The area that DOS Windows uses to display text and graphics is fixed. When `Resize` is selected from the `Frame` menu to enlarge DOS Windows, it will use a portion of the window for display and leave the rest blank. Thus, the DOS Window and the information displayed therein cannot be resized.

Installing PC Applications

As most PC users know, software vendors sometimes implement various copy protection schemes that prevent unauthorized copying of PC application programs. When an independent PC is used, the user can always run a PC application from the distribution floppy disk, or a backup copy of that disk.

Additionally, depending on the copy protection scheme, the application can be installed on the PC's hard disk.

When using DOS Windows, installing and running PC applications can become more complicated, as follows:

- The Sun workstation might not have an associated floppy disk subsystem.
- Some copy protection schemes prevent the user from installing certain PC applications on any type of hard disk. In this case, if the user does not have a floppy disk subsystem, the application cannot be run.
- Although some copy protection schemes do allow the user to install PC applications on hard disks, these installation procedures might not work with the Redirector. This is because the Redirector uses the SunOS file structure, and may not match the structure expected by these copy protection schemes. In this case, install the software on the DOS Windows logical hard disk.

If No Floppy Disk Drive is Available

If there is no floppy disk drive available for the Sun workstation, one of the following methods can be used to access the PC application:

- Use the Redirector to gain access to the file system where the files are located.
- Install the application on a SunOS file system, and mount the appropriate file system. This can be done with a PC running PC-NFS, with a Sun386*i*, or with a SunIPC board with an attached floppy subsystem.
- Install the applications on the associated DOS Windows logical hard disk.

Hard Disk Copy Protection

Some copy protection schemes do not allow users to install a given application on any type of hard disk. In this case, none of the above options will work. Refer to the installation documentation of the PC application software to be installed for such information prior to any installation.

> **NOTE:** Keep in mind that if an attempt is made to install an application on a logical or physical hard disk when the application's copy protection scheme does not permit such installation, there is a risk of damaging the application distribution disks.

Using Key Disks

Some copy protection schemes allow the user to install a given PC application on a hard disk, but require that a *key disk* be used when running the application. The key disk is a floppy disk whose presence guarantees that only one user at a time can use the application software. In general, such applications with this kind of

copy protection can be installed on either a DOS Windows logical hard disk or on a redirected drive.

Reading License Agreements

It is recommended that users carefully read the license agreements for each PC application before an attempt is made to install it on a DOS Windows logical disk or SunOS file system. If there is any question as to whether it is legal for the user to install the application, check with the software vendor.

Loading an MS-DOS Application Using PC-NFS

Assuming that the user has an IBM PC-compatible computer running PC-NFS with a qualified Ethernet board (refer to *Installing PC-NFS: A Guide to the User and System Administrator*), loading an MS-DOS Application onto the Sun workstation is straightforward.

First, mount the SunOS file system where the MS-DOS software is to reside. For example, to mount to a directory called `/usr/dos` on a workstation named `system` onto drive `Q:`, the following command could be used:

```
C:> net use Q: \\\\system\\usr\\dos
```

For more information on using the `net use` command, refer to the *PC-NFS User's Manual.*

After the SunOS file system has been mounted with PC-NFS, the MS-DOS `copy` command can be used to copy files from the PC to the Sun workstation for DOS windows, as follows:

```
C:> copy a:*.* Q:
```

This copies files from the floppy disk loaded on the PC into a SunOS directory on the Sun workstation.

Once the MS-DOS software is loaded into a directory on the Sun workstation, it can be installed in the same way as used on an IBM PC-compatible computer. In the simplest case, all that is needed is to make the SunOS file system into a DOS drive with the `extend` command, and add the new drive and directory to your DOS path. This information will probably be included in the `AUTOEXEC.BAT` file, as follows:

```
path m:;c:
```

Larger applications generally require more complicated installation procedures. For example, a spreadsheet or database application might have an entire installation suite. These procedures usually work unchanged in DOS Windows.

File Transfer with Serial Lines

If no access is available to PC-NFS, a Sun386*i*, or a SunIPC board with a floppy drive, you may be able to transfer files with a modem and a serial line. Since telephone lines are much slower than Ethernet, this could be a very slow procedure.

To perform such a file transfer, a communications program on an IBM PC-compatible computer is needed to control communications. Establish a connection with a Sun workstation attached to a serial line, and log in to SunOS. Start up a related file transfer program on the Sun workstation, and transfer files. There are usually two methods that can be employed to transfer files: one for text and one for binary files. The user should be able to use binary transfer for MS-DOS executable files.

Running PC Applications

When working with DOS Windows, PC application software can be run using three different methods, as follows:

- Install the application in a SunOS directory, then use the Redirector to make that directory available to DOS Windows. This is the preferred method.
- Run the application directly from a floppy disk inserted into the optional DOS Windows floppy disk subsystem.
- Install the application on the DOS Windows logical hard disk and run the software from that location.

Controlling DOS Windows

There are four ways to control the facilities of DOS Windows through SunView1 and SunOS, and they behave according to the following hierarchy:

- DOS Windows menu
- DOS Windows command line
- DOS Windows environment variables
- DOS Windows configuration file `setup.pcs`

From MS-DOS, the `AUTOEXEC.BAT` and `CONFIG.SYS` files can be edited to modify the characteristics of DOS Windows.

Moving Information Between SunOS and DOS Windows

There are two basic methods for moving information between SunOS and DOS Windows: the SunView1 cut-and-paste facility, and the Redirector.

The SunView1 cut-and-paste facility can be used to transfer portions of text between DOS Windows and other SunView1 applications. This is recommended for moving small amounts of text between DOS Windows and other SunView1 applications.

The Redirector can be used to gain access to SunOS files from within applications running on DOS Windows. Files being transferred between the two must be translated to convert the line breaks used within each environment. DOS Windows has two programs, *dos2unix(1)* and *unix2dos(1)*, that are used to translate text files between the two environments.

DOS Windows Usage Considerations

The following usage considerations will be of importance to DOS Windows users.

Swap Space Requirements

DOS Windows consumes a fair amount of swap space. Each instance of DOS Windows takes between 6 and 8 Mb of swap space. Refer to the *Software READ THIS FIRST for DOS Windows for the Sun-3 and Sun-4*, part number 800-3345-05, for instructions on increasing swap space.

Logical Disk Drives for Booting DOS Products

Logical disk drives are not compatible for booting different DOS products. For example, a `C:` drive used for the Sun386*i* cannot be used for booting DOS Windows or a SunIPC board. These logical drives are structurally the same, so they can be mounted as the `d:` drive to permit file usage.

GW-BASIC

GW-BASIC is not provided with the DOS Windows software; however, the *Sun GW-BASIC User's Manual*, part number 814-1030-10, is included with the DOS Windows Release 1.0 documentation as a convenience to the DOS Windows user.

DOS Windows 1.0 Documentation

The following documentation is included in the DOS Windows 1.0 Manual Set:

- *Software READ THIS FIRST for DOS Windows for the Sun-3 and Sun-4*, part number 800-3345-10
- *DOS Windows Sun-3 and Sun-4 User's Guide*, part number 800-3249-10
- *Sun MS-DOS Reference*, part number 800-1008-10
- *Sun GW-BASIC User's Manual*, part number 814-1030-10

3

STB SHORT SUBJECTS

STB SHORT SUBJECTS **1167**

SunOS 4.0 Security 1167
SunOS 4.0.3 Upgrades 1168
Starting `suntools` 1169
SunView Variables 1170
`textedit` Memory 1171
`suntools` Segmentation 1172

3

STB SHORT SUBJECTS

SunOS 4.0 Security

Enhancing Your SunOS Security

Those customers wishing to enhance their SunOS security against malicious use of *rwall(1C)* and *wall(1)* will find the below workaround helpful.

This workaround corrects bug reference id 1021702 and immediately enhances the OS security. You must be root when making the below changes.

```
# cd /bin
# ls -lg wall
-rwxr-sr-x  1 root     tty          5168 Apr 24 18:23 wall*
# chown nobody wall
# chmod u+s wall
# ls -lg wall
-rwsr-sr-x  1 nobody   tty          5168 Apr 24 18:23 wall*
#
```

On systems running SunOS 4.x.x, also edit `/etc/inetd.conf` by changing one line as shown below.

Old Line:

```
walld/1        dgram   rpc/udp wait root /usr/etc/rpc.rwalld rpc.rwalld
```

New Line:

```
walld/1        dgram   rpc/udp wait nobody /usr/etc/rpc.rwalld rpc.rwalld
```

SunOS 4.0.3 Upgrades

SunOS 4.0.3 Upgrades: A Problem and Solution

NOTE:

This short subject is a correction to a short subject appearing on page page 1054 the August 1989 STB entitled 'SunOS 4.0.3 and SunLink'. Please disregard the previous article.

Many customers will be upgrading to SunOS 4.0.3 during the upcoming weeks and months. They may be using *sunupgrade(8)* to do this.

Customers who have placed files or directories in `/export/exec` which contain the string `'sun'` may see error messages resulting from a known bug, reference id number 1022502. `sunupgrade` may erroneously attempt to install these files or directories as though they were client architectures.

sunupgrade(8): The Problem

The *sunupgrade(8)* utility treats all filenames, with three minor exceptions, of the form `/export/exec/*sun*` as though they are 'implementation architecture types'.

The Workaround

Customers with such files or diectories in `/export/exec` should temporarily rename all `/export/exec/*sun*` file or directory names *before* running *sunupgrade(8)*. Of course, this does not include the files for `sun2`, `sun3`, `sun3x`, `sun4`, or `sun4c`.

The temporary name must be chosen such that it no longer contains the string `'sun'`. Upper case `'SUN'` will work. When the upgrade has completed, restore the original file or directory names.

Aborting *sunupgrade(8)*

If the files were not renamed prior to beginning the upgrade, `sunupgrade` may be aborted with a Control-C at the time the bogus prompt for a tape is issued.

You then need to run the 'mop-up' script (`/usr/etc/upgrade/mop_up`). This should work in most cases and does seem to work in the case of `sunlink` . Note, however, if the bogus 'architecture type' name sorts out before any of the legitimate 'architecture types', upgrades for those architecture types will still need to be performed.

Starting `suntools`

SunOS 4.0 and Starting `suntools`

Some customers running SunOS 4.0 receive the below error message when starting `suntools`.

```
open: permission denied
```

The Problem Defined

This problem is usually caused by wrong permissions on the `/var/tmp` directory. To test this, start `suntools` as root. If you do not get the error message as root, but do get it as any other user, then it is the permissions problem.

If you use the `ls -la` command on the `/var/tmp` directory, you should see the results below.

```
drwxrwsrwx  2 bin          512 Mar 20 09:29 ./
drwxr-sr-x  9 bin          512 Jan 16 15:05 ../
-r--r--r--  1 hvr        73728 Mar 11 14:22 vm_fonts-n0
```

Possible Solutions

Make sure that the 'dot' directory (.) has permissions of 777, and that the `vm_fonts*` file has permissions of 444 as shown above. Note that this file is used for virtual memory fonts.

This error can also be caused by accidently changing the protection by using the interactive mode of `restore` and specifying that the user and mode be set.

Another similar problem is caused by misspelling the name of a default font or root background icon, or not including the correct path.

SunView Variables

SunView Environment Variables and Windows

SunView programs may start and then give the error message shown below.

```
window: Base frame not passed parent window in environment
Segmentation fault (core dumped)
```

The Problem Defined

SunView code operates properly only when it has access to the SunView environment variables that refer to the current or parent window.

If you use `su`, `login`, `rlogin`, or are not running in the `suntools` environment on the console, you do not have any of those environment variables.

The fact that something dumps core is a bug, caused by the `window_create` routine returning a null pointer.

The Workaround

Avoid making the assumption that a routine will not fail and then not check for the return of a null pointer. Any program using the `window_create` routine should check for a null pointer being returned before using the routine's results.

Please note that `shelltool` does not check for the return of null pointers.

`textedit` Memory

`textedit` Window Maximum Memory

Customers using `textedit` windows may see the below error message when editing in memory. This occurs when inserting text into a `textsw`, `textedit`, or `cmdtool`.

```
Insertion failed.  The memory buffer is full.
```

The Problem Defined

The known limitation for editing in memory is described in bud reference id 1002225. A text window cannot have more than 19975 characters inserted into it. Once this limit has been reached, no more data may be added to the text window, even if the existing contents are erased from the window.

No messages appear either from the console or from the client application. The documentation indicates that the insert failure is due to a `malloc` failure (running out of memory). However, this error message is misleading.

The SunOS 3.0 documentation does not describe this situation. The SunOS 3.2 documentation describes `TEXTSW_MEMORY_MAXIMUM` that allows you to get or set the memory buffer size, which is initialized to a fixed size of 20000 characters and does not auto expand.

The Workaround

You can workaround the limitation for editing in memory using the defaults editor by increasing the field `TEXTSW_MEMORY_MAXIMUM`. Do this by changing the field `Memory_Maximum` in the 'Text' category of `defaultsedit`.

`suntools` Segmentation

Dynamic `suntools` Segmentation Violations

Some customers have observed the dynamic version of `suntools` dies immediately with a segmentation violation. The static version works, but complains with the error shown below.

```
read: error 0
```

The Problem Defined

This problem is caused by the user process not being able to create the file `/var/tmp/vm_fonts*`. This could be caused by permissions or by the way in which `/var/tmp` is mounted.

When you examine the core file produced by `suntools`, you see that the segmentation fault occurred in the `pf_close` procedure. This procedure works with the virtual memory fonts which are stored in `/var/tmp/vm_fonts*`.

The Solution

To fix this problem, examine the permissions of the `/var/tmp/vm_fonts*` directory, and the way in which it is mounted.

As a fail-safe test, create a directory `/var/tmp` on a local disk with enough space, and then bring up the dynamic version of `suntools`.

4

IN DEPTH

IN DEPTH **1175**

Porting to SunGKS 3.0 1175

4

IN DEPTH

Porting to SunGKS 3.0

Guide for Porting Programs from SunCGI to SunGKS 3.0

Abstract:

Gives summary information on porting programs from a SunCGI environment to that of SunGKS.[2] Gives information that is unique to SunGKS, discusses the differing coordinate systems, and provides a mapping of SunCGI C-language calls to SunGKS.

1: Introduction

This guide contains information on porting C application programs from a SunCGI environment to a SunGKS environment. Though it is primarily targeted for C-language interfaces for both SunCGI and SunGKS, the concepts remain the same regardless of the language interface. To apply the discussions within to the FORTRAN implementations of SunCGI and SunGKS, just change the calls and types in accordance with the rules for doing so already described in existing SunCGI and SunGKS manuals.

SunCGI and SunGKS are similar and yet at the same time very different. This guide will attempt to clarify both the similarities and the differences, and will provide a mapping from one world to the other. There are two sections to this guide.

A. A discussion of concepts in SunGKS that are not in SunCGI. For example, the concept of a SunGKS workstation, the differences in coordinate systems, the different abstraction models for inputs, the concept of a workstation type, and the concept of multiple active workstations (as opposed to view surfaces.)

B. A reference chapter giving mappings between SunCGI and SunGKS call interfaces.

---

[2] This In Depth feature is submitted by Marty Hess, Graphics Standards Group, Sun Microsystems, Inc.

2: New Concepts in SunGKS

This sections includes information on SunGKS and SunCGI similarities, segments, the concept of a workstation, transformation, viewing surfaces, workstation types, and higher-level input models.

2.1: Similarities

SunGKS and SunCGI have much the same reference model. They are both comprised of a series of function calls which implement output primitives, input primitives, attribute-setting functions, and control operations. Both contain primitives to draw polylines, polymarkers, fill Areas, cell arrays, text, circles, arcs, and so forth.

They both implement five equivalent types of input: locators, strokes, strings, valuators, and choices; in addition SunGKS implements the pick input. Both work in a normalized coordinate system, and transform coordinate values from a normalized system to an absolute screen coordinate system. In many of their respective function calls, they differ only in the types of C or FORTRAN structures passed to the implementation functions.

2.2: Segments

SunGKS provides the concept of a segment, which is not in SunCGI. Segments provide retained storage of graphics primitives. These segments can be transformed, highlighted, picked, and displayed. Priorities can be set which affect the order of drawing them on the display surface. Segments are outside the scope of this document, as they are not part of SunCGI. Refer to the SunGKS Reference Manual and use segments, for they can be an extremely powerful tool for creating highly sophisticated applications.

2.3: The Concept of a Workstation

SunGKS introduces the concept of a workstation. A workstation in SunGKS terminology can consist of a work surface with outputs and inputs or both. A workstation can also be a disk file containing metafile outputs or inputs, or it can be the Workstation Independent Segment Storage (WISS). SunCGI also uses a display surface for output and mouse or keyboard for inputs.

The difference is that with SunGKS one is allowed to open more that one workstation at a time; SunCGI allows multiple active view surfaces on a display screen, but does not allow multiple workstations of different types. SunGKS distinguishes between *open* and *active* workstations. An open workstation is one that has been opened and prepared for output/input/metafile/segment handling, whereas an active workstation is one which is actually receiving a primitive stream or transmitting a stream of primitives or input values. SunCGI has a call to initialize a view surface, and SunCGI has extensions to activate and deactivate view surfaces.

To convert then to a SunGKS world, it will be necessary to add calls to activate and deactivate workstations. Use of this can be very powerful. For example, with a single set of calls, one can send a stream of primitives to both the display surface and to metafile output, or to the screen and to a plotter or printer. One can also store primitives in the Workstation Independent Segment Storage (WISS). This mechanism is used to save picture images for later copying to a printer or plotter, or for sending to the metafile output workstation.

