sun microsystems

Software Technical Bulletin
October 1989

Technical Information Services

Part Number 812-8910-01
Issue 1989-10
October 1989

sun microsystems

Software Technical Bulletin
October 1989

Technical Information Services

Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415-960-130

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.

Sun-2, Sun-2/xxx, Sun-3, Sun-4, Deskside, SunStation, Sun Workstation, SunCore, SunPHIGS, DVMA, SunWindows, NeWS, NFS, NSE, SPARC™, SunUNIFY™, SunView™, SunGKS, SunCGI, SunGuide, SunSimplify, SunLink, Sun Microsystems, SunOS™, TOPS®, Flashcaard™, SunPaint™, SunWrite™, SunDraw™, TOPS Terminal, View2, and the Sun logo are trademarks of Sun Microsystems, Inc. UNIX, UNIX/32V, UNIX System III, UNIX System V, and OPEN LOOK are trademarks of AT&T Bell Laboratories.
DEC, DNA, VAX, VMS, VT100, WPS-PLUS, MicroVAX, and Ultrix are registered trademarks of Digital Equipment Corporation. Courier 2400 is a trademark of U.S. Robotics, Inc. Hayes is a trademark of Hayes Microcomputer Products, Inc. Multibus is a trademark of Intel Corporation. PostScript and TranScript are trademarks of Adobe Systems, Inc. Ven-Tel is a trademark of Ven-Tel, Inc. UNIFY™ is a trademark of Unify Corporation. ENTER, PAINT, ACCELL, and RPT are trademarks of Unify Corporation. IBM, IBM PC, PC, IBM PC/XT, IBM PC/AT, and SQL™ are registered trademarks of International Business Machines Corporation. Applix® is a registered trademark of Applix, Inc. SunAlis™ is a trademark of Sun Microsystems, Inc. and is derived from Alis, a product marketed by Applix, Inc. SunINGRES™ is a trademark of Sun Microsystems, Inc. and is derived from INGRES, a product marketed by Relational Technology, Inc. VEGA Delux is a trademark of Video Seven, Inc. Micro Enhancer Delux is a trademark of Everex Systems, Inc. VxWorks is a trademark of Wind River Systems, Inc. Cabletron is a trademark of Cabletron Systems. Apple, Finder, Macintosh, Appletalk, MacWrite, and Laser Writer™ are registered trademarks of Apple Computer, Inc. PostScript® is a registered trademark of Adobe Systems, Inc. Excel, MS-DOS, and Microsoft Word are registered trademarks of Microsoft Corporation. Fastpath is a trademark of Kinetics, Inc. Ethernet is a registered trademark of Xerox Corporation. Lotus 1-2-3 is a trademark of Lotus Development Corporation. Word Perfect is a trademark of the Word Perfect Corporation. Wyse-50 is a trademark of Wyse Technology Corporation. AMD 7990 LAN Controller is a trademark of Advanced Micro Devices, Incorporated. Proximity (R) is a registered trademark of Proximity Technology, Inc. Merriam-Webster (R) is a registered trademark of Merriam-Webster Inc.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

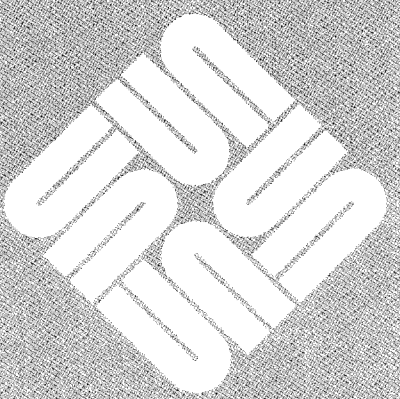
Copyright (c) 1989, Sun Microsystems, Inc. Printed in U.S.A. All Rights Reserved. No part of this work covered by copyright hereon may be reproduced or used in any form or by any means -- graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems -- without permission of the copyright owner.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

Contents

1

NOTES & COMMENTS



1

NOTES & COMMENTS

Editor's Notes

Editor's Notes

The editor's notes for this October 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: `cgfour` and moving windows faster, and using `enscript` on Sun386*i* machines
- The Hackers' Corner: the `mush` Mail Utility
- Configurations: updated software release level tables, effective August 25, 1989
- Product Dependency Tables

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 new items of interest. The first is a hint on how to move windows faster on Sun-3 workstations using `cgfour` framebuffers. This hint works for OpenWindows (not for SunView).

The second hint is how to print in landscape mode from DOS on Sun386*i* machines using `enscript`.

The Hackers' Corner

This month's **Hackers' Corner** contains an introduction to the `mush` mail utility. Similarities and differences to the regular mail utility are summarized. The `mush` code does not appear, since it is over 30 pages long.

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.

Product Dependency Tables

Expanded product dependency tables appear in the Hardware, Upgrades, & Dependencies section. These tables summarize software and hardware product interdependencies for Sun-4, Sun-3, Sun-2, and Sun386*i* platforms.

These tables are published quarterly and are expanded for this quarter to reflect all SunOS release levels appropriate for the families of hardware platforms.

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) 477-6745
	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 their local support center. 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 engineering and support personnel. An internal bug report is generated if the reported bug is new and verifiable.

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

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.

- Keyword(s)
- 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 their local support center. 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.

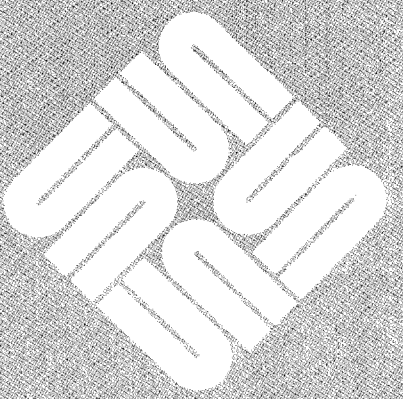
Look to this column for an announcement regarding the purchase of extra STB copies.

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*.

2

ARTICLES



2

ARTICLES

OBD Change Notes

Online Bugs Database Change Notes

The Online Bugs Database (OBD) is a tool for disseminating information pertaining to Sun's known software bugs. This tool is a special feature available to all customers holding software support contracts.

The OBD allows you to query a specified bug by *bug reference number*, *category*, *subcategory*, *release*, and *keyword(s)* through a search facility. Once a particular bug report is located, detailed information about the bug is given, along with a temporary workaround (when available).

Using the OBD can save time and energy by eliminating the need to interact with customer support. Information can be quickly retrieved from the OBD. Whenever possible, customers should use the OBD to find the workaround for a problem before contacting the U.S. Answer Center,[2] and to avoid resubmitting duplicate bug reports.