2.4: Transformations

SunCGI has two coordinate systems: Virtual Device Coordinates (VDC) and Device Coordinates (DC). The VDC extent can be from -32767 to +32767 in both the X and Y dimensions, and device coordinates can be any portion of the physical display surface. SunCGI is always *isotropic*, and uses only that portion of a window (always centered) that maintains that mapping. The system will always be scaled to fit in the specified window unless explicitly overridden by the application.

In the SunGKS world, there are three coordinate systems.

A. World Coordinates (WC). These are user units and are specified as single-precision floating point numbers. The values for a world coordinate window can be anything, but the direction vectors for the window must be to the right and up. The normalization transform (from World Coordinates to Normalized Device Coordinates) can be *anisotropic*, that is the magnitudes of the x-range and y-range can be different.

   SunGKS maintains a list of normalization transforms, and the application can use any one of these at a time. There is a priority ordering of these transforms available for use in transforming input values. If an input value is entered, SunGKS uses the highest priority normalization transform to convert values from normalized to world coordinates.

B. Normalized Device Coordinates (NDC). SunGKS transforms coordinate pairs from world to normalized coordinates(NDC) using the current *Normalization Transform.* NDC values are single-precision floating point numbers in the range from 0 (zero) to 1 (one), inclusive. NDC is the 'level playing field' of the SunGKS world. There can be many normalization transforms with widely different ranges of values. But in the end, all coordinate pairs transform down to NDC values in the range 0<=x<=1.0 and 0.0 <= y <= 1.0.

C. Device Coordinates (DC). SunGKS transforms coordinates in NDC to DC using the *Workstation Transformation. Device Coordinates* can be in pixels, for display workstations, or metres or 'other', for plotting or printing devices. The workstation transformation is device dependent. SunGKS provides a *Workstation Window* and a *Workstation Viewport*, which represent a *window* on the Normalized Device Coordinate space and a *viewport* on the device surface.

   There is only one transformation for a given device, although different workstations can map different portions of the NDC to their specific device. The workstation transform is isotropic; the aspect ratio of the NDC workstation window governs the transformation. If the aspect ratios are not the same, the device coordinate range is adjusted to the display or device surface in a device-dependent way.

The meaning for all of this to users of SunCGI will become obvious later. A simple approach for converting from SunCGI to SunGKS coordinate systems would be to take the current VDC coordinates, convert them to floating point, and specify the VDC extent as a normalization transform. The NDC viewport for the normalization transform would be 0.0 to 1.0 in both X and Y.

The device coordinates would be the same as for SunCGI. This scheme would give the easiest transition path to SunGKS. In this environment, input values would be transformed from DC to NDC to WC, giving the same values as the SunCGI application would give, except that the return values would be floating point numbers.

2.5: Viewing Surfaces

SunCGI and SunGKS have similar view surface models and the calls to set them up are similar. Both allow the user to specify creation of a *tool* by SunGKS (SunCGI), or to pass in an already-created *canvas*.

A key difference is that the default view surface for SunGKS is a tool, while the default workstation for SunCGI is an *overlay canvas*. Also, the default workstation is not as completely automatic in SunGKS as in SunCGI.

2.6: Workstation Types

SunGKS uses the concept of a workstation coupled with a workstation type. A workstation is an abstract notion, whereas a workstation type specifies an actual physical device. SunCGI deals only with the display surface, whereas SunGKS allows output to workstations defined as types `sun_tool`, `sun_canvas`, `hpgl`, `postscript`, `cgmo` (metafile output), `cgmi` (metafile input), or `wiss` (Workstation Independent Segment Storage).

In addition there are `mi`, `moasc`, and `mobin` which are the SunGKS metafile input or output. These workstations provide an 'audit trail' capability for SunGKS, and are different than the CGM metafiles which are strictly concerned with picture capture and restore.

2.7: Higher-Level Input Models

SunCGI allows a great deal of control over an input device, allowing for the specification of triggers, echo/prompt types, etc. SunGKS allows all of this as well, but does it with a higher-level set of input calls.

SunGKS and SunCGI share five classes of input: locator, string, choice, stroke, and valuator. In addition, SunGKS has a pick input class. SunGKS has two control calls: `Initialize {device type}`, and `Set {device class} mode`, where `mode` can be one of `REQUEST`, `SAMPLE`, or `EVENT`. These two calls (two for each of six classes) replace the `initialize_lid`, `release_input_device`, `associate`, `dissociate`, `set_initial_value`, and so on.

The SunGKS call to initialize device is different from SunCGI in that if the device is already active it will be reinitialized; the only proviso being that the device not be in `SAMPLE` or `EVENT` modes. The differences in input calls will be discussed fully in the next chapter.

3: Reference Manual for SunCGI-SunGKS Call Mapping

3.1: Initializing and Terminating SunCGI

This chapter gives a mapping from SunCGI to SunGKS calls. Each chapter of the SunCGI Reference Manual has a corresponding section in this chapter.

The calls in this section of the SunCGI manual map as follows:

A. `open_cgi()` maps to the SunGKS call `gopengks (perrfile, memory)`, where `perrfile` is the stream pointer to the error file, and `memory` is the constant `GMEMORY`. This call performs the same function as `open_cgi`.

B. `open_vws(name, devdd)` maps to the SunGKS call `gopenws(ws, conid, type)`, where `ws` is the user-specified handle for a workstation, `type` is the workstation type, and will be one of `sun_tool` or `sun_canvas`. SunGKS has other kinds of workstations for outputs, segment storage, and inputs, but these two map most directly to those used in SunCGI. The SunCGI view surface identifier `name` and the SunGKS workstation identifier `ws` perform the same function and have the same meaning.

They are both arguments to activate and deactivate view surfaces. Again, the only difference between SunCGI and SunGKS in this regard is that SunGKS can have more than one active workstation of different types, where both receive primitive streams. SunCGI, like SunGKS, allows more that one display view surface. `conid` is either `NULL` for type `sun_tool`, or is a valid handle to a SunView canvas for type `sun_canvas`. To convert to SunGKS, the `name` argument to `open_vws` should be changed to a unique integer workstation handle.

C. `activate_vws (name)` maps to the SunGKS call `gactivatews (ws)`. With the call to `gopenws` more that one view surface can be created, and with calls to `gactivatews` more than one view surface can be actively drawn to, similar in both SunGKS and SunCGI.

D. `deactivate_vws (name)` maps to `gdeactivatews (ws)`, where `name` and `ws` have the same meaning.

E. `close_vws (name)` maps to `gclosews (ws)` and has the same function as the SunCGI equivalent.

F. `close_cgi ()` maps to the SunGKS call `gclosegks ()` and performs the same function.

G. `vdc_extent (c1, c2)` does not have an exact equivalent in SunGKS. See the previous discussion concerning the different coordinate systems. The `vdc_extent` can be mapped to the SunGKS call `gsetwindow (transform, pwindow)`, where `transform` is the normalization transform number, which does not

have an equivalent in SunCGI, and `pwindow` is the World Coordinate (WC) window. `pwindow` is of type `Gwlimit`, which defines the minimum and maximum extent of the WC window.

H. `device_viewport (name, c1, c2)` maps to the SunGKS call `gsetwsviewport (ws, pviewport)`. In SunGKS there are two additional SunGKS calls to `gsetviewport` and `gsetwswindow`. These two calls must be made to set up the transformation from World to Normalized coordinates and from Normalized to Device coordinates.

I. clip_indicator (cflag) maps to the SunGKS call to `gsetclip (indicator)` where the arguments have the same meaning.

J. clip_rectangle (xmin, xmax, ymin, ymax) has no equivalent single function in SunGKS. If clipping is ON in SunCGI, one can convert calls to `clip_rectangle` to SunGKS by careful use of `gsetwindow` and `gsetviewport`.

K. `hard_reset` and `reset_to_defaults` have no equivalents in SunGKS.

L. clear_view_surface (name, defflag, index) map to the SunGKS call `gclearws (ws, clearflag)`, where name and ws have the usual equivalence, and `clearflag` is set to `CONDITIONALLY`, meaning the view surface should be cleared only if it has visible data on it, or to `ALWAYS`, which means always clear the surface.

M. `clear_control`, `set_error_warning_mask`, and `set_up_sigwinch` have no SunGKS equivalents. (It is possible to emulate `set_up_sigwinch` through the use of the `GKS_RESIZE_PROC` workstation configuration function.)

N. `inquire_device_identification (name, devid)` maps to the SunGKS function `Gwsct ginqwsconntype (ws, perrind)` which returns a structure containing the connection id (`NULL` or canvas handle) and the workstation type (`sun_tool` or `sun_canvas` for display workstations).

O. `inquire_device_class`, `inquire_output_capabilities`, `inquire_input_capabilities`, `inquire_trigger_capabilities`, `inquire_lid_capabilities` and `inquire_output_function_set` have no direct equivalents in SunGKS.

P. `inquire_physical_coordinate_system` maps loosely to the SunGKS call `ginqmaxdisplaysize (type, perrind)`

which returns the maximum size of the display surface in pixels or metres as well as in raster units.

Q. `inquire_vdc_type` has no SunGKS equivalent.

3.2: Output Primitives

In the area of output capabilities, SunCGI and SunGKS are similar in calls but slightly different in implementation. The biggest difference is in the coordinates used. SunCGI uses integers as coordinates, while SunGKS uses single-precision floating-point numbers.

The SunCGI data type `Ccoor` can be mapped to the SunGKS data type `Gwpoint` being mindful that `Ccoor` has integer members, whereas `Gwpoint` uses floats. The SunCGI data type `Ccoorlist` has no equivalent in SunGKS. The SunGKS output primitives use two arguments, `points` which is the point array, and `npoints` which is the number of these points.

A. `polyline (polycoors)` maps to the SunGKS call `gpolyline (npoints, points)` where the discussion above regarding the mapping of the `Ccoorlist` date type to separate `npoints` and `points` arguments must be taken into account in porting from SunCGI to SunGKS.

B. `disjoint_polyline (polycoors)` has no equivalent in SunGKS, and will have to be mapped into a call to `gpolyline`. The following model can be used. The `Coorlist` data type can be redefined for the SunGKS environment by the use of an `npoints` variable and a `points` array.

```
disjoint_polyline ( npoints, points )

        Gint    npoints;

        Gwpoint *points;

        {
        register int i;

        for ( i=0 ; i <n points ; i += 2 )

                gpolyline ( 2, &points[i] );
        }
```

C. `polymarker (polycoors)` maps to the SunGKS call `gpolymarker (npoints, points)`, where the mapping from `polycoors` to `Gwpoint` is as stated previously.

D. `polygon (polycoors)` maps to the SunGKS call `gfillarea (npoints, points)`. `partial_polygon (polycoors, cflag)` has no exact equivalent in SunGKS. However, SunGKS version 3.0 provide a Generalized Drawing Primitive (GPD) to create

complex polygons which is similar to the SunCGI call to `partial_polygon`. The SunGKS call is a GDP with function specifier of `gfillareasetgdp`.

There are differences between these two calls however. The SunGKS call specifies a collection of sets of points, each set describing a single closed area. The closed areas will draw depending on whether there are an even or odd numbers of overlaps between them. SunCGI permits the sets to be open, with `invisible` lines drawn between them.

E. `rectangle (rbc, ltc)` maps to a Generalized Drawing Primitive (GDP) in SunGKS. SunGKS uses the call `ggdp (npoints, points, function, datarec)` to give most of the SunCGI drawing functions an equivalent. For `rectangle` the two points are put into a point array `points` and the following call made:

```
ggdp (2, points, grectanglegdp, datarec);
```

This call is for non-filled rectangles. Changing the function parameter to `gfillrectanglegdp` results in a filled rectangle.

F. `circle (c1, rad)` maps to a SunGKS GDP with a function call of `gcirclegdp` or `gdiscgdp` for non-filled and filled circles, respectively. The calling sequences on the `circle` calls are different; SunCGI specifies the center point and the radius, whereas SunGKS specifies the center point and a point on the circumference. This is in general true of the other GDP calls of SunGKS. The general SunGKS model passes an array of points to the output primitive calls.

G. `circular_arc_center (c1, c2x, c2y, c3x, c3y, rad)` maps to the SunGKS GDP with a function call of `garcgdp` and behaves in the same way. Again, the arguments are different. Where SunCGI gives the center, two rays and a radius. SunGKS gives a center point, a start point for the arc, and a third point. The radius of the arc is specified by the distance from the center point to the start point. The third point is used for stopping the arc only. Each of the two ray points in SunCGI are used for angle purposes only, not for computing the radius of the arc.

H. `circular_arc_center_close ( ... )` maps to the SunGKS GDP with function call of `gclosedarc`. Both SunCGI and SunGKS allow the closure type to be specified as `PIE` or `CHORD`.

I. `circular_arc_3pt (...)` maps to a SunGKS GDP with function call of `garc2gdp`. The calls are similar in function and arguments. Both give a start point, an end point and an intermediate point. The SunCGI call is a subset of the SunGKS call. In SunCGI, the arc is guaranteed to pass through the intermediate point. This

means that the arc may be drawn in a clockwise `or` counter-clockwise direction.

In SunGKS, the arc is always drawn in a counter-clockwise direction. If the intermediate point lies on a counter-clockwise arc from the starting to ending points, then the arc will pass through the intermediate point; otherwise, the intermediate point is used only to calculate the arc.

J. `circular_arc_3pt_close (...)` is similar to the SunGKS GDP with function specifier `gclosedarc2gdp`. The closure rules can be specified in either SunCGI or SunGKS as `PIE` or `CHORD`. The discussion above regarding the meanings given to the arguments also applies.

K. `ellipse (...)` maps to the SunGKS GDP with function specifier `gellipticaldisc1`. The arguments are somewhat different, however. In SunCGI, the three arguments are the center point and the lengths of the major and minor axes. In SunGKS, the arguments specify the center point, the `endpoint` of one of the axes and the `endpoint` of the other. The first endpoint is precise. Since the two axes must be perpendicular, the second point is used to obain the length of the second axis; the angle being determined as 90 degrees in rotation from the first axis.

Note also that the SunCGI `ellipse` function is a closed object; there is no equivalent call in SunCGI for an open ellipse.

L. `elliptical_arc (...)` and the SunGKS GDP with function specifier `gellipticalarc1gdp` are equivalent. Both calls give an open elliptical arc. The arguments in SunCGI are the center of the arc, a starting ray, an ending ray, and the lengths of the major and minor axes.

In SunGKS, the arguments are the center, the end point of the first axis, the end point of the other axis (as above, the second point is used to compute distance, angle being determined from the first axis point), and starting and ending rays. In both cases, the ray points are used only for angles; the end points (or lengths of axes) are used to specify the elliptical arc parameters.

M. `elliptical_arc_close (...)` is mapped to the SunGKS GDP with function parameter `gclosedellipticalarc1gdp`. The same discussion applies to the argument list to this call as to the `elliptical_arc` call. In either SunCGI or SunGKS, the closure parameter is specified as `PIE` or `CHORD`.

N. `text (c2, tstring)` maps to the SunGKS call `gtext (pposition, text)` and produces similar results.

O. `vdm_text (cl, flag, tstring)` is equivalent to `text` except for handling of the append flag.

P. `append_text (flag, tstring)` has no exact equivalent in SunGKS. However, by use of the following sequence, `append_text` can be duplicated.

```
Gextent tex = ginqtextextent (ws, position, "sometext",&err);

        gtext (point, "some text");

        gtext (&tex.concat, "some more text");
```

The call to `ginqtextextent` returns the concatenation point for a string of text. This returned point can then used in a subsequent call to `gtext` to concatenate one string onto another. This is the same affect as the SunCGI call to `append_text`.

Q. `inquire_text_extent` maps into the SunGKS call `ginqtextextent (...)` and returns much the same values. SunGKS encapsulates these values into the `Gextent` structure.

R. `cell_array (p, q, r, dx, dy, colorind)` maps into the SunGKS function call `gcellarray (prectangle, pdimensions, rowsize, colourarray)`. One very important difference between the two calls is that the SunCGI call specifies a parallelogram for the output, where SunGKS specifies a rectangle. To get the same affects as SunCGI, the cell array must be put into a segment, and the segment transformation must be set to give the shear affect, by the use of unequal scaling in X and Y.

S. `pixel_array (pcell, m, n, colorind)` has no exact equivalent in SunGKS. However, with judicious setting of the normalization and workstation transformations, an affect similar to `pixel_array` can be achieved in SunGKS through `gcellarray`. (In this case the SunGKS emulation will be significantly slower than the SunCGI equivalent.)

T. `bitblt_source_array`, `bitblt_pattern_array`, and `bitblt_patterned_source_array` have no equivalents in SunGKS.

U. `inquire_cell_array` has no direct equivalent in SunGKS. However, the call to `inquire_pixel_array` discussed below has similar function to `inquire_cell_array`.

V. `inquire_pixel_array` maps to the SunGKS function `ginqpixelarray (ws, ppoint, pdimen, perrind)` The SunGKS call to `ginqpixelarraydim (ws, prect, perrind)` can be used to obtain the dimensions of the returned pixel array before calling `qinqpixelarray`.

W. The SunCGI calls `inquire_device_bitmap`, and `inquire_bitblt_alignments` have no equivalent in SunGKS.

X. `set_drawing_mode` in SunCGI is not implemented in SunGKS, because this call refers to bitblt operations. The call to `set_global_drawing_mode` is implemented as an `Escape` function in SunGKS with a parameter of `GESC_SRASTEROP`. In SunCGI, the `set_global_drawing_mode` call specifies a raster op of `REPLACE`, `AND`, `OR`, `NOT`, or `XOR`. The following table gives the equivalent values between SunCGI and SunGKS.

```
REPLACE = GROP_COPY

AND = GROP_AND

OR = GROP_OR

NOT = GROP_COPY_INVERTED

XOR = GROP_XOR
```

3.3: Attributes

SunGKS and SunCGI have similar sets of attributes for primitives. Both have the concept of `bundles` which allow the the goruping or `bundling` of attributes. The only exception to this is the SunGKS does not have the concept of a perimeter attribute.

3.3.1: Bundled Attribute Functions

The concept of aspect source flags are similar in both SunCGI and SunGKS. The method of setting them is slightly different. SunCGI assigns each aspect source flag a number; to define them a list is filled in with the desired attributes and values for them. In SunGKS, there is a structure with 13 enumerated data types in it. SunCGI has a total of 18 aspect source flags, while SunGKS has 13.

The differences are three flags for perimeter attributes in SunCGI, which SunGKS does not have, two flags for hatch index and pattern index, where SunGKS uses a single flag for both. In addition, SunCGI has separate flags for text font index and text precision, where SunGKS uses a single flag for bundling text font/precision.

A. `set_aspect_source_flags (flags)` is mapped (roughly) to the SunGKS call `gsetasf`. See the paragraph above for differences in the call.

B. `define_bundle_index (index, entry)`. There is no equivalent function for this in SunGKS.

3.3.2: Line Attributes

Most line attributes are similar between SunCGI and SunGKS.

A. `polyline_bundle_index (index)` maps to the SunGKS call `gsetlineindex (index)`. The two calls are equivalent.

B. `line_endstyle (type)` has no equivalent in SunGKS. SunGKS has no way of setting the end styling for lines.

C. `line_width_specification_mode` also has no equivalent in SunGKS. SunGKS has only `SCALED` line widths. To quote from the SunGKS Reference Manual, "... the linewidth scale factor is applied to the nominal linewidth on a workstation; the result is mapped by the workstation to the nearest available linewidth. The nominal linewidth is a part of a workstation description table."

D. `line_width (index)` maps to the SunGKS function call `gsetlinewidth (value)` and each performs similar functions, with the difference being that the SunGKS call always refers to a line-width scale factor, never to an absolute line-width that is possible with the SunCGI call.

E. `line_color (index)` maps directly to the SunGKS call `gsetlinecolour (index)`; both are indexes into the color lookup table.

F. `line_type (type)` and the SunGKS function `gsetlinetype (type)` are equivalent. The SunCGI types map to {`SOLID`, `DOTTED`, `DASHED`, `DASHED_DOTTED`, `DASH_DOT_DOTTED`, and `LONG_DASHED`}, whereas the SunGKS types are mapped to {`Solid`, `Dashed`, `Dotted`, and `Dashed-Dotted`}.

3.3.3: Polymarker Attributes

As with line attributes, the polymarker attributes are similar between SunCGI and SunGKS. The biggest difference is that, like line widths, polymarker sizes in SunGKS are scale factors from a nominal marker size on a given workstation, whereas in SunCGI, marker sizes can either be absolute sizes in pixels, or as a percentage of VDC space.

A. `polymarker_bundle_index (index)` maps exactly to the SunGKS function `gsetmarkerindex (index)` and perform the same function of setting the marker bundle index.

B. `marker_type (type)` and the SunGKS function `gsetmarkertype (type)` have the same processing. Both the SunCGI and SunGKS marker types are mapped to {`DOT`, `PLUS`, `ASTERISK`, `CIRCLE`, and `X`}.

C. `marker_size_specification_mode (mode)` has no equivalent call in SunGKS. SunGKS only allows the `SCALED` method of computing marker sizes, where the scaling is to a nominal marker size and is workstation dependent.

D. `marker_size (index)` maps to the SunGKS function `gsetmarkersize (size)` and both do the same thing; however, the SunGKS size is always a scaled size, whereas the SunCGI index can be `SCALED` or `ABSOLUTE`.

E. `marker_color (index)` and the SunGKS function `gsetmarkercolour (index)` are equivalent; both are indexes into the color lookup table.

3.3.4: Fill Attributes

As with the other attribute sets, the fill attributes are similar between SunCGI and SunGKS. The biggest difference is that SunGKS has a single attribute for both hatch index and pattern index. The index has two different meanings depending on whether the fill style is `HATCH` or `PATTERN`. The other big difference is that SunGKS does not have the fill perimeter attribute. This can be simulated in SunGKS by drawing the perimeter after drawing the fill itself.

A. `fill_area_bundle_index(index)` and the SunGKS function `gsetfillindex(index)` and equivalent in function.

B. `interior_style (istyle, perimvis)` and the SunGKS function `gsetfillintstyle (style)` are equivalent in the first argument. SunCGI maps the interior style to `{HOLLOW, SOLID, PATTERN, HATCH}`, while SunGKS maps the style to `{HOLLOW, SOLID, PATTERN, and HATCH}`. The second argument `perimvis` in the SunCGI function has no equivalent in SunGKS, because SunGKS does not have the concept of fill perimeter attributes.

C. `fill_color(color)` and the SunGKS function `gsetfillcolour(index)` are similar in function with one important difference. If the fill interior style is `HOLLOW`, the fill color in SunGKS is the color used to draw the perimeter of the fill area. In SunCGI, since it has perimeter attributes, uses the perimeter values to draw the outline of the fill area if the fill style is `HOLLOW`.

D. `hatch_index (index)` and `pattern_index (index)` in SunCGI both map to the SunGKS function `gsetfillstyleindex (index)`. SunGKS uses the call to `gsetfillstyleindex` for both fill styles `PATTERN` and `HATCH`. The corresponding bundle tables have only one attribute as well.

E. `pattern_table (index, m, n, colorind)` and the SunGKS call `gsetpatrep (ws, index, prep)` are similar in nature. The `Gptbundl` structure of SunGKS contains the factors `m` and `n` as well as the pattern array.

F. `pattern_reference_point (begin)` and the SunGKS call `gsetpatrefpoint (ppatref)` are equivalent: both are specified in VDC (WC) and give the reference point for the pattern.

G. `pattern_size (dx, dy)` is equivalent to the SunGKS function `gsetpatsize (ppatsize)`. Both specify VDC (WC) values that specify the actual size of the pattern. In both cases, the minimum pattern size in device coordinates is one pixel.

H. `pattern_with_fill_color (flag)` has no equivalent in SunGKS.

3.3.5: Perimeter Attributes

Perimeter attributes in SunCGI have no direct equivalents. However, for all of the output primitives it is fairly straightforward to draw the perimeter using polylines after issuing the fill primitive. There is an equivalent polyline function for each conic primitive in SunGKS which allows this.

3.3.6: Text Attributes

Text attributes in SunGKS are very similar to those in SunCGI, with only a few minor differences.

A. `text_bundle_index (index)` is equivalent to the SunGKS function `gsettextindex (index)`, although the contents of the text bundle in SunGKS is slightly different, as will be seen below.

B. `text_precision (type)` and `text_font_index (index)` as a pair are equivalent to the SunGKS function `gsettextfontprec (ptxfp)`. The SunGKS structure `ptxfp` encapsulates both the text font and precision into a single structure. Care must be taken when converting from SunCGI to SunGKS to make sure that both can be set at the same time.

C. `character_set_index (index)` has no equivalent in SunGKS; the font index in SunGKS can be used to specify national character sets.

D. `character_expansion_factor (efac)` and the SunGKS call `gsetcharexp (exp)` are equivalent, and both have the same default value (1.0).

E. `character_spacing (spcratio)` and the SunGKS function `gsetcharspace (spacing)` are different in function. The SunCGI sets a character height-width ratio, whereas the SunGKS function is a factor which adds to or subtracts from the default spacing of 0.0.

In SunGKS, normal character spacing is 0.0; numbers greater that 0.0 increase spacing. For example, a spacing factor of 0.25 indicates that spacing should be 1.25 times the normal spacing. Negative numbers reduce spacing between characters, and could cause overlap of characters.

F. `character_height (height)` and the SunGKS function `gsetcharheight (height)` are equivalent in function. In SunCGI, changing the height could change the character spacing; in SunGKS, characters will have a `normal` character spacing.

G. `fixed_font (flag)` has no equivalent in SunGKS. SunGKS text precision STRING, CHAR, or STROKE may defined either fixed- or variable-width fonts.

H. `text_color (index)` is equivalent to the SunGKS call `gsettextcolour (index)`.

I. `character_orientation (xbase, ybase, xup, yup)` and the SunGKS function `gsetcharup (pcharup)` are similar but not identical in function. In SunCGI, both the `base` and up `vectors` are defined; in SunGKS, only the `up` vector for the character is defined. The base vector is -90 degrees from the `up` vector.

The SunCGI orientation thus allows shearing of characters by giving `base` and `up` vectors that are not orthogonal. In SunGKS, one would have to put the characters into a segment and give the segment a transform with unequal scaling in X and Y, in order to get a shearing or oblique affect on the character.

J. `character_path (path)` and the SunGKS function `gsettextpath (path)` and identical in function. In SunCGI, the path maps to `{TP_RIGHT, TP_LEFT, TP_UP, and TP_DOWN}`, as does SunGKS.

K. `text_alignment (halign, valign, hcalind, vcalind)` and the SunGKS function `gsettextalign (ptxalign)` are similar, except that the SunCGI function allows continuous alignment in both the horizontal and vertical directions. The SunCGI function maps horizontal alignment to `{LFT, CNTER, RGHT, and NRMAL}` and vertical alignment to `{TOP, CAP, HALF, BASE, BOTTOM, or NORMAL}`.

The SunGKS equivalent mappings are `{TH_NORMAL, TH_LEFT, TH_CENTRE, and TH_RIGHT}`, and `{TV_NORMAL, TV_TOP, TV_CAP, TV_HALF, TV_BASE, and TV_BOTTOM}`. In both cases the `NORMAL` values for horizontal and vertical alignments map to different defaults depending on the text path. Both map to the same defaults.

3.3.7: Color Attributes

Both SunCGI and SunGKS let the user set color map table entries and to create custom color values. The calls to do this are slightly different.

A. `color_table (istart, clist)` and the SunGKS function `gsetcolourrep (ws, index, prep)` are similar in nature. The SunCGI call allows the user to set a list of color values, where the SunGKS call only sets one color table entry at a time. The SunCGI RGB values are integers between 0 and 255, whereas the SunGKS values are floats in the range 0.0 to 1.0.

3.3.8: Attribute Inquiry Functions

SunCGI and SunGKS both have inquiry functions to inquire the output primitive attribute settings. SunCGI returns each of line, polymarker, text and fill attributes in a single structure, where SunGKS has individual calls for each attribute.

A. `inquire_line_attributes` map to the SunGKS functions `ginqlinecolour (perrind)`, `ginqlineindex (perrind)`, `ginqlinetype (perrind)`, and `ginqlinewidth (perrind)`.

B. `inquire_marker_attributes` map to the SunGKS functions `ginqmarkersize (perrind)`, `ginqmarkertype (perrind)`, `ginqmarkercolour (perrind)`, and `ginqmarkerindex (perrind)`.

C. `inquire_fill_area_attributes` map to the SunGKS functions `ginqfillcolour (perrind)`, `ginqfillindex (perrind)`, `ginqfillintstyle(perrind)`, and `ginqfillstyleindex(perrind)`. The SunCGI call also returns perimeter visibility, perimeter style, width and color. The SunCGI call distinguishes between `hatch_index` and `pattern_index`, where the SunGKS call only returns the fill style index.

D. `inquire_pattern_attributes` is similar to the SunGKS calls `ginqpatrefpoint (perrind)`, `ginqpatrep (perrind)`, `qingpatheight (perrind)`, and `ginqpatwidth (perrind)`; The SunGKS functions return both the pattern width and height vector, where the SunCGI functions return the pattern size as integers. SunGKS pattern sizes may be subject to a normalization transformation which could result in unequal scaling of the pattern size in x and y.

E. `inquire_text_attributes` is similar to a collection of SunGKS calls: `ginqcharbase (perrind)` which returns the character base vector (although this vector is not directly settable), `ginqcharexp (perrind)`, `ginqcharheight (perrind)`, `ginqcharspace (perrind)`, `ginqcharup (perrind)`, `ginqcharwidth (perrind)`, `ginqtextfontprec`

(perrind), ginqtextindex (perrind), and ginqtextpath (perrind).

F. inquire_aspect_source_flags is equivalent to the SunGKS call ginqasf (perrind) with the exception that SunCGI has more flags than SunGKS.

3.4: Inputs

SunCGI and SunGKS both implement similar kinds of input. They have five type of input in common: locators, strings, choices, strokes, and valuators. SunCGI tends to require several different calls to set up inputs, whereas SunGKS uses fewer higher level calls. Both SunCGI and SunGKS specify in input device by using the device class (LOCATOR, CHOICE, STRING, STROKE, VALUATOR) and the device number.

SunCGI initializes a device by using calls to initialize_lid, associate, set_initial_value, track_on, and track_off. SunGKS, on the other hand, has an initialization function for each class of input, i.e. ginitloc, ginitstroke, ginitstring, ginitchoice, and ginitval.

SunGKS specifies initial values with the initialization calls, uses the prompt/echo code to specify how triggers will be associated, and also uses the prompt/echo codes to determine whether echoing will be on or off. SunCGI puts inputs into five states: RELEASED, NO EVENTS, RESPOND EVENT, REQUEST EVENT, and QUEUE EVENT. Mode changing is done implicitly with input calls. On the other hand, SunGKS has three input states: REQUEST, SAMPLE, and EVENT. By default a device is in REQUEST mode. An explicit call to, for example, gset{loc}mode is made to change the input state.

In REQUEST mode, only calls of the form greq{loc} can be make to get input. In SAMPLE mode, calls of the form gsample{loc} are used, and in event mode gawaitevent is used. There are no explicit calls to release a device in SunGKS. This is done by putting the device into REQUEST mode.

SunCGI uses an initialized input device in state NO EVENTS to do both REQUEST and SAMPLE input. The call to request_input is used for both modes of input, depending on the associated triggers. If the 'mouse movement trigger is specified then the cursor value is sampled at the time a request_input call is made. If the trigger is a mouse button, then pressing a mouse button is required to return the measure of an input device.

SunGKS transforms an input data point using both the workstation and normalization transforms. Input data is first mapped to NDC using the current workstation transform. Then SunGKS considers the list of normalization transforms in order of priority. It transforms the points from NDC to WC using the highest priority normalization transform that contains the point. Input transform priorities are set using gsetviewportpri.

3.4.1: Locator Inputs

`initialize_lid (LOCATOR, dev, ival)` is roughly equivalent to the SunGKS call `ginitloc (ws, dev, pinit, pet, parea, precord)`. To specify all of the locator initial values, additional calls in SunCGI are required. The initial value for locators is in the field `ival`, which becomes `pinit` in SunGKS. The echo area parameter has no meaning for locator inputs.

The prompt/echo type `pet` in SunCGI specifies only the default cursor or a printers-fist. In SunGKS, there are a wide range of prompt/echo types available. In SunGKS, setting the prompt/echo type will cause echoing by default, as affected by the call to `gsetlocmode` which could turn echoing off. In SunCGI, calls to `tracking_on` and `tracking_off` are made to turn the echo on or off. The SunCGI call `release_input_device` is necessary to release input on a locator device.

In SunGKS, setting the input mode to `REQUEST` allows for the device to be reinitialized with the `ginitloc` call. In `REQUEST` mode of SunGKS, calls can be made to `greqloc` which is similar to SunCGI calls to `request_input`. There is one major difference, however. In SunCGI, calls to `associate` and `dissociate` can be made to specify the trigger values for locator inputs. One of the triggers is 'mouse moved.

This would be the equivalent of `gsampleloc` in SunGKS; that is, return the current position of the mouse without waiting for a mouse button to be depressed. `REQUEST` input in SunGKS would wait for a mouse button to be pushed. `gsetlocmode` is used in SunGKS to specify the input mode (`REQUEST`, `SAMPLE`, or `EVENT`), and whether echoing is to occur. To initialize a device in SunGKS, the input mode must be `REQUEST`.

3.4.2: Choice Inputs

The discussion for choice inputs is similar to that of locator inputs, except that `ginitchoice` is called to initialize the device, as opposed to the SunCGI call to `initialize_lid (CHOICE...)`. The SunGKS call `gsetchoicemode` is used to specify `REQUEST`, `SAMPLE`, or `EVENT` modes, or to turn the echoing on or off. In SunCGI the choices are selected with mouse buttons by setting the associated trigger, and the choice value returned in the button struck. In SunGKS, choice input is done either with a SunView cycle item, or by calling up a pop up menu with one of the mouse buttons. Calls to `gsetchoicemode` can be made to turn this echoing on or off.

3.4.3: Stroke Inputs

The discussion of the calls for stroke inputs is similar to the other types. `initialize_lid (STROKE...)` is replaced by `ginitstroke`, `gsetstrokemode` being used to set the stroke mode (`REQUEST`, `SAMPLE`, or `EVENT`) and the echoing state. In SunCGI, the track of a stroke can be a solid line from point-to-point, a vertical or horizontal line to the current point, or a rubber-band box to the current point. In SunGKS, echoing can be a plus sign (+) at each point of the stroke, straight lines connecting successive points, or a marker at each point.