For further information, see the subheading entitled 'The Online Bugs Database (OBD)' in the *Reporting Bugs* note in Section I of this STB.

OBD Change Summary

The Online Bugs Database changes reflect a software upgrade from SunUNIFY 2.0 to SunUNIFY 3.0. These changes corrects some SunUNIFY 2.0 bugs, as well as add a new `keyword(s)` search facility. Note that the OBD upgrade requires no action on your part.

---

[2] The OBD is currently available only in the U.S. Future plans include worldwide introduction and distribution.

OBD Keyword(s) Search Criteria

The OBD multiple search strategies have been expanded to include search by `keyword(s)` and `category/subcategory` within a specified `keyword(s)` . These changes appear on the *Search by Detail* screen. The following help screen has been added for the `keyword(s)` search:

```
==========================================================================
                          Selecting Keyword(s)

        To EXIT this help file:     q <CR>
        For HELP:    h

        To select keyword(s):

            Enter keyword(s) at the Keyword prompt.

          NOTE:  Enter keyword(s) in lower case
==========================================================================
```

Customer Distributed BugsList

The OBD was designed to provide convenient online access to bugs. The Customer Distributed BugsList (CDB) is also available as a hardcopy reference to known bugs.

Sun386*i* 4.0.2 Telemarketing

U.S. Telemarketing Ordering of Sun386*i* SunOS 4.0.2

Shipments of the Sun386*i* SunOS 4.0.2 upgrade, beginning in the fall of 1989, are available to U.S. customers using a quick-turnaround telemarketing program for those running Sun386*i* SunOS 4.0.1. Customers holding software service contracts receive this upgrade automatically. This program has a cutoff date of October 31, 1989. Please contact your sales representative after that date.

Sun is offering the telemarketing approach to get the Sun386*i* SunOS 4.0.2 release to as many customers as quickly as possible. The upgrade is priced at $100 for a floppy diskette or a tape, plus an updated Owner's Set documentation.

1-800-553-4265

To use the telemarketing ordering program, simply do the following:

- Call **1-800-553-4265**
- Order either the diskette or tape media
- Use your VISA or MasterCard
- Shipment will follow within 24 hours via Federal Express two-day service

Note that this program is implemented by a third party (Telespectrum, Inc.) on contract to Sun Microsystems.

Upgrade Package Documentation and Contents

Available with Sun386*i* SunOS 4.0.2 is a new and improved Owners Set. The Owner's Set now includes a guide to all Sun386*i* documentation and a complete index of the four manuals in the set. The title of this booklet is *Sun386i Owner's Set Index.*

The following items are included in the Sun386*i* SunOS 4.0.2 upgrade package:

- Upgrade diskette set (eight 3-1/2") or tape
- Updated Owner's Set documentation (four PC-style user guides and administration guides)
- Master Index
- Installation notes for Sun386*i* SunOS 4.0.2

- Updated Owner's Bulletin (release notes), including a summary of major performance and quality improvements
- Updated Administrators and Developer's Notes

Installing Sun386*i* SunOS 4.0.2

Installing the upgrade takes between 30 and 60 minutes, depending on your machine configuration. It is an 'incremental' upgrade, and does not require a complete reinstallation of the operating system. It can also be done on an as-needed basis; various systems on a network can run Sun386*i* SunOS 4.0.1 and 4.0.2 without conflict.

C++ 2.0 Announcement

C++ Release 2.0 Announcement

This article announces the availability of C++ release 2.0 for Sun-3 and Sun-4 systems, running Sun Operating System (SunOS) release 4.0 or greater, with a minimum of 4 Mb of memory.

This article provides a brief overview of C++ release 2.0. Detailed information regarding the C++ language will be presented in next month's *Software Technical Bulletin.*

Introduction to C++

C++ is designed as an extension of the C programming language. C++ retains C's facility for efficient low-level programming, and adds the following:

- Stronger type-checking
- Extensive data abstraction features
- Support for object-oriented programming

This last feature allows for good design of modular, extensible interfaces among program variables.

Sun C++ supports C++ as described in *The C++ Programming Language* by Bjarne Stroustrup, with a few deletions and a number of extensions. Those deviations from *The C++ Programming Language* are briefly described below. The *AT&T C++ Language System Reference Manual* is included in the C++ documentation set, and covers the current version of the language.

Compatibility with C

C++ is almost entirely compatible with C. The language was purposely designed for compatibility, which enables an experienced C programmer to learn C++ at his or her own pace, and incorporate features of the new language when appropriate. Note that what is new about C++ is intended to supplement what is good and useful about C; most importantly, C++ retains C's efficient interface to the hardware of the computer, including types and operators that correspond directly to components of computing equipment.

C++ does have some important differences with C; as such, an ordinary C program probably won't be accepted by the C++ translator without prior modifications. Another key difference is that even though the differences between C and C++ are most evident in the way the user can design interfaces between program modules, C++ retains all of C's facilities for designing such interfaces. For example, C++ modules can be linked to C modules, which allows you to use C libraries wit C++ programs.

The C++ Translator and Translator Script

C++ provides a translator script `CC` to process C++ programs. `CC` invokes the processor `cpp`, then the C++ translator `cfront` to translate C++ source code to C source code. By default, `CC` then invokes the Sun C compiler and the link editor `ld`. Thus, when given a complete program, the C++ translator script can produce an executable program.

The Sun C++ translator is based on the AT&T C++ translator, version 2.0 GA. Its action is identical to the AT&T C++ translator release 2.0, except that the Sun C++ translator can include additional information in the executable file to allow improved debugging.

The Translator Package

Aside from the translator `cfront` and the translator script `CC`, the translator package includes improved versions of `dbx` and `dbxtool`, the Sun debuggers, for use with C++ programs. It also includes upgraded C language tools, such as `ctags++`, `nm++`, `prof++`, `gprof++`, `yacc++`, `lex++`, and `rpcgen++`. `cpp` has also been enhanced to process the C++ comment token `//`. The translator package also includes C++ library functions for stream I/O, complex arithmetic, tasking, and other operations. In addition, it has `#include` files that allow users to use standard UNIX features, such as signals and ctype, with C++ programs, and to interface with SunView1 window system libraries, such as `libsuntool`.

Key Additions to C++

Key additions to C++ include the following topics:

- Type Checking
- Classes and Data Abstraction
- Object-Oriented Features
- Other Differences from C
- Use Within the Network Software Environment (NSE)

Type Checking

A compiler or interpretor performs *type checking* when it ensures that operations are applied to data of the correct type. C++ has stronger type checking than C, though not as strong as with Pascal. The approach to type checking is different from the approach in languages like Pascal. Whereas Pascal always objects to attempts to use data of the wrong type, the C++ translator objects only in some cases, and in other cases, converts data to the correct type. Rather than allowing the translator to do these automatic conversions, the user can explicitly convert between types, as is possible in C.