3.4.4: Valuator Inputs

Valuator inputs in SunCGI are realized using a digital representation of the value or a printers-fist to show the value. In SunGKS, a Sun View slider item is used. The equivalent calls are `initialize_lid (VALUATOR...)` in SunCGI and `ginitval` in SunGKS. `gsetvalmode` is used to turn echoing on or off, and to set `REQUEST`, `SAMPLE`, or `EVENT` modes.

The SunGKS call to `ginitval` specifies the initial value as the `init` value and the ranges for valuator input in the `precord` field. In SunCGI, the ranges and initial values can be changed with calls to `set_initial_value` and `set_valuator_range`. This can be done at any time. In SunGKS, repeated calls to `ginitval` can be made to achieve the same result.

3.4.5: String Inputs

In SunCGI string inputs are initialized using `initialize_lid (STRING` `gsetstringmode` is used to turn echoing on or off, or to set the string into `REQUEST`, `SAMPLE`, or `EVENT` mode. SunCGI echos strings by using the concatenation point of text. Setting of prompts and this text concatenation point is done outside of the calls to get strings. In SunGKS, the echo area specified in `ginitstring` is used for echoing of strings, and the call to `ginitstring` can specify an initial string to be displayed in the echo area.

3.4.6: Event Inputs

In both SunCGI and SunGKS, event inputs are allowed on all five common input types. In SunCGI, a call to `enable_events (devclass, devnum)` is made to enable events for a specific device. In SunGKS, a call is made to `gsetlocmode`, `gsetstringmode`, `gsetstrokemode`, `gsetvalmode`, or `gsetchoicemode` to set the mode to `EVENT`. In SunCGI, `disable_events (devclass, devnum)` is used to disable the event processing, while in SunGKS, the same mode setting functions are used, with an input mode other that `EVENT` being the mechanism to disable events.

In both SunCGI and SunGKS, a call to `await_event` or `gawaitevent` is made to wait for an event. In both cases the device class and device number are returned in the call. There is also a return which indicates that a timeout value was reached without an event taking place.

In SunCGI, the event is returned in the call to `await_event` while in SunGKS, a second call must be made to retrieve data from the last event. For this purpose, a call to `ggetloc`, `ggetstring`, `ggetstroke`, `ggetval`, or `ggetchoice` is made.

5

HINTS AND TIPS

HINTS AND TIPS **1197**

`vi` Hints 1197
FORTRAN Tips 1199

5

HINTS AND TIPS

`vi` Hints

Sun386*i* and `vi` Hints and Tips

Customers using Sun386*i* machines running Sun386*i* SunOS 4.0.1 with 15" monochrome monitors may find the two `vi` hints in this article useful in avoiding corrupted displays.

The Problem

Customers using `vi` to edit a file outside of SunView may see a corrupted display. Moving the cursor along a line may cause about four lines to appear over the current line.

The `nvram` setting for location 16 was reset from 15 to 0. Checking the `stty` and using `tset -s` showed everything to seem normal, including the number of columns and rows.

Hint Number 1

If you need to use `vi` outside of SunView on the 15" monochrome or the 14" color monitors, as root, use the command shown below to set the number of rows to match the size of the screen.

```
# stty rows 30
#
```

Hint Number 2

You can make a permanent change to work around the problem on the 14" and 15" monitors by editing, as root, your `termcap` file.

```
# mount -o remount /usr
# cd /usr/share/lib
# vi termcap
```

You will change the `Mu|sun|Sun Microsystems Workstation console:` entry by changing the number of lines to '29'. This is the `li#` field in the second line of the console entry. See the field to be changed, emboldened in the below example.

```
Mu|sun|Sun Microsystems Workstation console:\
        :am:bs:km:mi:ms:pt:li#29:co#80:cl=^L:cm=\E[%i%d;%dH:\
        :ce=\E[K:cd=\E[J:md=\E[1m:us=\E[4m:ue=\E[m:so=\E[7m:se=\E[m:rs=\E[s:\
        :al=\E[L:dl=\E[M:im=:ei=:ic=\E[@:dc=\E[P:\
        :AL=\E[%dL:DL=\E[%dM:IC=\E[%d@:DC=\E[%dP:\
        :up=\E[A:nd=\E[C:ku=\E[215z:kd=\E[221z:kr=\E[219z:kl=\E[217z:\
        :k1=\E[224z:k2=\E[225z:k3=\E[226z:k4=\E[227z:k5=\E[228z:\
        :k6=\E[229z:k7=\E[230z:k8=\E[231z:k9=\E[232z:
```

FORTRAN Tips

FORTRAN: Optimization Tips

This article contains tips on combinations of options and examples to get improved performance for most FORTRAN programs.

Please note that independent of hardware, and in the absence of pointers the following is true.

```
-O4 = -O3 = -O
```

SPARC and FORTRAN 1.2: Example 1

The below example applies to Sun-4/1xx and Sun-4/2xx systems running FORTRAN 1.2.

```
f77 -O4 /usr/lib/f77/libm.il -dalign any.f -Bstatic -lm
```

`-dalign` This option usually improves performance, sometimes significantly, when all double precision variables are aligned on double precision boundaries.

`-Bstatic` This option usually improves performance in systems where only one application is run at a time.

`-lm` This must be the last option on the line.

SPARC and FORTRAN 1.2: Example 2

The below example applies to Sun-4/80, Sun-4/3xx, and Sun 4-2xx systems with FPU2 running FORTRAN 1.2.

```
f77 -O4 -dalign -Qoption as -Ff0  -Qoption iropt -l4  any.f  \
    /usr/lib/f77/libm.il  /usr/lib/libm.il  -Bstatic  -lm
```

`Qoption as -Ff0`
This option causes code to be generated that is suitable for the Sun-4 models listed above.

Sun-3 and FORTRAN 1.2

The below example applies to Sun-3 systems running FORTRAN 1.2 *without* a Floating Point Accelerator (FPA).

```
f77 -O4 -f68881 /usr/lib/f77/f68881/libm.il  /usr/lib/f68881/libm.il \
    any.f -Bstatic -lm
```

The next example applies to Sun-3 systems running FORTRAN 1.2 *with* a Floating Point Accelerator (FPA).

```
f77 -O4 -ffpa  /usr/lib/f77/ffpa/libm.il /usr/lib/ffpa/libm.il \
    any.f -Bstatic -lm
```

`-fsoft` This option is recommended only for a Sun-3/50 that does not have a 68881.

`-fswitch` This option is not recommended.

Sun386*i* and FORTRAN 1.1

The below example applies to Sun386*i* systems running FORTRAN 1.1.

```
f77 -O4 /usr/lib/libm.il any.f -Bstatic -lm
```

6

THE HACKERS' CORNER

THE HACKERS' CORNER **1203**

FORTRAN Referencing 1203

6

THE HACKERS' CORNER

FORTRAN Referencing

FORTRAN Cross Referencing: An Introduction to `lex` and `awk`

This month's **Hackers' Corner** contains a shell archive about FORTRAN cross referencing and a brief introduction to `lex` and `awk`.[3]

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.

Why Use `lex` and `awk`?

A common question[4] arises from FORTRAN programmers starting the UNIX learning curve: 'How may we get the listings, memory maps, and cross reference table that our mainframe FORTRAN compiler gives us when we use this UNIX FORTRAN compiler?'

We know that compilers are for generating object code and to get things like listings we use *indent(1)*, memory maps are produced by *nm(1)*, and so forth; and in fact we know that we use *cxref(1)* for C language cross reference listings when we debug a program we have written. A careful, detailed examination of the manual (that is, we try `man -k`) reveals no mention of a comparable program (which would obviously be called `fxref`, right?) in the SunOS distribution. The answer is obvious: write our own `fxref` program.

Thinking back, we find that there are two UNIX utilities that are commonly used to help write new languages or new implementations of old languages: *lex(1)* and *yacc(1)*. `yacc` is useful in writing a compiler, but the actions we will be taking after parsing the input file are so simple, we will not need it.

---

[3] The shell archive in this month's **Hackers' Corner** is submitted by Michael Fischbein, Technical Consultant, Sun Professional Services, Albany District Field Office, Albany, New York, USA.

[4] Well, two people asked about it once.

An examination of the FORTRAN standard shows some potential trouble spots. For example, most computer languages have their key words reserved for that key action. That is, a Pascal programmer could never use a variable called `begin`, or a C programmer use `case`. In FORTRAN, however, it is perfectly legal to have a variable called `WRITE`, or even `DO10I`. Given that spaces are ignored in FORTRAN, that last variable could also be written as `DO 10 I`. We find that it could be very difficult to do a full, 100% solution on this `fxref` utility.

The parser for a FORTRAN compiler (which is basically what we are writing) is complex compared to that of most other languages because of this lack of reserved words. Doing a complete, released, and supported solution for this problem is more difficult than the quick solution we will do here. On the other hand, as long as the users do not use `if`, `endif`, `common`, and so forth as variable names, they will not see the difference. Fortunately, most FORTRAN programmers find code that uses those (quite legal) identifiers as variable names difficult to read themselves and so use them infrequently.

It is prudent, however, to caution the users that this is not a 100% solution. The final polishing touches would take more effort than this entire exercise which was written to learn about `lex` as well as generating a useful utility. That said, let us continue creating our quick but useful utility. The script that ties it all together, which the users will execute, is called `fxref` . We have two helper functions, called `xrefa` and `xrefb` , written in `lex` and `awk`, respectively, and a Makefile to compile `xrefa` automatically.

What we need is a program that can look at a line of FORTRAN, discard the structure words (such as `program`, `if`, and so forth), and give us the identifiers referenced and the line number. Since we want to be able to feed programs that contain many subroutines and functions into the program, we ought to get the module name also. Eventually, we will want to combine all the references to a given identifier and sort the output, but we will save that for later.

One source of confusion is that some FORTRAN programmers use tabs in their programs. First, let us remove all of the tabs. How? Well, we could make it part of a complex *lex(1)* grammar, or we could make all the conditionals use *isspace(3)*, and try to keep track of the column number (which is significant in FORTRAN), but we will simply run through the *expand(1)* program that has been provided with standard SunOS. Now we can be sure our source file has no embedded tabs. Similarly, we deal with FORTRAN's case insensitivity with the *tr(1)* utility.

At this point, we really want to start to write the `lex` program, or grammar, that will generate our parser, but first we have to find out what the input to `lex` looks like. A `lex` input file is divided into three sections, called the *definitions*, *rules*, and *subroutines* sections, separated by lines containing double percents (%%). The only required section is the rules section that contains the actual `lex` grammar. The definitions section allows macros that simplify the writing of rules, lets the framework for the main executable be entered if other than the

default behavior is desired, and allows some control over the size of certain internal tables `lex` uses. The subroutines section contains the subroutines that are referenced in the grammar or the definitions section.

So let us start writing our grammar and see what we get. The first thing we want to do is ignore comments, which are distinguished by a letter 'c' or 'C' in the first column or an asterisk (*) for box comments. A regular expression that will match any of those lines is `^[Cc*\].*$` . The caret (^) indicates the beginning of the line, then the brackets ([ ]) enclose a list of characters, any one of which will match with the first character on the line. Note that the asterisk is escaped from special expansion by the backslash. We match the rest of the line with the `.*$` , which means any character (.), zero or more times (* with no backslash), through the end of the line ($). Now we want to discard that line, so we do not specify any action for it, just follow it with some white space and a semicolon (;), so the first line of our grammar (which comes after the definitions section, so look after the first %% line) looks like:

```
^[Cc*\].*$      ;
```

In a similar fashion, we write more complex regular expressions to discard the key words that FORTRAN uses for operators (`".and"  ;`), declarations (`double[ ]*precision  ;`), and so on. We also want to recognize subroutine calls (`call[ ]+[_a-zA-Z0-9]+  subcal();`), format statements (`^[ ]*[1-9][ 0-9]*format[ ]*"(".*  fndfmt();`), and the like, and go to their respective special subroutines (defined in the subroutines section). Finally, what is left over are the identifier references we really want to keep track of, so we do the actions to take care them. Note the C code fragment prints a minus sign in front of the line number if the identifier is followed by an equals on the line; this lets us mark those uses of the identifier as a place where its value changes and allows sorting those references out by numeric comparisons.

This routine will give us a file with each identifier reference on a separate line, in the order they were encountered in the FORTRAN source. We can sort the output using the standard UNIX utility, *sort(1)*. We want to sort alphabetically by identifier name, then module name, then numerically by line number. Finally, let us remove multiple references in a single line. A few quick options to `sort`, and we do not have to concern ourselves with programming that!

Now we have the identifiers grouped appropriately and sorted properly, but still only one reference per line. We want to collect all the consecutive references to a given identifier in a given module on a single line. This does not sound common enough to match the capabilities of a provided utility, and another detailed perusal of the manual (that is, we try `man -k` again) shows no obvious utilities that will be of help.

We think about using our new expertise in writing regular expressions and using `sed`, but that does not seem helpful. The pattern matching and particularly the memory requirements to accumulate values seem to be a little beyond `sed`'s

capabilities, so let us try *awk(1)*. To make the output more readable we will put a blank line between letters of the alphabet, and change the minus sign (for assignments) to a more user friendly equals sign.

We start the `awk` script by explicitly setting the Output File Separator (OFS) to a tab character before reading in anything. On the first line in we want to set up our current identifier and module, so that a blank line does not get printed before the list. Then we check to see if the current module is different from the module name on this line; if so we print the accumulated information from the old identifier and module, check to see if we need to print a blank line (print with no arguments echos the input line), set up the new identifier and module, and start accumulating line numbers (checking for a '-' in the line number). We could replace the string oriented minus sign check with '($3 < 0)' but this saves the numeric conversion and seems more in keeping with our use of the '-' as a flag character, rather than a negative indication. Similar code occurs for a new identifier, and if we encounter the same identifier we simply accumulate the results.

Now we test the program on a few FORTRAN files our users have and then let them enjoy the same sort of debugging aids they are used to using on the mainframe. Of course, for our own debugging we use `dbx` or `dbxtool`, since that is a lot easier than checking memory maps and cross-reference tables, but the users will start using those very useful tools soon enough.

We now have a quickly written, useful utility that does a fairly complex task reasonably well. We have used a variety of the facilities of the SunPro environment to accomplish this, and find that our total source code fits fairly easily in only five pages. Further, by using appropriate utilities we have cut our development time to a fraction of what it would take to write the identical cross-referencer entirely in Pascal, say, or C. Such a program would probably execute slightly faster than our script, but it is likely that the execution time of either would be dominated by I/O time. The compactness of the source code obtained by using an appropriate language (such as *awk(1)* for line oriented pattern matches) minimizes bugs and bug tracing. If you do have a problem, it is easier to find in a 35 line program then a 350 line program. It also lets you produce working, tested code quickly.

An Example Usage

To use the shell archive shown in the following section, save the code into a file named `fortran.hack.corn` and then use the commands shown below to unpack the code, compile the programs, and print a copy of the above section that explains the programming exercise.

```
Be Happy :-) sh fortran.hack.corn
x - article.ms
x - fxref
x - xrefa.l
x - Makefile
x - xrefb
Be Happy :-) make all

lex  xrefa.l

413/450 nodes(%e), 5925/6000 positions(%p), 672/700 (%n),
73293 transitions, 244/250 packed char classes(%k),
6045/6100 packed transitions(%a), 7415/7500 output slots(%o)

cc -O -c lex.yy.c

rm lex.yy.c
mv lex.yy.o xrefa.o

cc -o xrefa xrefa.o -ll

Be Happy :-) troff -ms -Tlw article.ms
Be Happy :-)
```

The Shell Archive

The shell archive code appears on the following pages.

```
# This is a shell archive.  Remove anything before this line,
# then unpack it by saving it in a file and typing "sh file".
#
# Wrapped by rotary!msf on Fri Jun  2 16:50:23 EDT 1989
# Contents:  article.ms fxref xrefa.1 Makefile xrefb

echo x - article.ms
sed 's/^@//' > "article.ms" <<'@//E*O*F article.ms//'
@.ND
@.nr LL 6.5i
@.nr PS 12
@.nr VS 14
@.ll 6.5i
@.PP

@.sp -3
@.TL
@.B
FORTRAN Cross-reference and a Brief Introduction to
@.I lex
and
@.I awk
@.R
@.sp .5
@.PP
A common question*
@.FS
* Well, two people asked about it once.
@.FE
arises from FORTRAN programmers starting the
@.UX
learning curve: ``How may we get the
listings, memory maps, and cross reference table that our mainframe
FORTRAN compiler gives us when we use this
@.UX
FORTRAN compiler?''
@.PP
We know that compilers are for generating object
code and to get things like listings we use
@.I indent(1) ,
memory maps are produced by
@.I nm(1) ,
and so forth; and in fact we know that we use
@.I cxref(1)
for C language cross reference listings when we
debug a program we have written.  A careful, detailed examination of the
manual (that is, we try
@.I man
@.I -k )
reveals no mention of a comparable program (which would obviously
be called
@.I fxref ,
```

right?) in the SunOS distribution. The answer is obvious: write our
own
@.I fxref
program.
@.PP
Thinking back, we find that there are two
@.UX
utilities that are commonly used to help write new languages or new
implementations of old languages:
@.I lex(1)
and
@.I yacc(1) .
@.I yacc
is useful in writing a compiler, but the actions we will be taking after
parsing the input file are so simple, we will not need it.
@.PP
An examination of the FORTRAN standard shows some potential
trouble spots. For example, most computer languages have their key words
reserved for that key action. That is, a Pascal programmer could never
use a variable called ''begin,'' or a C programmer use ''case.'' In FORTRAN,
however, it is perfectly legal to have a variable called ''WRITE,'' or
even ''DO10I.'' Given that spaces are ignored in FORTRAN, that
last variable could also be written as ''DO 10 I.'' We find that it could
be very difficult to
do a full, 100% solution on this
@.I fxref
utility.
@.PP
The parser for a FORTRAN compiler (which is basically what we
are writing) is complex compared to that of most other languages
because of this lack of reserved words. Doing a complete,
released, and supported solution for
this problem is more difficult than the quick
solution we will do here. On the other hand, as long as the users do not use
''if,'' ''endif,'' ''common,'' and so forth as variable names, they will
not see the difference. Fortunately, most FORTRAN programmers find
code that uses those (quite legal) identifiers as variable names difficult
to read themselves and so use them infrequently.
@.PP
It is prudent, however, to caution the users that this is not a 100% solution.
The final polishing
touches would take more effort than this entire exercise
which was written to learn about
@.I lex
as well as generating a useful utility.
That said,
let us continue creating our quick but
useful utility. The script that ties it all together, which the users will
execute, is called
@.I fxref .
We have two helper functions, called
@.I xrefa
and

```
@.I xrefb ,
written in
@.I lex
and
@.I awk ,
respectively, and a Makefile to compile
@.I xrefa
automatically.
@.PP
What we need is a program that can look at a line of FORTRAN, discard
the structure words (such as ``program,'' ``if,'' and so forth), and
give us the identifiers referenced and the line number.
Since we want to be able to feed programs that contain many subroutines
and functions into the program, we ought to get the module name also.
Eventually, we will want to combine all the references to a given
identifier and sort the output, but we will save that for later.
@.PP
One
source of confusion is that some FORTRAN programmers use tabs in their
programs.  First, let us remove all of the tabs.  How?  Well, we
could make it part of a complex
@.I lex(1)
grammar, or we could make all the conditionals use
@.I isspace(3) ,
and try to keep track of the column number (which is significant in
FORTRAN), but we will simply run through the
@.I expand(1)
program that has been provided with standard SunOS.  Now we can be sure
our source file has no embedded tabs.  Similarly, we deal with
FORTRAN's case insensitivity with the
@.I tr(1)
utility.
@.PP
At this point, we really want to start to write the
@.I lex
program, or grammar, that will generate our parser, but
first we have to find out what the input to
@.I lex
looks like.  A
@.I lex
input file is divided into three sections, called the
@.I definitions ,
@.I rules ,
and
@.I subroutines
sections, separated by lines containing double percents: %%.  The only
required section is the rules section that contains the actual
@.I lex
grammar.  The definitions section allows macros that simplify the
writing of rules, lets the framework for the main executable be entered
if other than the default behavior is desired, and allows some control
over the size of certain internal tables
@.I lex
```

uses. The subroutines section contains the subroutines that are referenced in the grammar or the definitions section.
@.KS
@.PP
So let us start writing our grammar and see what we get. The first thing we want to do is ignore comments, which are distinguished by a letter ''c'' or ''C'' in the first column or an asterisk (*) for box comments. A regular expression that will match any of those lines is ^[Cc*\].*$. The caret (^) indicates the beginning of the line, then the brackets ([ ]) enclose a list of characters, any one of which will match with the first character on the line. Note that the asterisk is escaped from special expansion by the backslash. We match the rest of the line with the .*$, which means any character (.), zero or more times (* with no backslash), through the end of the line ($). Now we want to discard that line, so we do not specify any action for it, just follow it with some white space and a semicolon (;), so the first line of our grammar (which comes after the definitions section, so look after the first %% line) looks like:
@.LD
```
^[Cc*\].*$      ;
```
@.DE
@.KE
@.PP
In a similar fashion, we write more complex regular expressions to discard the key words that FORTRAN uses for operators (".and";), declarations (double[ ]*precision ;), and so on.
We also want to recognize subroutine calls
(call[ ]+[_a-zA-Z0-9]+ subcal();), format statements
(^[ ]*[1-9][0-9]*format[ ]*"(".* fndfmt();),
and the like, and go to their respective special subroutines (defined in the subroutines section). Finally, what is left over are the identifier references we really want to keep track of, so we do the actions to take care them. Note the C code fragment prints a minus sign in front of the line number if the identifier is followed by an equals on the line; this lets us mark those uses of the identifier as a place where its value changes and allows sorting those references out by numeric comparisons.
@.PP
This routine will give us a file with each identifier reference on a separate line, in the order they were encountered in the FORTRAN source. We can sort the output using the standard
@.UX
utility,
@.I sort(1) .
We want to sort alphabetically by identifier name, then module name, then numerically by line number. Finally, let us remove multiple references in a single line.
A few quick options to
@.I sort ,
and we do not have to concern ourselves with programming that!
@.PP
Now we have the identifiers grouped appropriately and sorted properly,

but still only one reference per line.
We want to collect all the consecutive references to a given identifier
in a given module on a single line. This does not sound common enough to
match the capabilities of a provided utility, and
another detailed perusal of the manual (that is, we try
@.I man
@.I -k
again)
shows no obvious utilities that will be of help.
We think about using our new expertise in writing regular expressions and
using
@.I sed ,
but that does not seem helpful. The pattern matching and particularly
the memory
requirements to accumulate values seem to be a little beyond
@.I sed 's
capabilities, so let us try
@.I awk(1) .
To make the output more readable we will put a blank line between
letters of the alphabet, and change the minus sign (for assignments)
to a more user friendly equals sign.
@.PP
We start the
@.I awk
script by explicitly setting the Output File Separator (OFS)
to a tab character before reading in anything. On the first line in we want
to set up our current identifier and module, so that a blank line
does not get printed before the list. Then we check to see if the
current module is different from the module name on this line; if so
we print the accumulated information from the old identifier and module,
check to see if we need to print a blank line (print with no arguments
echos the input line), set up the new identifier and module, and start
accumulating line numbers (checking for a ``-'' in the line number).
We could replace the string oriented minus sign check with ``($3 < 0)''
but this saves the numeric conversion and seems more in keeping with our
use of the ``-'' as a flag character, rather than a negative indication.
Similar code occurs for a new identifier, and if we encounter the
same identifier we simply accumulate the results.
@.PP
Now we test the program on a few FORTRAN files our users have and then
let them enjoy the same sort of debugging aids they are used to using
on the mainframe. Of course, for our own debugging we use
@.I dbx
or
@.I dbxtool ,
since that is a lot easier than checking memory maps and cross-reference
tables, but the users will start using those very useful tools soon enough.
@.PP
We now have a quickly written, useful utility that does a fairly complex
task reasonably well. We have used a variety of the facilities of the
SunPro environment to accomplish this, and find that our total source
code fits fairly easily in only five pages. Further, by using appropriate
utilities we have cut our development time to a fraction of what it would

```
take to write the identical cross-referencer entirely in Pascal, say, or C.
Such a program would probably execute slightly faster than our script, but
it is likely that the execution time of either would be dominated
by I/O time.  The compactness of the source code obtained by
using an appropriate language (such as
@.I awk(1)
for line oriented pattern matches) minimizes bugs and bug tracing.  If you
do have a problem, it is easier to find in a 35 line program then a 350 line
program.  It also lets you produce working, tested code quickly.
@//E*O*F article.ms//
chmod u=rw,g=r,o=r article.ms

echo x - fxref
sed 's/^@//' > "fxref" <<'@//E*O*F fxref//'
#!/bin/sh
#
# dir=/local/src/fxref
dir=`pwd`
case "$1" in
    -m)
        shift
        expand $* | tr A-Z a-z | ${dir}/xrefa -m | \
                    sort '-ut   ' +0 -1 +1 -2 +2n -3n | awk -f ${dir}/xrefb
        ;;
    *)
        expand $* | tr A-Z a-z | ${dir}/xrefa | \
                    sort '-ut   ' +0 -1 +1 -2 +2n -3n | awk -f ${dir}/xrefb
        ;;
esac
@//E*O*F fxref//
chmod u=rwx,g=rx,o=rx fxref

echo x - xrefa.l
sed 's/^@//' > "xrefa.l" <<'@//E*O*F xrefa.l//'
%k 250
%a 6100
%o 7500
%n 700
%e 450
%p 6000
%Start postfmt
%{
#include <strings.h>
#include <ctype.h>

#define NAMESIZE 64

char    *filename="-";
char    modname[NAMESIZE];
int numbymod = 0;   /* number by module or by file? */

main(argc, argv)
int argc;
```

```
char    *argv[];
{
    if (argc <= 1) {    /* input from stdin */
        numbymod = 0;
    } else {
        if(argc > 2 || strcmp(argv[1], "-m") != 0) {
            fprintf(stderr, "Usage: %s [-m] <files\n", argv[0]);
            exit(1);
        }
        numbymod = 1;
    }
    yylineno = 1;
    yylex();
    return(0);
}
%}
%%
write[ ]*([^)]+)    ;
^[cC*].*$   ;
".and"      ;
call[ ]+[_a-zA-Z0-9]+       subcal();
common[^a-zA-Z]*        ;
do[ ]*[1-9][0-9]*[ ]*[^_a-zA-Z] ;
double[ ]*precision     ;
^[ ]+end[ ]*$           ;
".eq"       ;
^[ ]*[0-9]*[ ]*end[ ]*if[ ]*$   ;
".ge"       ;
".gt"       ;
go[ ]*to[ ]*[1-9][0-9]*     ;
if[ ]*"("   ;
".le"       ;
".lt"       ;
".not"      ;
".or"       ;
real[^a-zA-Z]*              ;
return[^a-zA-Z]*            ;
^[ ]*[0-9]*[ ]*stop[ ]*[0-9][ ]*$   ;
then[^a-zA-Z]*              ;
".xor"      ;
"'"     strings();
^[ ]*[1-9][ 0-9]*format[ ]*"(".*        fndfmt();
<postfmt>.*format.*             fndfmt();
program[ ]*[_a-zA-Z0-9]+            fndef();
subroutine[ ]*[_a-zA-Z0-9]+         fndef();
function[ ]*[_a-zA-Z0-9]+           fndef();
[a-zA-Z_][a-zA-Z0-9_]*[ ]*"("           fncal();
[a-zA-Z_][a-zA-Z0-9_]*[ ]*[^a-zA-Z0-9".("]{
    /* identifier reference */
    char    *p = yytext;
    char    e = yytext[yyleng-1];

    while ( isalnum( *p) )
```

```
        ++p;
    *p = '\0';
    if ( (p - yytext) <= 6 )
        printf("%s\t%s\t%s%d\n", yytext, modname, (e == '=' ? "-" : ""),yylineno);
}
@.      ;
\n      BEGIN 0;
%%
strings()
{
    char    c;

    while (c = yyinput()) {
        if (c == '\'')
            break;
        else    if (c == '\\')
            yyinput();
    }
}

fndef()
{
/* function, or subroutine definition.  note that () are not necessary
 * in fortran.
 */
    register char   *p, *s = yytext;

    if (numbymod) {
        yylineno = 1;
    }
    /* get past reserved word */
    while ( *++s != ' ' )       /* reserved word */
        ;
    while ( *++s == ' ' )       /* get to name */
        ;
    p = s;
    while ( *s != ' ' && *s != '(' {   /* name */
        ++s;
    }
    *s = '\0';
    strcpy(modname, p);
    printf("+%s\t%s\t%d\n", p, modname, yylineno);
}

int fncal()
{
    /* functions and array references  */
    register char   *p = yytext;
    register char   *q = yytext;

    /* space to ( */
    while ( *++p != '(' )
        ;
```

```
    *p-- = '\0';
    /* backup to name */
    while( *p == ' ' ) {
        *p-- = '\0';
    }
    *++p = '(';
    /* clear leading spaces */
    while ( *q == ' ' )
        ;
    if ( ( p - q )  <= 7)
        printf("%s)\t%s\t%d\n", q, modname, yylineno);
}

int subcal()
{
/* print out name of called subroutine with ()
 */
    register char   *s = yytext;

    /* assume 'call' is first part of yytext. Skip it. */
    s += 4;

    /* skip to start of name */
    while ( *++s == ' ' )
        ;

    printf("%s()\t%s\t%d\n", s, modname, yylineno);
}

fndfmt()
{
    int c, i;

    /* skip through to the end of the format statement
     * (including continuation lines)
     * also, set start condition postfmt
     */

    BEGIN postfmt;
    c = yyinput();
    for (;;) {
        c = yyinput();      /* get the newline */
        c = yyinput();      /* check for comment */
        if (c == 'c' || c == 'C') { /* we have comment */
            while ( yyinput() != '\n')
                ;
            BEGIN 0;
            return;
        }
        /* not followed by a comment */
        for(i=2; i < 6; ++i){   /* go to continuation column */
            c = yyinput();
        }
```

```
        if ( c == ' ' || c == '0') {    /* not continued */
            break;
        }
        /* it is a continuation.  go to end of line */
        while ( c != '\n' )
            c = yyinput();
    }
}

listmatch()
{
    printf("->%s<-\n",yytext);
}
@//E*O*F xrefa.l//
chmod u=rw,g=r,o=r xrefa.l

echo x - Makefile
sed 's/^@//' > "Makefile" <<'@//E*O*F Makefile//'
CFLAGS=-O
MFLAGS=

INSTALLDIR= /.bin

all:    xrefa

xrefa:  xrefa.o
    cc -o xrefa xrefa.o -ll

clean:
    rm -f xrefa.o core

blank: clean
    rm xrefa

install: fxref all
    mv ./fxref  $(INSTALLDIR)/fxref
    mv ./xrefa $(INSTALLDIR)/xrefa
    mv ./xrefb $(INSTALLDIR)/xrefb
    chmod 755 $(INSTALLDIR)/fxref $(INSTALLDIR)/xrefa

@//E*O*F Makefile//
chmod u=rw,g=r,o=r Makefile

echo x - xrefb
sed 's/^@//' > "xrefb" <<'@//E*O*F xrefb//'
BEGIN   { OFS="\t" }

NR==1   { curident = $1 ; curmod = $2 ;
        if($3 < 0)
            curline = "=" substr($3,2) ;
        else
            curline = $3
        next
```

```
        }

curmod != $2 { print curident, curmod, curline ;
        if(substr($1, 1, 1) != substr(curident,1,1)) { print ""}
        curident = $1 ; curmod = $2 ;
        if($3 < 0)
            curline = "=" substr($3,2) ;
        else
            curline = $3
        next
        }

curident != $1 { print curident, curmod, curline ;
        if(substr($1, 1, 1) != substr(curident,1,1)) { print ""}
        curident = $1;
        if($3 < 0)
            curline = "=" substr($3,2) ;
        else
            curline = $3
        next
        }

curident == $1 {
        if(substr($3,1,1) == "-")
            curline = curline "\t=" substr($3,2) ;
        else
            curline = curline "\t" $3
        next
        }

@//E*O*F xrefb//
chmod u=rwx,g=rx,o=rx xrefb

exit 0
```

7

HARDWARE, CONFIGURATIONS, & UPGRADES

HARDWARE, CONFIGURATIONS, & UPGRADES **1221**

Software Release Levels 1221
Consulting Specials 1225
SCSI Hardware 1233
Type 4 Keyboards 1234

7

HARDWARE, CONFIGURATIONS, & UPGRADES

Software Release Levels

As of July 25, 1989

Operating Systems

Product Name	Current Release
SunOS	4.0.3
SunOS SPARCstation 1	4.0.3c
SunOS 386*i*	4.0.2

Communications Products

Product Name	Current Release
SunLink BSC3270 (SunOS 3.x)	3.0
SunLink BSC3270 (SunOS 4.x)	6.1
SunLink SCP	6.0
SunLink TE100	6.0
SunLink BSCRJE	6.0
SunLink Local 3270	6.1
SunLink SNA3270	6.1
SunLink Peer-to-Peer	6.0
SunLink IR	6.0
SunLink DDN	5.0
SunLink DNI	6.0
SunLink OSI	6.0
SunLink MCP	6.0
SunLink X.25	6.0
SunLink Channel Adapter SCA	6.0
SunLink CG3270	6.0
SunLink MHS	6.0
SunLink HSI	6.0
Notes: SunLink release 5.x products are only compatible with SunOS release 3.x. SunLink release 6.x products are only compatible with SunOS release 4.0.	

Unbundled Languages

Product Name	Current Release
Sun Modula-2 (Sun-2,3 and SunOS 3.x)	2.0
Sun Modula-2 (Sun-3,4,386*i* and SunOS 4.x)	2.1
Sun FORTRAN* (Sun-2,3)	1.0
Sun FORTRAN* (Sun-4 and Sys4-3.2)	1.05
Sun FORTRAN* (Sun-2 and SunOS 4.0)	1.1
Sun FORTRAN* (Sun 386*i* and SunOS 4.0)	1.1R
Sun FORTRAN* (Sun-3,4 and SunOS 4.0)	1.2
SPE for SCLisp 2.1	1.0
Sun Common Lisp-E	1.1
Sun Common Lisp-D	2.1
Sun Common Lisp-D (Sun-3, Sun-4)**	3.0
Cross Compilers (SunOS 3.x, Sys4-3.2)	2.0
Pascal*** (Sun-4 and Sys4-3.2)	1.05
Pascal*** (Sun-2,3,4,386*i* and SunOS 4.0)	1.1

Notes:

* The `f77` compiler is automatically included with SunOS Release 3.x, which includes SunOS Releases 3.2, 3.4, and 3.5. Sun FORTRAN 1.0 (for Sun-2,3 systems and SunOS 3.x), Sun FORTRAN 1.05 (for Sun-4 systems running Sys4-3.2), Sun FORTRAN 1.1 (for Sun-2,Sun386*i* systems and SunOS 4.0), and SunFORTRAN 1.2 (for Sun-3,4 and SunOS 4.0) are value-added products that support VMS extensions to the `f77` compiler, and must be purchased separately from the SunOS. There is no bundled FORTRAN or Pascal for Sys4-3.2 or SunOS 4.0.