A related area involves *overloaded function names.*. In C++, you can give any number of functions the same name. The translator decides which function should be called by checking the types of the parameters of the function call. (Users may notice that this could lead to ambiguous situations. If the resolution is not clear at translation time, the translator issues an 'ambiguity' error.)

Classes and Data Abstraction

A *class* is a user-defined type. Like the pre-defined types (and unlike user-defined types in languages such as Pascal), classes are defined not only with data storage, but with operations that apply to the new type. In C++, these operations include operators and functions. For example, if a class is defined as `class`, the `+` operator can be defined so it has a meaning when used with `class`. Thus, the expression

```
class + class2
```

has a value determined by the definition of `+` in the case of `class`. Note that this does not override the original definition of `+`; as with overloaded function names, the translator determines from context which definitions of `+` it should use. (Note that operators with extra definitions such as this are called *overloaded operators*.)

In addition to operators, classes may have *member* functions, which are functions that exist to operate on objects of that class.

C++ provides classes as a means for *data abstraction*. Programs that are designed with data abstraction are designed by deciding what types (that is, classes) are wanted for the program's data, then deciding what operations each type needs.

The members of a class can be divided into `public`, `private`, and `protected` parts. The public part is available to any function; the private part is available only to member and friend functions; the protected part is available to members, friends, and members of derived classes.

Object-Oriented Features

A program is *object-oriented* when the program is designed with classes, and the classes are organized so that common features are embodied in *base* classes. Base classes are sometimes called *parent* classes. The feature that makes this possible is *inheritance*. A class in C++ can inherit features from one base class or from several classes. A class that has a base class is said to be *derived* from the base class.

The greatest utility of this idea is in extending existing programs or libraries; you can define a new descendant that differs from its parent in some way that was not imagined when the parent class was designed. For example, if a class defines a kind of window with scroll bars and the user later wishes to implement windows with a different kind of scroll bar, the user can create a descendant of the original window class and simply change the implementation of the scroll bar functions without reimplementing or even examining the implementation of other parts of the program.

Other Differences from C

C++ differs from C in a number of other details, as briefly highlighted below.

- Defined constants in C++ permit the user to avoid using the preprocessor to use named constants in a program.

- Default values for function parameters are not used in C++. The user must generally specify function parameter types.
- C++'s free store operators `new` and `delete` create dynamic variables.
- C++ includes references, which are alternate "handles" on the same object. A reference is an automatically dereferenced pointer, and acts like an alternate name for a variable. References can be used as function parameters.
- C++ includes functional syntax for type coercions.
- C++ permits programmer-defined automatic type conversion.
- Variable declarations are allowed anywhere within code, not just at the beginning of a block.
- A new comment delimiter begins a comment that continues to the end of the line.
- The name of an enumeration or class is also automatically a type name.
- Declarations can be placed within blocks.
- Default values can be assigned to function parameters.
- Inline functions allow the user to ask the translator to replace a function call with the function body, thus improving program efficiency.

Use Within the Network Software Environment (NSE)

C++ is completely compatible with Sun's Network Software Environment (NSE) release 1.2. The default filename extension for C++ source files under the NSE is `.cc`. If you would prefer to use a filename extension other than `.cc`, refer to the *Network Software Environment: Administration Guide* for further information.

Netgroups and NFS Access

Netgroups, Trusted Host Capability, and Limiting Access

There is some confusion regarding the proper the use of `/etc/netgroups` for achieving trusted host capability, and for limiting access on exported NFS file systems.[3] Netgroups contain descriptions of valid members who can either access certain NFS file systems, or `rlogin` to systems without the need of a password.

Netgroups and a Faulty Access Assumption

A netgroup generally consists of triples such as (*host,user,domain*) . The natural implication is that any user that is a valid member of one of these triples will have trusted access, and any user that is not a valid member will not have trusted access.

The confusion occurs when the assumption is made that a triple specified as (,,*domain*) will limit access to any machine/user within the specified domain.

True User Domains Not Discernible

An anomaly exists in that the true domain of any machine/user is not discernible. Therefore, it is not possible to exclude those members who are not of the specified domain.

As an example, if the netgroup 'sales' is defined as `(,,sales)` and an entry is placed in `/etc/hosts.equiv` of `+@sales`, one would think that only those machines/users in domain 'sales' would have trusted host capability. However, since it is not possible to determine the domain of any given machine/user, *all* machines/users will get trusted host capability.

A triple definition of `(,,domain)` has the same affect as `(,,)` when the server's domainname is 'domain'. The effect being, that *everyone* will get trusted access.

However, if the server's domainname is not 'domain' then a single triple definition `(,,domain)` effectively *denies access to all* users and systems.

In other words, to have the netgroup domain field either empty or containing the server's own domain will *give access to all* hosts/users. To have the domain field contain anything other than the server's own domain or `NULL` will *deny access* to all hosts/users.

Netgroups and Limiting Access

Users must be careful not to use the netgroup domain field definition if the intent is to restrict access to a particular set of members.

The correct way to limit access to a specific set of members, is to explicitly specify the machine and user in each triple.

---

[3] This article is submitted by Albert Lopez and Eric Voss, Data Comm Group, U.S. Answer Center, Mountain View, California, USA.

The domain field of netgroups has no current use for network security or administration. Networking commands such as `mount`, `rlogin`, `rexec`, and `rsh` (there may be more), do not pass any parameter indicating which domain from which it is coming. Therefore, there is no way for the server to verify from which domain the client is coming. The server assigns its own domainname to any system which attempts to connect to it, and then checks the netgroup for verification.

`syslogd` Configurations

Configuring the System Logging Daemon

The information contained in this article is contained in the most recent version of the *SunOS 4.0.3 READ THIS FIRST* (RTF), part number 800-3816-11. This information is of use to those having earlier SunOS 4.0.3 RTF versions.

`syslogd` Confusion

Since SunOS 4.0 first shipped there has been confusion concerning the configuration of the system logging daemon `syslogd` . This confusion was heightened by the existence of a bug in the `syslogd` initialization code. Some confusion remains despite the bug fix appearing in SunOS 4.0.3.

The bug is reference id number 1010651: `syslogd` fails to define loghost name, causes `syslog` race.

The Problem Defined

Upon start up and whenever `syslogd` receives an HUP signal, `syslogd` reads the configuration file `/etc/syslog.conf` to determine the proper disposition of various types of system messages.

Sample scenarios and the means of implementing them are shown below. Please note that the following examples assume that you are running SunOS 4.0.3.