** Sun Common Lisp-D release 3.0 does not obsolete Sun Common Lisp release 2.1 at this time.

*** The `pc` (Pascal) compiler is automatically included with SunOs Release 3.x, which includes Release 3.2, 3.4, and 3.5. Sun Pascal 1.05 (for Sun-4 systems) and Sun Pascal 1.1 (for Sun-2, Sun-3, Sun-4 and Sun386*i* systems running SunOS 4.0) are value-added products that support many extensions to the `pc` compiler, and must be purchased separately from the SunOS.

Unbundled Graphics

Product Name	Current Release
SunGKS	2.2.1
SunPHIGS	1.0
Sun58TE	1.0

Unbundled Applications

Product Name	Current Release
SunSimplify	1.1
SunTrac (Sun-2 and Sun-3)	1.2
SunTrac (Sun-4)	1.0/3.2
SunIPC	1.1
Transcript	2.1
SunUNIFY	3.0
PC-NFS	3.0
SunAlis	2.1
SunINGRES (Sun-2 and Sun-3)	5.1

Other Products

Product Name	Current Release
NeWS	1.1
NSE	1.1

TOPS Network Products

Product Name	Current Release
TOPS for the PC	2.1
TOPS for the Sun Workstation (Sun-3, SunOS 3.5)	2.1
TOPS for the Sun Workstation (Sun-3, Sun-4, Sun386i, SunOS 4.X)	2.2
TOPS for the Macintosh	2.1
TOPS NetPrint	2.0

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.

Consulting Specials

Sun Consulting Specials, Availability, and Compatibility

This article contains a description of Sun Consulting Specials, along with information on their availability and compatibility with existing SunOS release levels. This information is effective June 15, 1989.

Contact your local Sales office, or Sun Consulting Services, for more current information.

New, Deleted, or Changed Specials

New, deleted, or changed Sun Consulting Specials are described below. Please note that if you have a need addressed neither by the standard Sun product offerings nor by the specials described in this article, call your sales representative or Sun Consulting for a customized special to be billed on a Time and Materials (T&M) basis.

- CONSULT-FAXTOOLS

  CONSULT-FAXTOOLS contains SunView applications and a daemon which allow reception, viewing, mailing, printing and saving of facsimiles on Sun workstations. A central 'fax server', consisting of a Sun workstation and attached fax modem, can be used to deliver incoming faxes to any workstation accessible via email. It designed to work with the 'Genius Transfax' fax modem.

- CONSULT-SCANTOOL

  CONSULT-SCANTOOL is a source-level software application that provides image capture and archive capabilities using either the Eikonix 850/1412 series camera or the Sharp JX-450 color scanner. Designed for use on Sun workstations with color or grey-scale frame buffers, and interfaced to the Sun through a National Instruments IEEE-488 to VME interface, SCANTOOL provides a straightforward Sunview-based user interface.

- CONSULT-HPLJET

  CONSULT-HPLJET, a set of filters and utilities to take advantage of the capabilities of the HP LaserJet printer for SunOS 4.0 on all architectures. These programs allow users to make full use of the LaserJet, supplying command line access to set page size, orientation, fonts and feed mode for text and density, orientation (supports software rotation), centering, inverting of standard raster images. The filters are also designed to be incorporated into the print spooling mechanism of SunOS for direct *lpr(1)* access.

  With use of the manual feed mode, for instance, you can easily print to letterhead or envelopes without having to bypass the spooling

mechanism. Using the compressed font and the *pr(1)* command, two column printouts of source code on landscape paper for developers is possible. Printing of raster images is a breeze, for instance full screendumps can be rotated, centered and printed at 150 dpi for creation of overheads and documentation. Includes utilities for partial screendumps and manual pages for all programs as well as a sample `/etc/printcap` file.

- CONSULT-HSFS

  CONSULT-HSFS provides a NFS mountable High Sierra File System. Device Driver for Toshiba XM2100 CD-ROM drive is also included. This special will replace the current CONSULT-CDROM special.

- *Upgraded:* CONSULT-DISKPURGE

  CONSULT-DISKPURGE is now available as a user program under SunOS 4.0, for Sun-3 and Sun-4. This special will become obsolete with SunOS 4.0.3, since will become part of that general release.

- *Deleted:* CONSULT-ACTII, CONSULT-IMPRESS, CONSULT-CDROM

  CONSULT-ACTII and CONSULT-IMPRESS have been removed from the pricelist. CONSULT-CDROM has been replaced by CONSULT-HSFS as noted above.

Sun Consulting Compatibility

See the following tables for information regarding Sun Consulting Specials and compatibility with hardware and SunOS release levels. The first column contains the product name. The second column represents SunOS release 3.x and is applicable for Sun-3 systems. The third through seventh columns represent different hardware platforms running SunOS 4.0 or 4.0.1. These tables will be revised to include SunOS 4.0.3 in an upcoming STB issue.

An asterisk indicates that the special is available at this time on the SunOS release level and hardware platform indicated. 'NA' indicates that the product is Not Applicable for the indicated hardware platform.

Network	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	**Sun-3**	**3/50/60**	**3/1xx/2xx**	**4/1xx**	**4/2xx**	**386*i***
CONSULT-GATEWAY	*	NA	*	*	*	NA
CONSULT-PROXYARP	3.3,3.4,3.5					
CONSULT-CRAYATTACH	3.5	NA	*	NA	*	NA
CONSULT-HYPERCHANNEL	3.5	NA	*	NA	*	NA
CONSULT-XNS	3.2,3.5	*	*	*	*	
CONSULT-X25UUCP	*					

Interfaces	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-IKON85	*					
CONSULT-IKON88	*	NA	*	NA	*	NA
CONSULT-HSPEED	*	*	*	*	*	*
CONSULT-DR11W	*	NA	*	NA	*	NA
CONSULT-SIP	*	*	*	*	*	

Print Filters	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-HPLJET		*	*	*	*	*

SCSI Peripherals	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-MULTISCSI	*					
CONSULT-OPTIMEM	*	*	*	*	*	
CONSULT-HITACHI	*	*	*	*	*	
CONSULT-FLOPPY	*					

File Systems	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-DOSVFS	*					
CONSULT-HSFS		*	*	*	*	

System V Extensions	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-PLOCK	*					

I/O Devices	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-GTCO	*	*	*	*	*	*
CONSULT-SUMMAMM	*					
CONSULT-SUMMABP	*					
CONSULT-SUMMAMG	*					
CONSULT-BBAD	*	NA	*	NA	*	NA
CONSULT-BBDA	*	NA	*	NA	*	NA
CONSULT-CGONE	*					

Utilities	SunOS 3.x	SunOS 4.0/4.0.1				
Sun Consulting Specials	Sun-3	3/50/60	3/1xx/2xx	4/1xx	4/2xx	386*i*
CONSULT-TPUTIL	*	*	*	*	*	
CONSULT-DPURGE	*	*	*	*	*	
CONSULT-SECURE	*					
CONSULT-STTY	*					
CONSULT-DBXWORKS	*	*	*	NA	NA	NA
CONSULT-FAXCOMP	*	*	*	*	*	
CONSULT-EPCTOOL	*	*	*	*	*	NA
CONSULT-TE100C	*					
CONSULT-AUTODUMP	*	*	*	*	*	*
CONSULT-FAXTOOLS	*	*	*	*	*	*
CONSULT-SCANTOOL		NA	*	NA	NA	NA

Sun Consulting Specials Descriptions

This section contains a brief description of available Sun Consulting Specials. Please contact your sales representative or Sun Consulting for further information.

Network

CONSULT-GATEWAY: Multiple Ethernet Controllers

The standard 'ie' driver shipped with SunOS is configured for up to two Ethernet controllers. This software is configured for up to four Ethernet controllers so that a machine can be a gateway among more than two networks.

CONSULT-PROXYARP: Support for Subnets

Subnetting is a popular way of breaking-up a single internet network number among several physical networks (cables). SunOS release 3.3 supports subnetting, but sites may not be able to upgrade all machines at once. CONSULT-PROXYARP allows the gateway machine, running SunOS release 3.3 or subsequent releases, to 'cover' for other machines on the subnets that do not yet have subnetting support.

CONSULT-CRAYATTACH: Cray Channel Attach

This special allows a Sun to communicate with a Cray over the 100Mbit Cray channel, and to act as a gateway. TCP/IP is supported. The driver is compatible with the SunLink family of products. Required hardware includes the FEI-3 VME channel adapter, available from Cray Research.

CONSULT-HYPERCHANNEL: HYPERchannel Interface

This special allows a VMEbus-based Sun workstation to serve as a HYPERchannel Internet Protocol (IP) gateway to ethernet using an IKON 10090 board and an NSC A400. This enables HYPERchannel-connected systems conforming to RFC 1044 to communicate with ethernet-connected systems using the IP (e.g. TCP/IP).

RFC 1044 is the NSC specification for IP communications over the HYPERchannel. Several systems conform to this RFC, including Cray, VAX (VMS and BSD), Silicon-Graphics, and others.

CONSULT-XNS: XNS Networking Software

Kernel support for the XNS protocol; targeted for experienced XNS hackers. Available for SunOS releases 3.2 and 3.5 *only*.

CONSULT-X25UUCP: X.25 UUCP

This special supports the UUCP f-protocol and provides UUCP support to run on X.25 lines.

Interfaces

CONSULT-IKON85, CONSULT-IKON88: Parallel Printer Interfaces

Drivers for the Ikon 10085 (Multibus) and 10088 (VMEbus) Centronix/Versatec parallel interface boards.

CONSULT-HSPEED: High-Speed Serial Driver

High-speed line discipline (kernel driver) for up to 38.4 kbps input on the Sun serial ports.

CONSULT-DR11W: Generic DR11-W driver

The DR11-W interface is highly programmable; most applications require a custom device driver. This driver provides a suite of `ioctl` calls that

allow a user-level application to program the Ikon 10084 or 10077 DR11-W interface board.

CONSULT-SIP: SCSI Driver Starter Kit

Starter kit for SCSI drivers. Includes a skeleton driver and a "How-to" manual. This is not intended as a driver for any particular device, but is useful as a basis for development.

Print Filters

CONSULT-HPLJET: HP Laserjet Print Filters

Intelligent HP LaserJet print filters providing extended access to the capabilities of the printer. Consists of text and graphic filters with command line ability to set page size, orientation, fonts and feed mode for text and density, orientation (supports software rotation), centering, and inverting of standard raster images. Includes source, utilities for partial screendumps, manual pages for all programs and sample printcaps.

SCSI Peripherals

CONSULT-MULTISCSI: Multiple SCSI Host Adapter

Software support for additional VME SCSI host adapters. This special used to be named CONSULT-SCSI2. Unnecessary for SunOS release 4.0.

CONSULT-OPTIMEM: Optimem 1000 Driver

SCSI driver for the Optimem 1000 write-once optical disk subsystem. Raw disk support only; no file system.

CONSULT-HITACHI: Hitachi Optical Driver

SCSI driver for the Hitachi OD301-1 write-once optical disk subsystem. Raw disk support only; no file system.

CONSULT-FLOPPY: Floppy Disk Driver

SCSI driver for an IBM PC compatible floppy disk drive and file system support for a Unix (4.2 BSD) file system.

File Systems

CONSULT-DOSVFS: MS-DOS File System

SCSI driver for an IBM PC compatible floppy disk drive and file system support for MS-DOS file system.

CONSULT-HSFS: High Sierra File System

SCSI driver for CD-ROM driver and file system Support for High Sierra Format. Read-only access - no mastering capability. This special replaces the former consulting special CONSULT-CDROM.

System V Extensions

CONSULT-PLOCK: Process Locking

Ability to lock pages of text or data into physical memory.

Input and Output Devices

CONSULT-GTCO: GTCo Tablet Drivers

Three line disciplines (kernel drivers) for the GTCo Micro Digipad digitizing tablet. One driver provides fast, buffered access to tablet data; another allows the tablet to replace the Sun mouse; the third allows the digitizer to be a SunView input device in addition to the mouse.

CONSULT-SUMMAMM, CONSULT-SUMMABP, CONSULT-SUMMAMG: Summagraphics Tablet Drivers

Each line discipline emits absolute coordinates in the form of VUID mouse events for SunView consumption. CONSULT-SUMMABP and CONSULT-SUMMAMG also send an out-of-proximity event to the user. All three line disciplines also can be used to send ASCII non-SunView event data back to the individual user applications. Each line discipline supports modification to the scaling factor using the *ioctl(2)* system call. This allows users to scale different-size tablets and to modify resolutions to meet user needs.

CONSULT-BBAD: Analog Input Driver

Driver for the 12-bit MPV952 and 16-bit MPV911 A/D boards by Burr-Brown. Input can be multiplexed through as many as eight data acquisition channels.

CONSULT-BBDA: Analog Output Driver

Driver for the 12-bit MPV954 DAC board by Burr-Brown. The board provides up to eight channels of analog output.

CONSULT-CGONE: Sun-1 Color on a Sun-3

Kernel modules necessary to connect a Sun-1 medium-resolution color frame buffer to a Sun-3 running SunOS releases 3.x. Sun-2s do not need this software, even if running SunOS releases 3.x.

Utilities

CONSULT-TPUTIL: Tape Copying Utility

Copies tapes with arbitrary numbers of files with arbitrary blocking factors. Good for reproducing SunOS boot tapes.

CONSULT-DPURGE: Disk Purge

Purges data from a disk. Good for low- to medium-security applications.

CONSULT-SECURE: Secure Single-User

Requires root password to boot single-user.

CONSULT-STTY: `sttytool`

Window system tool for programming serial port controller chips.

CONSULT-DBXWORKS: dbxWorks

An enhanced version of `dbx` which provides source-level cross-debugging with Wind River Systems' VxWorks real-time executive. This replaces the standard `dbx`, and can be used for local debugging as well. `dbxtool` can also be used with this modified `dbx`.

CONSULT-FAXCOMP: CCITT Compression and Decompression

This special includes library routines that perform CCITT groups III and IV compression and decompression, used by FAX machines.

CONSULT-EPCTOOL: Enhanced `pctool`

Enhanced version of `pctool` which extend the COM ports, allowing them to be defined as tty ports, UNIX files, or sockets. Allows configuration of tty ports as raw or high-speed to allow connections with commercial data feeds. Allows specification of user defined fonts. User must have already installed the standard `pctool`.

CONSULT-TE100C: Color VT100 Terminal Emulator

Enhanced version of SunLink TE100 which adds color attributes to the emulation. Eight colors are user defined and supported in any foreground or background combination accessed by the user through additional control sequences. The special also includes a color selection utility. This special may only be run on machines licensed for standard SunLink TE100.

CONSULT-AUTODUMP: Automatic Network Disk Dump

A SunView window-based application which enables a systems administrator on a large site to plan and perform nightly dumps of file system partitions from several workstations to a single medium. The history of all dumps is recorded in a single file. It is primarily intended for usage with a high capacity storage medium such as the Exabyte 8mm video cassette shoebox.

CONSULT-FAXTOOLS: Fax Modem Tool

SunView applications and a daemon which allow reception, viewing, mailing, printing and saving of facsimiles on Sun workstations.

CONSULT-SCANTOOL: Scanner Control and Image Processing

A SunView window-based application which provides image capture and archive capabilities using either the Eikonix 850/1412 series camera or the Sharp JX-450 color scanner. Designed for use on Sun workstations with color or grey-scale frame buffers, and interfaced to the Sun through a National Instruments IEEE-488 to VME interface, Scantool provides a straight-forward SunView-based user interface.

SCSI Hardware

Sun's SCSI Hardware Implementations

This article contains an overview of Sun's SCSI hardware implementations.

Single-Ended Transmission

All of the current Sun SCSI implementations use a single-ended method of signal transmission over the cable. This is a uni-polar signal with reference to SCSI signal ground. This arrangement requires only one signal driver, as opposed to differential transmissions requiring twice the number of drivers.

Cable Lengths and Noise

Single-ended SCSI transmissions use TTL logic levels and allow cable lengths of up to 6 meters or 19.68 feet. Note that differential SCSI transmissions use RS-485 signals, allowing cable lengths of up to 25 meters or 82.02 feet.

The maximum cable length is limited since the control and data lines do not have individual differential lines. With differential cabling, there are two wires for each control or data line and the voltages on them are equal and opposite. The other end can subtract the two voltages and determine the logic.

The advantage of the differential arrangement is that any noise which causes the signal to vary up or down will cancel out in the end signal since both lines will vary equally and in opposite polarities.

The disadvantage is that differential arrangements are relatively more expensive. For this reason Sun uses single-ended cables and limits the cable length.

Type 4 Keyboards

Type 4 Keyboard Reminder

Customers and engineers in the field are reminded of an easily-overlooked hardware switch setting for type 4 keyboards.

Those installing Sun-3 machines with type 4 keyboards and SunOS 3.5 are reminded that you need to change a keyboard dip switch setting for proper operation.

Type 4 keyboards arrive from the factory set up for SunOS 4.x. This setting is used by any Sun386*i* machine, as well as Sun-3 and Sun-4 machines running SunOS 4.x.

If you use the default, factory setting when running SunOS 3.5 on a Sun-3 machine you may see `boot` fail at random points, usually after buffer sizing and before the disk checks. You may also see error messages in the default edit mode.