Solution 1

If you wish messages to be logged to a centralized loghost machine and you are running YP, simply use the default configuration file. Messages will be logged to that single system in your YP domain with the name 'loghost'.

To see which machine in your domain is the loghost execute the command shown below.

```
% ypmatch loghost hosts
129.141.5.4      central mailhost loghost
```

If you are not running YP, edit your `/etc/hosts` file to assign the alternate name `loghost` to the system to which you wish your messages to be logged. The name `loghost` should be defined only once in your `/etc/hosts` file.

Solution 2

If you wish messages to be logged to a system other than your own that is not defined as the loghost (for this example, the system `altlog`), edit your `/etc/syslog.conf` file and replace each occurrence of `@loghost` with `@altlog` .

Solution 3

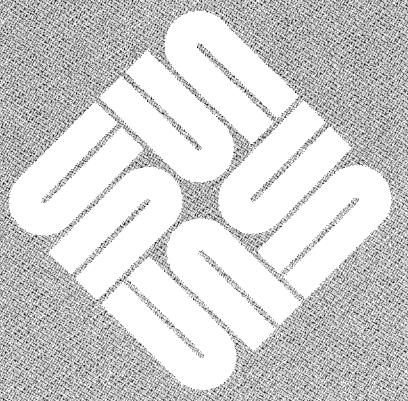
Finally, if you wish to log messages on your local system insert the line shown below as the first line your `/etc/syslog.conf` file.

```
define(LOGHOST, 1)
```

Please note, however, that if the workstation's hostname is 'loghost', you should use the default configuration file. In this case, redefining `LOGHOST` causes `syslogd` initialization to fail.

3

STB SHORT SUBJECTS



3

STB SHORT SUBJECTS

Cross-Compilers 3.0

Cross-Compilers Release 3.0 Announcement

This short subject announces the availability of Cross-Compilers release 3.0 for Sun-3 and Sun-4 systems (including the Sun3x and Sun4c kernel architectures) running Sun Operating System (SunOS) release 4.0 or greater. Target systems include all Sun-3 systems, Sun-4 systems, and the Sun-386*i* system.

Cross-Compilers Packaging by Language

Cross-Compilers release 3.0 is packaged by language: C, Pascal, and FORTRAN and are sold as separate products. Users will need access to multiple Cross-Compilers if a program uses inter-language program calls. For example, if a mixed FORTRAN and C program is to be cross-compiled, then FORTRAN program will need to be cross-compiled with the FORTRAN Cross Compiler, and the C program will need to be cross-compiled with the C Cross Compiler. An exception is when inter-language program calls are to SunOS library routines. These libraries are included with each Cross Compiler, and do not require multiple Cross-Compilers.

SunOS 4.0.3 machdep.c

SunOS Release 4.0.3
sun3/machdep.c

There is a typographical error in the code which affects the SunOS release 4.0.3 Sun-3 version of /usr/src/sys/sun3/machdep.c.

The Problem

A typo in the SunOS release 4.0.3 Sun-3 version of /usr/src/sys/sun3/machdep.c causes any Sun-3/260 or Sun-3/280 to crash whenever it gets an ECC memory error, even if the error is a correctable one. This problem does not affect the Sun-3x machines because the code is correct and does not contain the typo. The problem does not exist on a Sun-4 with ECC memory. It does affects Sun-3/260s and Sun-3/280s running SunOS release 4.0.3 (not 4.0 or 4.0.1).

The typo on the Sun-3 version is that EER_UE is accidentally EER_CE:

```
if (cpu == CPU_SUN3_260 && (((eer & EER_ERR) == EER_CE) &&
        ((eer & EER_ERR) != EER_CE))) {
```

The Fix

The fix for the current running kernel and future SunOS release 4.0.3 kernels is as follows:

```
# adb -w -k /vmunix /dev/mem
memerr+0x3a?w 7202
memerr+0x3a:    7201    = 7202
^D

# chmod u+w /usr/share/sys/sun3/OBJ/machdep.o
# adb -w /usr/share/sys/sun3/OBJ/machdep.o
memerr+0x3a?w 7202
memerr+0x3a:    7201    = 7202
^D
# chmod u-w /usr/share/sys/sun3/OBJ/machdep.o
```

Printing *plot(1g)* Files

Printing *plot(1g)* Files

This short subject lists printing procedures for *plot(1g)* files.

1. Assuming that a filter (`gf:`) is specified in the `printcap` file, type the command below:

```
% spline < spline.dat | graph | lpr -g
```

Note: Do not run it through `plot`.

2. If you do not have an effective filter in the `printcap` file, type the command below:

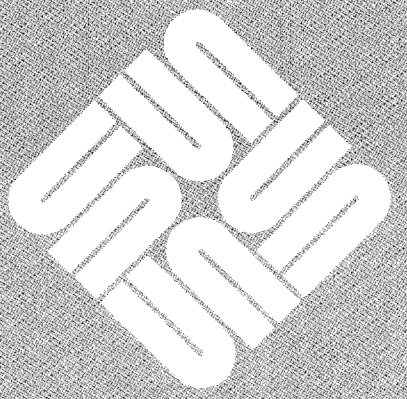
```
% spline < spline.dat | graph | plot -Ttek | ps4014 | lpr
```

3. Use `psplot` from the TranScript distribution by typing the command below:

```
% spline < spline.dat | graph | psplot | lpr
```

4

IN DEPTH



4

IN DEPTH

SunCore to SunPHIGS

SunCore to SunPHIGS Translation Guide

This month's STB In Depth feature contains the *SunCore to SunPHIGS Translation Guide*. The guide consists of the preface, four chapters, and two appendices.

Preface

This guide describes the various methods in which SunCore application programs may be upgraded to the SunPHIGS standard. SunCore is an implementation of the *ACM SIGGRAPH Core System* graphics library by Sun Microsystems, Inc. SunPHIGS is an implementation of PHIGS, the *ANSI/ISO Programmer's Hierarchical Interactive Graphics System.*

It is assumed that the SunPHIGS software has been installed as described in the *SunPHIGS Installation Guide.*

Chapter 2 describes public-domain software available from the Sun Microsystems User Group, Inc., an independent non-profit organization.

Audience

The intended reader of this guide is an applications programmer who is familiar with interactive computer graphics, the C programming language, and the SunCore graphics library. This guide acts as a launching pad into the SunPHIGS documentation set (*Getting Started with SunPHIGS* and the *SunPHIGS Reference Manual*).

Documentation Conventions

Italic font is used to indicate variables that are to be replaced by a specific values. Function names, file names, commands, and options are printed with `listing font`. Finally, example commands to be entered exactly as shown are printed in **`bold listing font`**.

Chapter 1

Chapter 1: An Overview of Translation

Chapter 1 includes an introduction, reasons for upgrading to SunPHIGS, methods of translation, and references for further reading.

Introduction

If you are an applications programmer planning on upgrading your SunCore programs to the powerful SunPHIGS standard, this guide is for you. Included is advice on, and justification for, various methods of translation between the two graphics libraries.

As a co-requisite, please refer to the SunPHIGS documentation set (*Getting Started with SunPHIGS* and the *SunPHIGS Reference Manual*).

Reasons for Upgrading to SunPHIGS

The ACM Core System was never ratified since its birth in 1977, as more powerful graphics libraries were continually introduced. Since then, PHIGS has rapidly gained industry acceptance as the standard, object-oriented, interactive, 3-D graphics library. Reasons for upgrading existing SunCore applications to the SunPHIGS standard include:

- With the release of SunOS 5.0, SunCore libraries will be moved to the `/usr/old` directory.
- SunPHIGS runs across the Sun product line.
- SunPHIGS supports SunTool, SunCanvas, and CGM (Computer Graphics Metafile) type workstations under the SunView and X11/NeWS-merge windowing environments.
- SunPHIGS's state-of-the-art hierarchical structuring benefits applications and application programmers. Many operations previously handled in SunCore application programs are now automated internally by SunPHIGS. Whereas the SunCore programmer describes pictures using simple drawing primitives, the SunPHIGS programmer describes pictures by the logical relation of objects. Thus SunPHIGS greatly reduces both the development time of new applications and the maintenance time of existing applications.
- SunPHIGS conforms to an industry-wide standard. The PHIGS standard was ratified by the International Organization for Standardization (with ANSI ratification soon to follow). PHIGS allows applications to be easily ported between machines and manufacturers, since it is both device and processor independent.

Reasons Not to Port to SunPHIGS

The following are reasons not to port to SunPHIGS.

- The PHIGS standard has no provision for lighting or shading models. If your application requires lighting or shading, you must either wait for the release of SunPHIGS+, or simulate lighting effects in your application program.
- SunPHIGS Release 1.0 does not yet fully support the `cellarray` primitive (PHIGS' equivalent of SunCore's `raster` output primitive).
- SunPHIGS Release 1.0 does not support `annotation` text, but will be provided in a future release.

Methods of Translation

Once you have decided to upgrade a SunCore application to SunPHIGS, the method of translation must be chosen. Table 1-1 lists the three methods discussed in this guide.

Method	Advantages	Disadvantages
1. Start from scratch and rewrite entire program. *Best for the long-term.*	Allows for easy maintenance of code due to hierarchical structure of SunPHIGS. Graphical code may be separated from functional code so that future standards may be accommodated easily. New program and graphical features may easily be added as the program matures. SunPHIGS conforms to industry standard. Probable increase in program and drawing speed. Allows program to run across the Sun product line. No dependence on SunCore.	May be time consuming to rewrite the program. Programmer must have an intimate knowledge of the program's function, and SunPHIGS. Some SunCore functions are not available in SunPHIGS, requiring the programmer to simulate them.
2. Replace SunCore functions inline with SunPHIGS. *Knee-patch method*	Possible increase in drawing speed. No dependence on SunCore. SunPHIGS conforms to industry standard. Allows program to run across the Sun product line. Does not require the programmer to have an intimate knowledge of the program's function. Takes less time to complete than method (1) above.	May be time consuming to recode sections of a program. Possible decrease in drawing speed. Does not allow for graphical code to be separated from functional code. Does not take advantage of SunPHIGS' hierarchical structure. Does not allow for easy implementation of new program features as program matures. Some SunCore functions are not available in SunPHIGS, requiring the programmer to simulate them.
3. Relink program to SunCore 'wrapper' library. *Quick short-term method*	Fastest translation method. Usually does not require any programming. Possible increase in drawing speed.	Original program contains SunCore calls. Wrapper functions may not emulate SunCore exactly. Some functions in SunCore are not supported in SunPHIGS. Probable decrease in drawing speed. Does not allow graphical code to be separated from functional code. Does not take advantage of SunPHIGS' hierarchical structure. Does not allow for easy implementation of new program features as program matures. Does not allow program to run across Sun's entire product line.

Table 1-1: Translation Paths from SunCore to SunPHIGS

Recoding from Scratch

As table 1-1 indicates, taking the time to rewrite a SunCore program is justified in the following situations:

- An application program whose expected end-of-life is far off in the future
- A continually modified or maintained application program
- A graphics-intensive application program in which drawing speed is important

By recoding a SunCore application, all the features of SunPHIGS are available. Chapter 3 'Conceptual Differences between SunCore and SunPHIGS' and chapter 4 'Categorical Comparison between SunCore and SunPHIGS' provide hints and advice on this method of translation.

In-Line Substitution

When time is a factor, a SunCore program may be translated to SunPHIGS by replacing all SunCore function calls with equivalent sets of SunPHIGS routines. Since the majority of the application's code remains unchanged, the inline substitution method works best with:

- A rarely modified or maintained application program
- An application program nearing its end-of-life
- An application program with limited use of graphics

Without restructuring an application program however, you surrender many of the advanced features of SunPHIGS. Chapter 4 'Categorical Comparison between SunCore and SunPHIGS' provides hints and advice on this method of translation.

Relinking to Wrapper Functions

The least effective, yet easiest method of translation is to relink a SunCore application program to a 'wrapper' library (as opposed to the `-lcore` library). Each function of a 'wrapper' library emulates the action of a specific SunCore function (using SunPHIGS routines). Relinking a SunCore program to a 'wrapper' library is advised for:

- An application program that will eventually be upgraded to SunPHIGS by recoding from scratch, or inline substitution
- An application program using only 'simple' SunCore functions, since 'wrapper' routines do not emulate SunCore exactly
- An application in which drawing speed is unimportant

Due to the additional calculations required for emulation, SunCore programs relinked to a 'wrapper' library generaly run slower than both programs using only SunCore or SunPHIGS. Many applications will fail with this method since

'wrappers' do not emulate SunCore exactly. Chapter 2 'Using Wrapper Functions' describes a 'C' wrapper library (FORTRAN wrappers are not available) available from the Sun Microsystems User Group.