The READ ME FIRST

See the *READ ME FIRST*, part number 800-4093, shipped with the *Hardware Installation Manual for the Sun-3/60 Workstation*, part number 800-1987, for example.

The illustration shows the location of the dip switches, under a plastic cover panel on the right side of the underside of the keyboard. For Sun-3 machines running SunOS 3.5, the left-most dip switch 1 must be moved from lower, OFF setting to the upper, ON setting.

See figure 1 on the next page for an illustration of type 4 keyboard dip switch locations and settings.

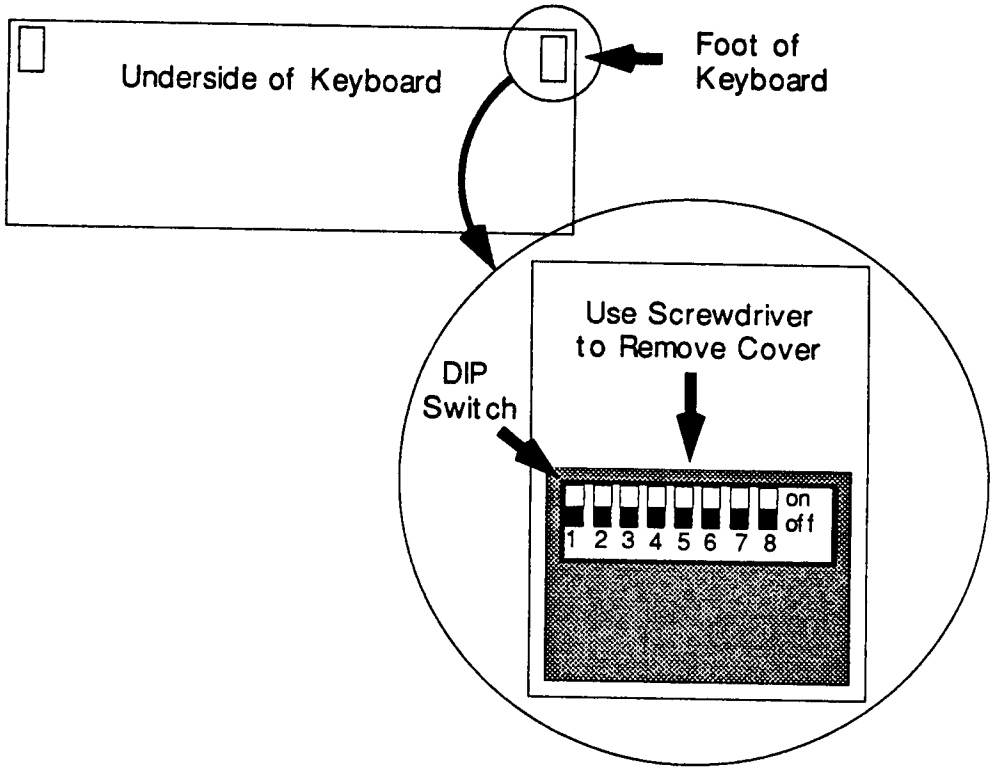


Figure 1: Type 4 Keyboard Dip Switch Locations and Settings

8

CUMULATIVE INDEX: 1989

CUMULATIVE INDEX: 1989 ..... **1239**

8

CUMULATIVE INDEX: 1989

Index

2

2000
  SunUNIFY 3.0 dates, 419

5

5210
  Micom-Interlan driver upgrade, 416

A

academic software portfolio, 254
address space
  MS-DOS emulation, 510
addresses
  classes of, 35
  Internet, 35
  network classes, 650
algorithms
  code tuning, 461
alignment
  SPARC porting issues, 266
AnswerLine, 95, 100
Apple
  TOPS, 66
application architectures, 750
  SunOS 4.1, 1046
arch(1)
  and kernel architectures, 752
architectures
  4.1 naming conventions, 1044
  application, 750
  kernel, 750, 751
  kernel and filesystems, 755
  kernel visibility, 753
  small kernels, 764
ARP, 37
AT&T
  OPEN LOOK ordering, 1101
ATbus
  Sun386*i* drivers, 510
attributes
  SunCGI-SunGKS primitives, 1185
  SunView1 and View2 comparisons, 390
auditing
  SunOS 4.0 security, 774
`automounter`
  proper YP server binding, 680
`awk`
`awk`, *continued*
  Hackers' Corner introduction, 1203

B

backups
  SNAP and symbolic links, 681
  Sun386*i* SNAP and restoring files, 1057
  Sun386*i* SunOS 4.0.1 SNAP, 784
base
  monitor size ordering, 1098
`base_devel`
  SunLink DNI 6.0 installation, 889
Beginner's Guides
  renamed User's Guides, 1062
benchmarking
  corporate, 261
benchmarks
  SDRWAVE and FORTRAN, 685
binaries
  SunOS 4.0.3 prices, 1103
boot
  `checkconfig` problems, 1056
booting
  tapeless installs, 949
broadcasting
  subnets, 35
buffer
  memory error message, 787
buffers
  Ethernet management, 86
bug
  online database, 228, 338, 494, 620, 739, 878, 1007, 1131
  reporting, 227, 337, 493, 619, 738, 877, 1006, 1130
  reporting in CSD Europe, 229, 339, 495, 621, 740, 879, 1008, 1132
  reporting in Intercon, 233, 343, 499, 625, 744, 883, 1012, 1136
  reporting in the US, 227, 337, 493, 619, 738, 877, 1006, 1130

C

C
  `dbx` debugging hints, 703
  finding zero-divides, 947
  porting to SPARC, 265
`c` partition
  whole-disk convention, 912
`c2`

c2, *continued*
SunOS 4.0 security, 768
c2conv
SunOS 4.0 security, 769
cabling
null-modems, 1095
serial I/O, 1095
cache
and device drivers, 55
flushing, 51
MC68020 on-chip, 42
overview, 44
performance and moving data, 63
Sun-3/200 and Sun-4/200, 42
tags, 50
variations, 43
virtual address, 46
caching
PC-NFS 3.0 XID, 417
call mapping
SunCGI-SunGKS, 1179
case study
SDRWAVE, 685
century
SunUNIFY 3.0 dates, 419
cg6
demonstration programs, 953
checkconfig
problems rebooting, 1056
checkmail
Hackers' Corner, 1083
checksum
Ethernet, 24
child processes
debugging with dbx, 643
classes
network addressing, 650
client side
NFS in depth, 924
client-server model
Sun386*i*, 79
cmdtool
disappearing on 3/50s, 1059
dying due to signal 1, 1059
code
tuning hints, 461
colormaps
bug 1007283, 367
flashing, 648
managing, 367
compatibility
Consulting Specials, 354, 1225
configuration
Sun386*i*, 984
Configuration Guide
Sun386*i*, 984
configurations
4.1 kernels, 1049
SCSI devices, 1037
SPARCserver 330 cable lengths, 1043
SPARCserver 330 SCSI devices, 1042
configurations, *continued*
SPARCstation 1 cable lengths, 1041
SPARCstation 1 SCSI devices, 1040
Sun-3/80 cable lengths, 1039
Sun-3/80 SCSI devices, 1038
CONSULT-PROXYARP
and Sys4-3.2 subnetting, 521
Consulting
available Specials, 354, 1225
available Sun specials, 354, 1225
Sun Germany uucico special, 259
controllers
SMD-4, 258
conventions
SunOS 4.1 naming, 1045
conversion
Sun-to-IBM FP, 469
conversions
SunView1 to View2 programs, 425
coordinates
SunGKS 3.0, 1177
courses
from Sun Educational Services, 349
cross-referencing
Hackers' Corner, 1203
CSD Europe
reporting bugs, 229, 339, 495, 621, 740, 879, 1008, 1132
cursors
colormap flashing, 648
cylinder groups
increasing inodes, 1019

D

daemons
printer, 361
troubleshooting printer, 569
data alignment
porting C to SPARC, 266
data structures
SCSI device drivers, 791
database
bugs online, 228, 338, 494, 620, 739, 878, 1007, 1131
databases
distributed, 1055
datagrams
fragmentation of, 37
reassembly of, 37
dates
SunUNIFY 3.0 and beyond 2000, 419
dbx
debugging child processes, 643
hints and tips, 703
dbxtool
hints and tips, 703
DC
SunGKS 3.0, 1177
de-support
Sun-2 languages, 415
debuggers
kernel, 246
debugging

debugging, *continued*
dbx and child processes, 643
demonstration programs
cg6, 953
demos
cg6, 953
demultiplexing
TCP/IP, 21
dependency tables, 505
errata, 1015
device coordinates
SunGKS 3.0, 1177
device drivers
4.0.3 SCSI device driver data structures, 791
4.0.3 SCSI high-level driver theory, 821
4.0.3 SCSI high-low interface, 801
4.0.3 SCSI interface example, 811
4.0.3 SCSI low-high interface, 807
4.0.3 SCSI specification, 791
and cache, 55
and kernel architectures, 758
Sun386*i* ATbus, 510
dialing
Sun386*i* and modems, 1033
differential
SCSI transmission, 1233
direct memory access
Sun386*i* ATbus drivers, 513
diskettes
used as filesystem tip, 948
disks
688MByte, 258
distributed databases, 1055
divide-by-zero
finding using dbx, 947
DMA
Sun386*i* ATbus drivers, 513
DMA channels
Sun386*i* ATbus, 510
domain system
Internet, 31
domains
multiple YP, 519
domestic kit
SunOS 386*i* 4.0.1, 635
DOS
maximum open files, 257
Windows 1.0 announcement, 1156
DOS Windows
1.0 announcement, 1156
drivers
Sun386*i* ATbus, 510
`dtree`
displaying file trees, 260

E

education
available Sun Courses, 349
catalog, 778
email
`checkmail` **Hackers' Corner**, 1083
email, *continued*
reporting bugs in CSD Europe, 229, 339, 495, 621, 740, 879, 1008, 1132
reporting bugs in Intercon, 233, 343, 499, 625, 744, 883, 1012, 1136
reporting bugs in the US, 227, 337, 493, 619, 738, 877, 1006, 1130
end of life
SunCGI and SunOS 4.1, 564
SunCORE and SunOS 4.1, 564
end-of-life plan
Sun-2 languages, 415
environment variables
SunView windows, 1170
EPS
and SunWrite, 631
errata, 1139
dependency tables, 1015
XView, 679
error logging, 776
error messages
Ethernet, 84
graphics, 785, 786, 787
`ioctl`, 787
`printcap` tips, 1077
errors
Ethernet table, 90
`ioctl #1C`, 1053
NFS write 13, 559
`es_file_read` error
`fseek`, 1058
ESDI shoebox
ordering information, 1099
Ethernet, 24
buffer management, 86
error messages, 84
error table, 90
header, 24
ethernet
maximum interfaces, 256
Ethernet
memory management, 89
panics, 90
Sun-3 hardware, 84
Ethernet addresses
Hackers' Corner, 109
External Data Representation
NFS in depth, 919

F

FCBs, 257
features
4.0.3 summary, 897
`fhandle`
NFS in depth, 921
file control blocks, 257
file handles
DOS maximum, 257
NFS in depth, 921
file locking
NFS in depth, 931
file translation

file translation, *continued*
TOPS, 74
file trees
displaying, 260
files
maximum DOS open, 257
filesystems
kernel architectures, 755
on diskettes tip, 948
`find`
displaying file trees, 260
flashing
colormaps, 648
floating point
Sun-to-IBM conversion, 469
floppy
Sun386*i* format, 911
floppy diskettes
creating Sun386*i* UNIX filesystems, 420
flushing
cache, 51
fonts
error when missing, 1053
LaserWriter II, 369
format
Sun386*i* floppy disks, 911
FORTRAN
cross-referencing in **Hackers' Corner**, 1203
dbx debugging hints, 703
finding zero-divides, 947
optimizing examples, 1199
porting hints, 291
SDRWAVE benchmark, 685
short warning message, 563
SunFORTRAN 1.2 announcement, 655
undefined `_units`, 1058
FP
Sun-to-IBM conversion, 469
FPU2
and SunFORTRAN 1.2, 655
hardware and software support, 852
fragmentation
datagrams, 37
`fseek`
`es_file_read` error, 1058
FTP, 14
function return values
porting C to SPARC, 273

G

gateways, 34
TOPS and PC-NFS, 241
Germany
`uucico` Consulting special, 259
graphics
error messages, 785, 786, 787
`ioctl` error message, 787
groups
increasing inodes, 1019
network filesystems, 891
YP, 891

H

Hacker's Corner
super kill skill, 299
Hackers' Corner
`checkmail`, 1083
Ethernet addresses, 109
`locate` script, 713
`tar -i`, 837
handles
file, 257
hardware
dependency tables, 505
Ethernet, 84
questionnaire, 849
Sun386*i* parallel port pins, 851
Hardware Technical Bulletin
hardware interest, 849
headers
IP, 23
octets, 19
overview, 21
Hints and Tips
modem installation, 95
modem problems, 100
terminal installation, 95
hotline@sun.COM
reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130
hotlines
world, 225, 335, 491, 617, 736, 875, 1004, 1128
HTB
hardware interest, 849

I

IBM
FP conversion to Sun, 469
IBM PC
TOPS, 68
ICMP, 30
`ilpr`
Interleaf to PostScript files, 915
implementation architectures
SunOS 4.1, 1047
INGRES
support transition, 561
inodes
maximum using `mkfs`, 1019
inputs
Sun-CGI-SunGKS, 1191
installation
Sun386*i* SunOS 4.0.1 remotely, 1021
installations
tapeless, 949
Intercon
hotline, 226, 336, 492, 618, 737, 876, 1005, 1129
reporting bugs, 233, 343, 499, 625, 744, 883, 1012, 1136
interface
4.0.3 SCSI example, 811
4.0.3 SCSI high-level driver theory, 821
4.0.3 SCSI high-low, 801
4.0.3 SCSI low-high, 807
interfaces

interfaces, *continued*
ethernet maximum, 256
Interleaf
to PostScript files, 915
Internet
addresses, 35
domain system, 31
protocols, 13
Internet Protocol
NFS in depth, 921
subnetting, 650
interrupt channels
Sun386*i* ATbus, 510
`ioctl`
`#1C` errors, 1053
error messages, 787
IP, 13
headers, 23
NFS in depth, 921
subnetting, 650
ISO
NFS in depth, 921

K

kernel
architectures, 750
architectures and kernel-level applications, 753
debuggers, 246
kernel architectures, 751
arch(1), 752
device drivers, 758
filesystem layouts, 755
kernel-level applications, 753
small kernels, 764
`sun3*`, 750
`sun4*`, 750
visibility, 753
kernel configurations
SunOS 4.1, 1049
kernels
small pre-configured, 764
SunOS 4.0 profiling procedure, 583
keyboards
type 4 dip switches, 1234
kill(1)
Hacker's Corner, 299
kit
SunOS 386*i* 4.0.1 domestic, 635

L

languages
Sun-2 de-support, 415
LANs
and the Sun386*i*, 1104
LaserJet II
on Sun386*i* parallel ports, 653
LaserWriter II
fonts, 369
LaserWriters
troubleshooting, 569
layering
`mail`, 19
left shifting
`textedit` bug, 1060
`lex`
Hackers' Corner introduction, 1203
line discipline
changing characteristics, 645
`lint`
use during porting, 265
Lisp
new products, 895
`locate`
Hackers' Corner script, 713
locking files
NFS in depth, 931
`lockscreen`
C2 and SunOS 4.0, 526
log
errors, 776
`login`
Sun386*i* security fix, 783
`logintool`
Sun386i security fix, 783
loopback packets
Sun386*i*, 893
`lpc`
aborting printing daemon, 361
unreliable daemon killing, 574
`lpd`
troubleshooting, 569
`lpq`
hints and tips, 1077
`lpr`
hints and tips, 1077
`ls`
displaying file trees, 260

M

Macintosh
TOPS, 66
`mail`, 15
layering, 19
routing, 33
manual pages
printing using `troff`, 418
mapping
SunCGI-SunGKS calls, 1179
maps
customized YP, 516
mass storage subsystems
ordering information, 1099
MC68020
on-chip cache, 42
memory
`textedit` window maximum, 1171
memory buffer
error message, 787
memory management
Ethernet messages, 89
Micom-Interlan 5210
driver upgrade, 416
`mkfs`

mkfs, *continued*
maximum inodes per group, 1019
Sun386i UNIX filesystems, 420
modems
Hints and Tips, 100
install Hints and Tips, 95
null-modem cabling, 1095
SPARCstation 1, 1061
Sun386*i* serial cards, 1033
monitors
base size ordering, 1098
corrupted Sun386*i* vi displays, 1197
Sun386*i*, 984
MS-DOS
address space emulation, 510
communications software, 1035
Sun386*i* and modems, 1035
multicast packets
Sun386*i*, 893

N

name servers, 1153
naming
4.1 conventions, 1045
NDC
SunGKS 3.0, 1177
netmasks
default, 650
subnetting, 650
network
address classes, 650
Network File System
client side, 924
file locking, 931
filesystem naming, 930
implementation, 928
in depth, 919
overall design goals, 920
porting experience, 936
security, 931
server side, 923
the protocol, 921
time skew, 932
versus RFS, 934
network window systems
Sun386*i*, 81
networks
filesystem groups, 891
Sun386*i*, 77
Sun386*i* windows, 81
NeWS 1.1
errata to RTF, 236
NFS, 16
client side, 924
file locking, 931
filesystem naming, 930
implementation, 928
in depth, 919
overall design goals, 920
porting experience, 936
security, 931
server side, 923
NFS, *continued*
Sun386*i*, 78
the protocol, 921
time skew, 932
versus RFS, 934
write error 13, 559
normalized device coordinates
SunGKS 3.0, 1177