Further Reading

In addition to the co-requisite SunPHIGS documentation (*Getting Started with SunPHIGS*, part number 800-3061-10; and the *SunPHIGS Reference Manual*, part number 800-2475-10), SunCore programmers unfamiliar with SunPHIGS may benefit from the following:

- *Understanding PHIGS*
  TEMPLATE Software Division of Magatek Corporation, 1985

- *A Brief Introduction to PHIGS*
  Bunshaft, Albert J. Computer Graphics '85 Conference Proceedings, Volume II, NCGA (National Computer Graphics Association), Fairfax, Virginia, 1985, pp. 326-331

- *Programmer's Hierarchical Interactive Graphics System (PHIGS)*
  ISO (the International Organization for Standardization), ISO/DIS 9592-1:1987(E)

- *SunView Programmer's Guide*, part number 800-1783
  Sun Microsystems

- *SunView System Programmer's Guide*, part number 800-1784
  Sun Microsystems

- *Pixrect Reference Manual*, part number 800-1785
  Sun Microsystems

Chapter 2

Chapter 2: Using Wrapper Functions

Without programming, you can port your SunCore application program to SunPHIGS by simply relinking to a 'wrapper' library. Unlike the `-lcore` library, the wrapper library reroutes all graphics output to SunPHIGS. Since the original SunCore application remains unchanged, this porting method is *not* recommended for the following cases:

- Application programs which must run across the Sun product line.
- Application programs which must conform to an industry-wide standard. PHIGS applications are easily ported between machines and manufacturers, since it is both device and processor independent.
- Continually modified or maintained application programs. Wrapper libraries do not allow use of SunPHIGS' hierarchical structuring (which reduces both development and maintenance time).

Where to Get a Wrapper Library

The Sun Microsystems User Group, Inc., an independent non-profit organization, distributes C source code for a SunCore to SunPHIGS wrapper library. Tapes of this and other public-domain software are available from your local chapter of the Sun User Group, or write to:

Sun Microsystems User Group, Inc.
2550 Garcia Avenue M/S 10-16
Mountain View, CA 94043
(415) 336–4343
TLX: 469327

The source code of each wrapper function provides an example of how to emulate a SunCore function using SunPHIGS routines. Thus, even if you plan on rewriting your SunCore application to use SunPHIGS, refer to the Sun User Group wrapper library for examples.

Linking to Wrappers

Using the wrapper library to port your SunCore C application to SunPHIGS simply requires a compilation and link. No wrapper library is provided for FORTRAN or Pascal application programs. First compile the wrapper library by typing:

```
tutorial% cc -c wrappers.c
```

Next, search through your SunCore application program's `Makefile` for any references to `-lcore` and replace them with `wrappers.o -lphigs`. For example, to compile the `Core_glass.c` program in Appendix B under SunCore, you would type:

```
tutorial% cc Core_glass.c -o glass -lcore -lsunwindow -lpixrect -lm
```

To compile this same program under the wrapper environment, however, you would type:

```
tutorial% cc -c wrappers.c -o wrappers.o
tutorial% cc Core_glass.c -o glass wrappers.o -lphigs -lsuntool -lsunwindow -lpixrect -lm
```

Problems with Wrappers

Some applications may fail after linking to the wrapper library since the wrapper functions do not emulate SunCore functions exactly. In each case, these problems may be avoided by rewriting your SunCore application to use SunPHIGS directly. Refer to chapters 3 and 4, and to *Getting Started with SunPHIGS* for information on rewriting your application. A list of problems and inaccuracies with the wrapper library follows, with some possible solutions:

- *Execution Speed*

  Due to the additional calculations required for emulation, SunCore programs relinked to the wrapper library generaly run slower than programs using only SunCore or SunPHIGS.

  Solution:

  Minimize SunPHIGS' screen-refresh with liberal use of the `begin_batch_of_updates` function.

- *Multiple View Surfaces*

  In order to reduce wrapper complexity, the wrapper library does not emulate multiple SunCore view surfaces. SunPHIGS however, allows multiple display surfaces called *workstations* (discussed in chapters 3 and 4).

  Solution:

  Manually insert SunPHIGS `ppoststruct` function calls in your application program for each segment that should be posted to additional workstations (view surfaces). Additional SunPHIGS workstations are added by inserting `popenws` function calls in your application program.

- *Lighting and Shading*

  Unlike SunCore, SunPHIGS release 1.0 does not support Gouraud or Phong shading (shading is not included in the ISO PHIGS standard).[4] The wrappers convert Gouraud shading to constant shading using the first vertex set by `set_vertex_indices`. The wrappers do not support Phong shading.

  Solution:

  Implement a shading routine in your application program by splitting polygons into a mesh of individually colored miniature polygons. Shading algorithms are discussed in *Fundamentals of Interactive Computer Graphics*, Foley, J.D. and Van Dam, A., Addison-Wesley, Reading, Massachusetts, 1982.

- *Image Transformations*

  Unlike SunPHIGS, SunCore allows a final image transformation after output primitives have been converted to NDC space (Normalized Device Coordinates). Therefore, the wrappers ignore all SunCore image transformation requests. The unshaded areas of figure 2-1 show SunPHIGS' composite modeling transformation in place of SunCore's image transformation.

  Solution:

  Use either SunPHIGS' Modeling transformation, or view orientation transformation to simulate SunCore's image transformation.

[4] SunPHIGS+ will provide additional features to the PHIGS standard, including lighting and shading models.

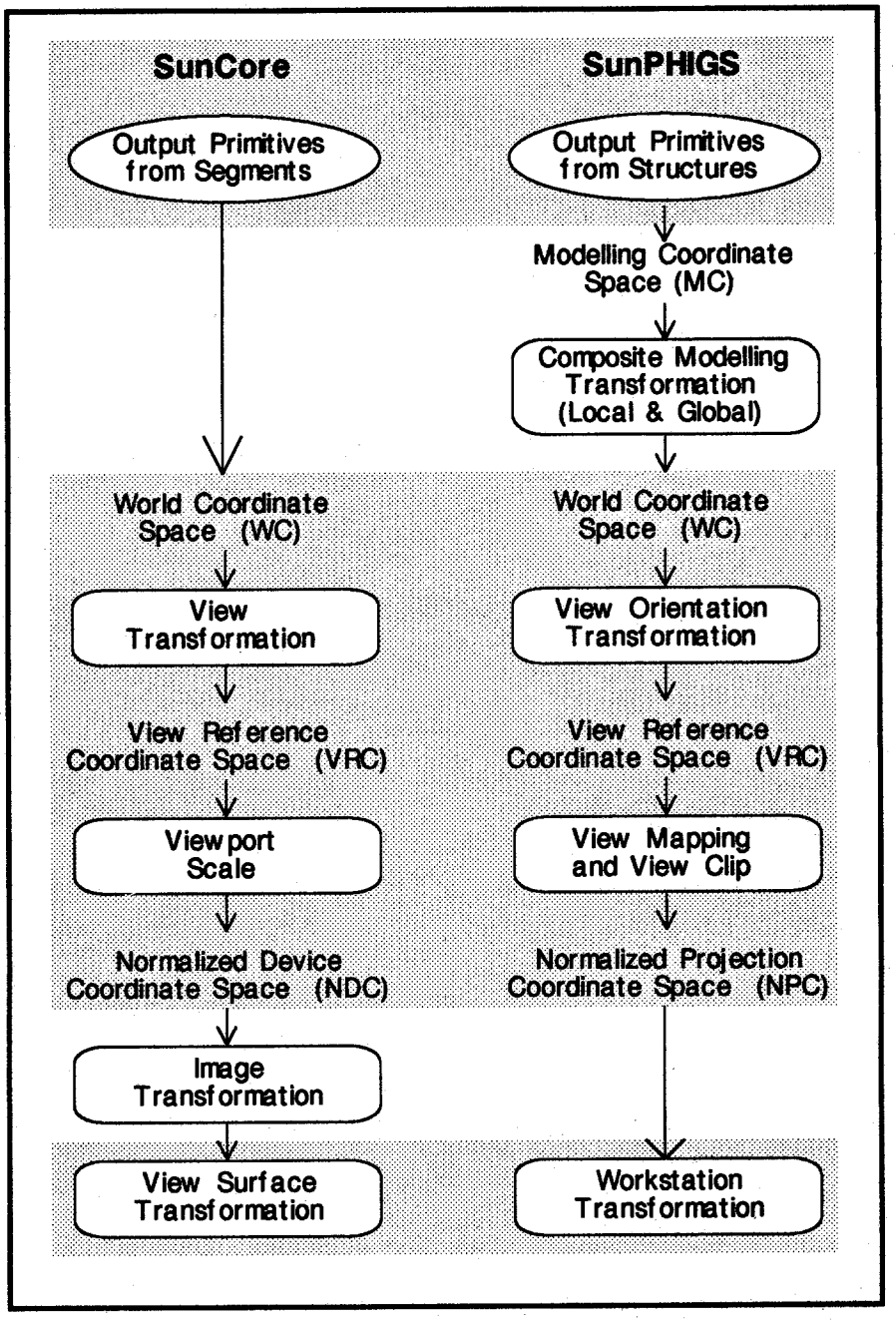


Figure 2-1: The Transformation Pipelines

- *Temporary Segments*

  As opposed to SunCore, SunPHIGS release 1.0 does not support temporary segments (non-retained data is not a part of the PHIGS standard). Since SunCore does not allow segments to have negative identifiers, the wrappers store temporary segments in negatively numbered SunPHIGS structures. The `new_frame` wrapper function

(which emulates the SunCore `new_frame` function) subsequently deletes all structures with negative identifiers.

- *Markers*

  Although SunCore allows the marker symbol to be any printable character, SunPHIGS markers may only be one of five symbols. Thus the wrappers translate SunCore markers into SunPHIGS text. Since SunPHIGS text is transformed by the current viewing matrix (unlike SunCore markers), markers produced by the wrappers may be unacceptable.

  Solution

  If possible, use only the five SunPHIGS symbols with direct calls to `psetmarkertype` and `ppolymarker` in your application program. A future release of SunPHIGS will support annotation text.

Unimplemented Functions

Due to the difference in methodology between SunCore and SunPHIGS, the wrapper library ignores the following functions:

allocate_raster	select_view_surface
deselect_view_surface	set_charpath_3
file_to_raster	set_charup_3
free_raster	set_coordinate_system_type
get_raster	set_drag
get_view_surface	set_image_transformation_2
inquire_image_transformation_2	set_image_transformation_3
inquire_image_transformation_3	set_image_transformation_type
inquire_image_transformation_type	set_image_translate_2
inquire_image_translate_2	set_image_translate_3
inquire_image_translate_3	set_light_direction
inquire_rasterop	set_output_clipping
inquire_retained_segment_surfaces	set_rasterop
inquire_segment_image_transformation_2	set_segment_image_transformation_2
inquire_segment_image_transformation_3	set_segment_image_transformation_3
inquire_segment_image_translate_2	set_segment_image_translate_2
inquire_segment_image_translate_3	set_segment_image_translate_3
map_ndc_to_world_2	set_shading_parameters
map_ndc_to_world_3	set_vertex_indices
print_error	set_vertex_normals
put_raster	set_zbuffer_cut
raster_to_file	size_raster
report_most_recent_error	

Table 2-1: Unimplemented Wrapper Functions

Each of the above functions may be performed by customizing the wrapper library or your application program with the SunPHIGS routines listed in Appendix A, 'SunCore C Functions with Related SunPHIGS Functions'.

Internal Operation

In order to transparently convert between the drawing philosophies of SunCore and SunPHIGS, the wrapper library performs several internal functions. When customizing the wrapper library, you must understand its use of an attribute segment, and viewing bundles.

Attribute Table

When the linewidth attribute is set by a SunCore application program, an entry is made in the SunCore Current Attribute Table. If the application program later draws a line, SunCore copies this linewidth attribute from the Current Attribute Table into the currently open segment. Thus SunCore attributes may be set whether or not a segment is currently open. Furthermore, after an attribute is set all subsequently opened segments will inherit that value (until the attribute is changed).

In SunPHIGS however, attributes (or attribute-bundle indices) are placed directly into a currently open structure. Unless hierarchically arranged, an attribute set in one structure will not propagate to the next opened structure.

Since attributes may be received when no structure is open, the wrapper library emulates SunCore by storing all current attributes in a Wrapper Attribute Structure. See table 2-2 below. Whenever a SunPHIGS structure is created, the wrapper library copies this Attribute Structure into the newly created structure. Furthermore, if a structure is open when an attribute is set, the wrappers not only update the Attribute Structure but also the opened structure.

character_height	line_style
character_expansion	line_width
character_spacing	marker_colour_index
global_matrix_3	text_colour_index
interior_colour_index	text_precision
line_colour_index	view_index

Table 2-2: SunPHIGS values stored in the Wrapper Attribute Structure

View Indices

Whereas SunCore's viewing parameters may be individually set like attributes, SunPHIGS' viewing parameters may only be set as a bundle. For example, in SunCore the reference point may be set by calling `set_view_reference_point` prior to opening a segment. To set the reference point in SunPHIGS however, the viewing parameters in table 2-3 are bundled into a view representation using the `psetviewrep3` function. The specific view-representation bundle a structure uses is set by the `psetviewind` function.