O

OBD, 227, 337, 493, 619, 738, 877, 1006, 1130
Sun386*i* bugs added, 253
octets
TCP/IP headers, 19
ONC
Sun386*i*, 77
online
bugs database, 228, 338, 494, 620, 739, 878, 1007, 1131
Online Bugs Database
Sun386*i* bugs added, 253
OPEN LOOK
ordering information, 1101
STAGE products, 530
optimizing
FORTRAN examples, 1199
output primitives
SunCGI-SunGKS, 1181
overviews
Sun386*i* on networks, 77

P

packets, 24
PC-NFS 3.0 trailers, 255
padding
porting C to SPARC, 270
panics
Ethernet errors, 90
parallel port
Sun386*i* signals, 851
parallel ports
LaserJet II on Sun386*i* parallel ports, 653
parameters
passing with SPARC, 275
partitions
whole-disk c convention, 912
passing parameters
porting C to SPARC, 275
PC
TOPS, 68
PC LANs
and the Sun386*i*, 1104
PC-NFS
3.0 and trailers, 255
TOPS gateways, 241
PC-NFS 3.0
XID caching and SunOS 4.0, 417
performance
cache, 63
SunOS 4.0 hints, 283
PIO
Sun386*i* ATbus drivers, 514

plan
Sun-2 language de-support, 415
plotters
Sun386*i* serial ports, 829
PNP
Sun386*i* client install, 421
portfolio
academic software, 254
porting
FORTRAN hints, 291
SunCGI to SunGKS 3.0, 1175
tutorial, 685
ports
SPARCstation 1 serial and modems, 1061
Sun386*i* parallel and LaserJet IIs, 653
Sun386*i* serial and plotters, 829
PostScript
encapsulated (EPS), 631
from Interleaf files, 915
primitive attributes
SunCGI-SunGKS, 1185
primitives
SunCGI-SunGKS output, 1181
`printcap`
hints and tips, 1077
printer
aborting daemon, 361
printers
LaserJet II on Sun386*i* parallel ports, 653
troubleshooting, 569
printing
manual pages using `troff`, 418
`printcap` hints and tips, 1077
procedures
SunOS kernel profiling, 583
processes
debugging children with `dbx`, 643
products
release levels, 223, 333, 489, 615, 734, 873, 1094, 1224
profiling
SunOS 4.0 kernel procedure, 583
programmed I/O
Sun386*i* ATbus drivers, 514
programs
converting SunView1 to View2, 425
porting C to SPARC, 265
pseudo teletype
example program, 587
`pstty`
changing characteristics, 645
`ptys`
pseudo example program, 587

Q

questionnaire
hardware interest, 849
queue
window error messages, 785

R

`read`
`suntools` error, 1172
real time
Sun386*i* SunOS, 1095
reassembly
datagrams, 37
reboot
`checkconfig` problems, 1056
releases
software products, 223, 333, 489, 615, 734, 873, 1094, 1224
Remote File System
versus NFS, 934
remote installation
Sun386*i* SunOS 4.0.1, 1021
Remote Procedure Call
NFS in depth, 921
reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130
resets
watchdog, 246
restoring files
Sun386*i* SNAP, 1057
return values
porting C to SPARC, 273
RFS
versus NFS, 934
RGB
colormap flashing, 649
routing
`mail`, 33
RPC
NFS in depth, 921
source availability, 1029
RPCSRC 4.0
availability, 1029
RTF
NeWS 1.1 errata, 236
RTI
INGRES support, 561
Rutgers University, 13

S

scalar
algorithm and code tuning, 461
SCCS
installation, 638
simplified operations, 638
`sccs`
the command, 639
SCSI
4.0.3 device driver specification, 791
4.0.3 high-level driver theory, 821
4.0.3 high-low interface, 801
4.0.3 interface example, 811
4.0.3 low-high interface, 807
configuration tables, 1037
SPARCserver 330, 1042
SPARCserver 330 cable lengths, 1043
SPARCstation 1, 1040
SPARCstation 1 cable lengths, 1041
specification data structures, 791

SCSI, *continued*
Sun hardware implementations, 1233
Sun-3/80, 1038
Sun-3/80 cable lengths, 1039
SDRWAVE
case study, 685
security
C2, 768
NFS in depth, 931
Sun386*i* SunOS 4.0.1, 783
SunOS 4.0, 767
SunOS 4.0 `c2conv`, 769
SunOS enhancement, 1167
`sendmail`
expansion fix, 560
serial cards
Sun386*i* modems, 1033
serial I/O
null-modem cabling, 1095
Sun386*i* SunOS, 1095
serial port
changing characteristics, 645
serial ports
SPARCstation 1 modems, 1061
Sun386*i* and plotters, 829
server side
NFS in depth, 923
servers
multiply YP, 518
name, 1153
`share` files
SunOS 4.1, 1049
shifting left
`textedit` bug, 1060
shoebox
ESDI ordering information, 1099
short
FORTRAN warning message, 563
silo overflow
error messages, 786
slay
killing `lpd`, 574
small kernels, 764
SMD-4 controllers, 258
SMTP
application example, 28
SNAP
Sun386*i* YP servers, 680
Sun386*i* client install, 421
Sun386*i* SunOS 4.0.1 backups, 784
symbolic links and backups, 681
`SOCK_RDM`
unimplemented socket, 913
`SOCK_SEQPACKET`
unimplemented socket, 913
sockets
unimplemented, 913
well-known, 25
software
dependency tables, 505
source
source, *continued*
SunOS 4.0.3 prices, 1103
SPARC
porting C programs, 265
SPARCserver 330
cable lengths, 1043
SCSI configurations, 1042
SPARCstation 1
cable lengths, 1041
modems, 1061
SCSI configurations, 1040
Specials
compatibility, 354, 1225
specials
Sun Consulting, 354, 1225
specification
4.0.3 SCSI data structures, 791
4.0.3 SCSI device drivers, 791
4.0.3 SCSI high-level driver theory, 821
4.0.3 SCSI high-low interface, 801
4.0.3 SCSI interface example, 811
4.0.3 SCSI low-high interface, 807
STAGE
OPEN LOOK, 530
product announcements, 530
SunDraw, 549
SunPaint, 542
SunWrite, 534
STB
duplication of, 235, 345, 501, 627, 746, 885, 1014, 1138
STREAMS
resources, 528
SunOS 4.0, 239
structures
SCSI device driver data, 791
stty(1)
changing characteristics, 645
subnets
broadcasting, 35
subnetting, 650
and SunOS Sys4-3.2, 521
restrictions, 651
Sun Academic Software Portfolio
Sun Academic Software Portfolio, 254
Sun Common Lisp
new products, 895
Sun Consulting
available specials, 354, 1225
Sun Education
catalog, 778
Sun Educational Services
available courses, 349
Sun workstations
TOPS, 69
sun!hotline
reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130
sun!stb-editor, 95, 100, 235, 345, 501, 627, 746, 885, 1014, 1138
sun!sunbugs
reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130
Sun-2
language de-support, 415

Sun-2, *continued*
last SunOS supported, 749
Sun-3/50s
disappearing `cmdtool`, 1059
Sun-3/80
cable lengths, 1039
SCSI configurations, 1038
Sun-4/110 TC
true color representation, 649
`sun3`
4.1 architecture, 1044
kernel architecture, 750
Sun386*i*
and PC LANs, 1104
ATbus drivers, 510
binding to YP servers, 680
bugs added to OBD, 253
configuration, 984
corrupted `vi` displays, 1197
floppy format, 911
installing SunUNIFY 3.0, 365
loopback/multicast packets, 893
network overview, 77
NFS, 78
ONC, 77
parallel port pins, 851
serial cards and modems, 1033
serial ports and plotters, 829
SNAP backups and restoring files, 1057
SNAP backups and symbolic links, 681
SunLink DNI 6.0 installation, 889
SunOS 386*i* 4.0.1 domestic kit, 635
SunOS 4.0.1 overview, 523
SunOS 4.0.1 performance, 562
Sun386*i* SunOS
4.0.1 security, 783
Sun386*i* SunOS 4.0.1
remote installation, 1021
SNAP backups, 784
Sun386*i* SunOS 4.0.2
announcement, 1143
`sun3x`
4.1 architecture, 1044
kernel architecture, 750
`sun4`
4.1 architecture, 1044
kernel architecture, 750
`sun4c`
4.1 architecture, 1044
kernel architecture, 750
sunbugs@sun.COM
reporting bugs, 227, 337, 493, 619, 738, 877, 1006, 1130
SunCGI
call mapping to SunGKS, 1179
end-of-life and SunOS 4.1, 564
inputs, 1191
output primitives, 1181
porting to SunGKS 3.0, 1175
primitive attributes, 1185
SunCORE
end-of-life and SunOS 4.1, 564
SunDraw 1.0
SunDraw 1.0, *continued*
announcement, 549
STAGE product, 530
SunFORTRAN
1.2 announcement, 655
SunGKS
3.0 porting from SunCGI, 1175
3.0 workstations, 1176
call mapping to SunCGI, 1179
inputs, 1191
new concepts, 1176
output primitives, 1181
primitive attributes, 1185
SunGKS 2.2.1
announcement, 566
SunINGRES
support transition, 561
SunLink
4.0.3 upgrade bug, 1054
SunLink DNI 6.0
installation, 889
SunOS
3.5 and type 4 keyboards, 1234
386*i* 4.0.1 domestic kit, 635
386*i* performance, 562
386*i* 4.0.1 remote installation, 1021
4.0 `c2conv`, 769
4.0 security, 767
4.0 security auditing, 774
4.0.3 announcement, 749
4.0.3 feature summary, 897
4.0.3 upgrade bug, 1054, 1168
4.0.3 upgrade paths, 853
4.1 and SunCGI end-of-life, 564
4.1 and SunCORE end-of-life, 564
4.1 Beginner's Guides renamed, 1062
4.1 directory layout, 1044
4.1 kernel configurations, 1049
4.1 naming conventions, 1045
C2 4.0 security, 768
determining version, 363
security enhancement, 1167
SPARCstation1 OS prices, 1103
Sun-2 language de-support, 415
Sun386*i* 4.0.1 overview, 523
Sun386*i* 4.0.1 security, 783
Sun386*i* 4.0.2 announcement, 1143
Sun386*i* null-modem cabling, 1095
Sun386*i* serial I/O, 1095
Sys4-3.2 and subnetting, 521
Sys4-3.2 tapeless installs, 949
SunOS 3.5
type 4 keyboard settings, 1234
SunOS 3.x.x
finding Ethernet addresses, 109
SunOS 4.0
C2 `lockscreen`, 526
error logging differences, 776
finding Ethernet addresses, 109
kernel profiling procedure, 583
PC-NFS 3.0 XID caching, 417
performance hints, 283
starting `suntools`, 1169

SunOS 4.0, *continued*
STREAMS, 239
SunOS 4.0.3
announcement, 749
feature summary, 897
SCSI device driver data structures, 791
SCSI driver specification, 791
SCSI high-level driver theory, 821
SCSI high-low interface, 801
SCSI interface example, 811
SCSI low-high interface, 807
SPARCstation1 OS prices, 1103
upgrade bug, 1054, 1168
SunOS 4.1
Beginner's Guides renamed, 1062
directory layout, 1044
kernel configurations, 1049
naming conventions, 1045
SunOS Sys4-3.2
tapeless installs, 949
SunPaint 1.0
announcement, 542
STAGE product, 530
SunSimplify
installing with SunUNIFY 3.0, 365
`suntools`
missing font errors, 1053
`read` errors, 1172
starting under 4.0, 1169
SunUNIFY 3.0
bug 1016117, 365
dates beyond 2000, 419
documentation updates, 365
installing on Sun386*i*, 365
sunupgrade(8)
bug, 1168
bug with SunLink, 1054
SunVideo
announcement, 1065
applications, 1070
features, 1066
hardware, 1067
software, 1069
SunView
missing font errors, 1053
window error, 1170
SunView1
attribute comparison with View2, 390
converting programs to View2, 425
SunWrite
and EPS, 631
SunWrite 1.0
announcement, 534
STAGE product, 530
survey
hardware interest, 849
symbolic links
SNAP backups, 681
Sys4-3.2
and subnetting, 521

T

TAAC
software release 2.3, 1031
true color representation, 649
TAAC 2.3
announcement, 1031
right to use, 1032
tables
hardware/software dependencies, 505
software release levels, 223, 333, 489, 615, 734, 873, 1094, 1224
tags
cache, 50
tapeless installations, 949
`tar`
damaged tapes, 837
Hackers' Corner, 837
scavenging files, 837
`tcopy`
`tputil` Consulting Differences, 244
TCP, 13
TCP/IP
demultiplexing, 21
references, 38
teletype
pseudo example program, 587
TELNET, 14
terminals
install Hints and Tips, 95
`textedit`
left shifting bug, 1060
maximum window memory, 1171
theory of operation
4.0.3 SCSI high-level drivers, 821
time skew
NFS in depth, 932
TOPS
file translation, 74
for Apple Macintosh, 66
for IBM PC, 68
for Sun workstations, 69
installation requirements, 75
Macintosh operation, 71
PC operation, 72
PC-NFS gateways, 241
product overview, 66
technical support, 243
use of, 71
`tputil`
Consulting special, 244
copying SunOS release tapes, 244
trailers
with PC-NFS 3.0, 255
TranScript
troubleshooting, 569
translation
TOPS files, 74
transmission
SCSI single-ended, 1233
trees
displaying files, 260
`troff`

`troff`, *continued*
printing manual pages, 418
troubleshooting
Ethernet errors, 90
LaserWriters, 569
TranScript, 569
true color representation
colormap flashing, 649
tuning
coding hints, 461
tutorial
porting, 685
type 4 keyboards
switch settings and SunOS 3.5, 1234

U

UDP, 30
NFS in depth, 921
Ultrix 2.x
with PC-NFS 3.0 trailers, 255
undefined `_units`
workaround, 1058
upgrades
4.0.3 and bug, 1168
4.0.3 and SunLink bug, 1054
paths to SunOS 4.0.3, 853
User Datagram Protocol
NFS in depth, 921
User's Guides
SunOS 4.1, 1062
`uucico`
Sun Consulting Germany special, 259
`uucp`
Sun Germany `uucico` special, 259

V

variables
SunView environment, 1170
VAX
and PC-NFS 3.0 trailers, 255
vector
algorithm and code tuning, 461
`version`
determining SunOS level, 363
VFS
NFS in depth, 919, 925
`vi`
corrupted Sun386*i* display, 1197
video
SunVideo announcement, 1065
SunVideo applications, 1070
SunVideo features, 1066
SunVideo hardware, 1067
SunVideo software, 1069
View2
attribute comparison with SunView1, 390
converting programs from SunView1, 425
XView errata, 679
virtual address
cache, 46
Virtual File System
Virtual File System, *continued*
NFS in depth, 919, 925
VMEbus
custom boards, 914
Revision B and Sun-3s, 914

W

watchdog resets, 246
WC
SunGKS 3.0, 1177
well-known sockets, 25
windows
colormap flashing, 648
Windows
DOS 1.0 announcement, 1156
windows
error messages, 785
maximum `textedit` memory, 1171
SunView error, 1170
workstation
SunGKS 3.0, 1176
workstations
TOPS, 69
world coordinates
SunGKS 3.0, 1177
world hotlines, 225, 335, 491, 617, 736, 875, 1004, 1128
write errors
NFS 13, 559
write-back cache, 43
write-through cache, 43

X

X.25
usage hints, 289
XDR
NFS in depth, 919
porting C to SPARC, 272
XID
PC-NFS 3.0 and SunOS 4.0, 417
XNS
NFS in depth, 921
XView
errata, 679

Y

yellow pages
customized maps, 516
hints, 516
multiple servers, 518
name servers, 1153
serving multiple domains, 519
YP
customized maps, 516
filesystem groups, 891
hints, 516
multiple servers, 518
name servers, 1153
serving multiple domains, 519
YP servers
Sun386*i* binding, 680
ypset(8)

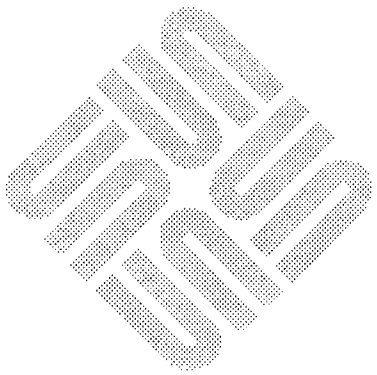
ypset(8), continued
fix available, 565

Z

zero-divides
finding using dbx, 947

Revision History

Revision	*Date*	*Comments*
FINAL	Septebmer 1989	Ninth issue of the 1989 Software Technical Bulletin, developed by Technical Information Services (TIS), WWFO Technical Support Services (TSS), Sun Microsystems, Inc.



Systems for Open Computing™

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

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

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

Australia: (02) 413 2666
Canada: 416 477-6745
France: (1) 40 94 80 00
Germany: (089) 95094-0
Hong Kong: 852 5-8651688
Italy: (39) 6056337
Japan: (03) 221-7021
Korea: 2-7802255
Nordic Countries: +46 (0)8 7647810
PRC: 1-8315568
Singapore: 224 3388
Spain: (1) 2532003
Switzerland: (1) 8289555
The Netherlands: 3133501234
Taiwan: 2-7213257
UK: 0276 62111

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

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