Following a call to any function listed in table 2-3, the wrapper library will create a new view-representation bundle at the next structure creation. However, by default SunPHIGS allows only 20 view bundles. When this limit is reached, the wrapper library redefines the oldest view bundle *even if it is being used.* When a structure is created, it inherits the current view bundle from the Wrapper Attribute Structure.

set_back_plane_clipping	set_view_reference_point
set_front_plane_clipping	set_view_up_2
set_ndc_space_2	set_view_up_3
set_ndc_space_3	set_viewport_2
set_projection	set_viewport_3
set_view_depth	set_viewing_parameters
set_view_plane_distance	set_window
set_view_plane_normal	set_window_clipping

Table 2-3: Functions which Create a New View Index

Chapter 3

Chapter 3: Conceptual Differences between SunCore and SunPHIGS

Before porting your application program to SunPHIGS, it is necessary to understand some conceptual differences between the graphics libraries. This chapter discusses the conceptual differences between SunCore programming and SunPHIGS programming. Command-level differences between the graphics libraries are discussed in chapter 4, 'Categorical Comparison between SunCore and SunPHIGS'. For a tutorial of SunPHIGS, refer to *Getting Started with SunPHIGS* in the SunPHIGS documentation set.

Modeling versus Viewing

Graphics libraries render pictures either by *viewing* a scene, or by *modeling* a system. Whereas SunCore is a viewing system, SunPHIGS is a modeling system.

As the name implies, a viewing system positions an imaginary camera to view output primitives. Since output primitives are the responsibility of an application program, Core does little more than transform a pre-defined scene to particular camera angle. Therefore, a large fraction of a SunCore programmer's time deals with the definition of complex scenes using only simple output primitives.

A modeling system however, *assembles* a scene from 'objects'. PHIGS applications form objects by grouping output primitives or simpler objects. The *relation* of objects, output primitives, and attributes are all stored in SunPHIGS's hierarchical graphical-database. Thus, to draw an electric circuit, you would enter basic components (resistors, capacitors, etc.) into the database, and then tell SunPHIGS how to assemble them. Since 'objects' in the database may be copied, edited, and joined in a hierarchical structure, SunPHIGS streamlines application programs.

Structures versus Segments

Unlike Core's *segments*, PHIGS stores output primitives in *structures*. Advantages of structures over segments include the ability to interactively edit existing structures and the ability to form 'objects' by linking structures together.

A major setback of SunCore is the inability to edit existing segments. This problem is compounded by SunCore's method of storing output primitives in segments *after* a viewing transformation. Figure 3-1, 'The SunCore Transformation Pipeline' displays the formation of SunCore segments. Since segments store normalized positions of output primitives, it is impossible for SunCore to edit primitives without knowing their original positions. As a result, not only must a segment be deleted and redrawn to change the linewidth of an object, but also to view the object from a new camera position. Orbiting the SunCore camera about a static object forces the application program to continually delete and redefine every segment.

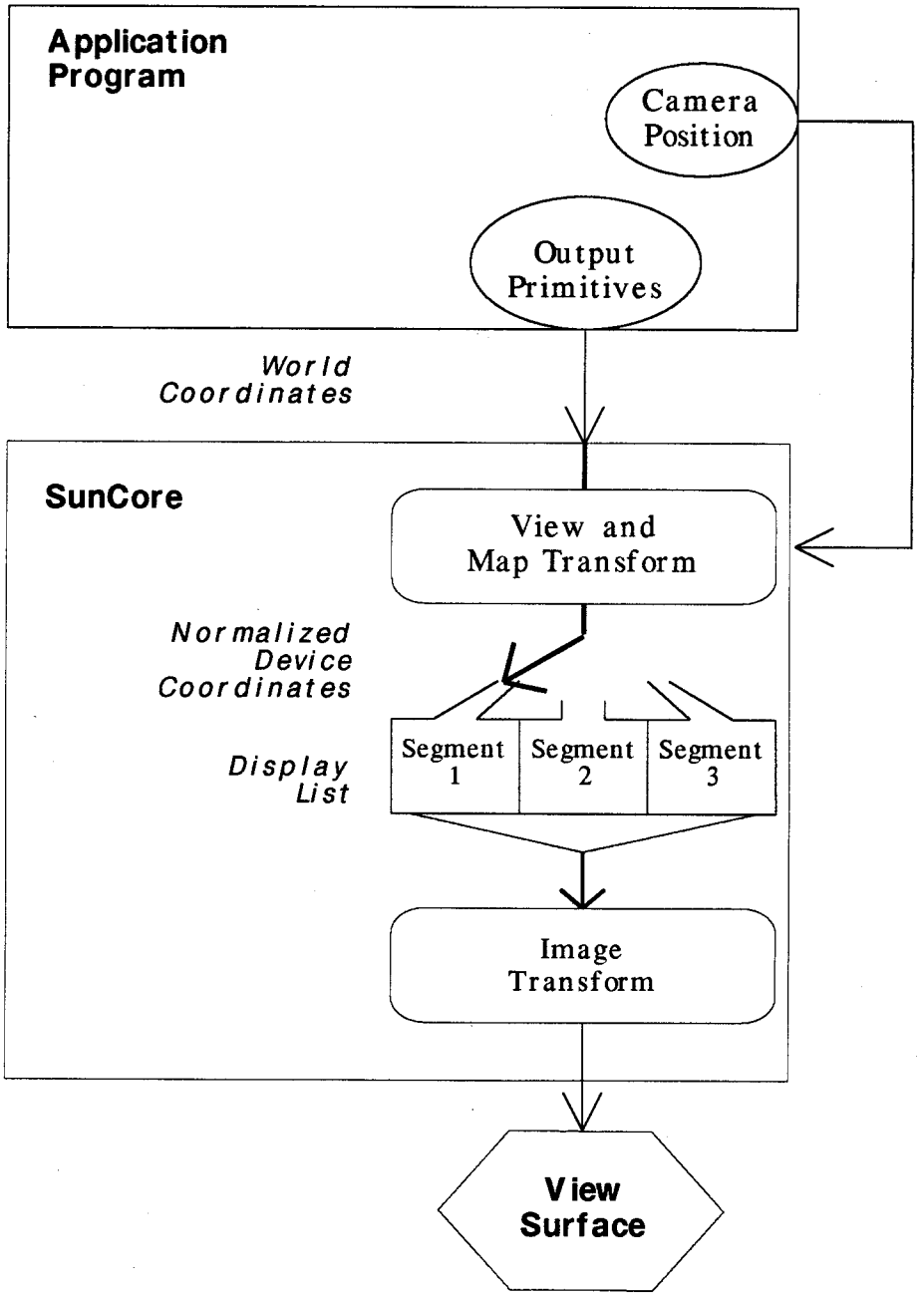


Figure 3-1: The SunCore Transformation Pipeline

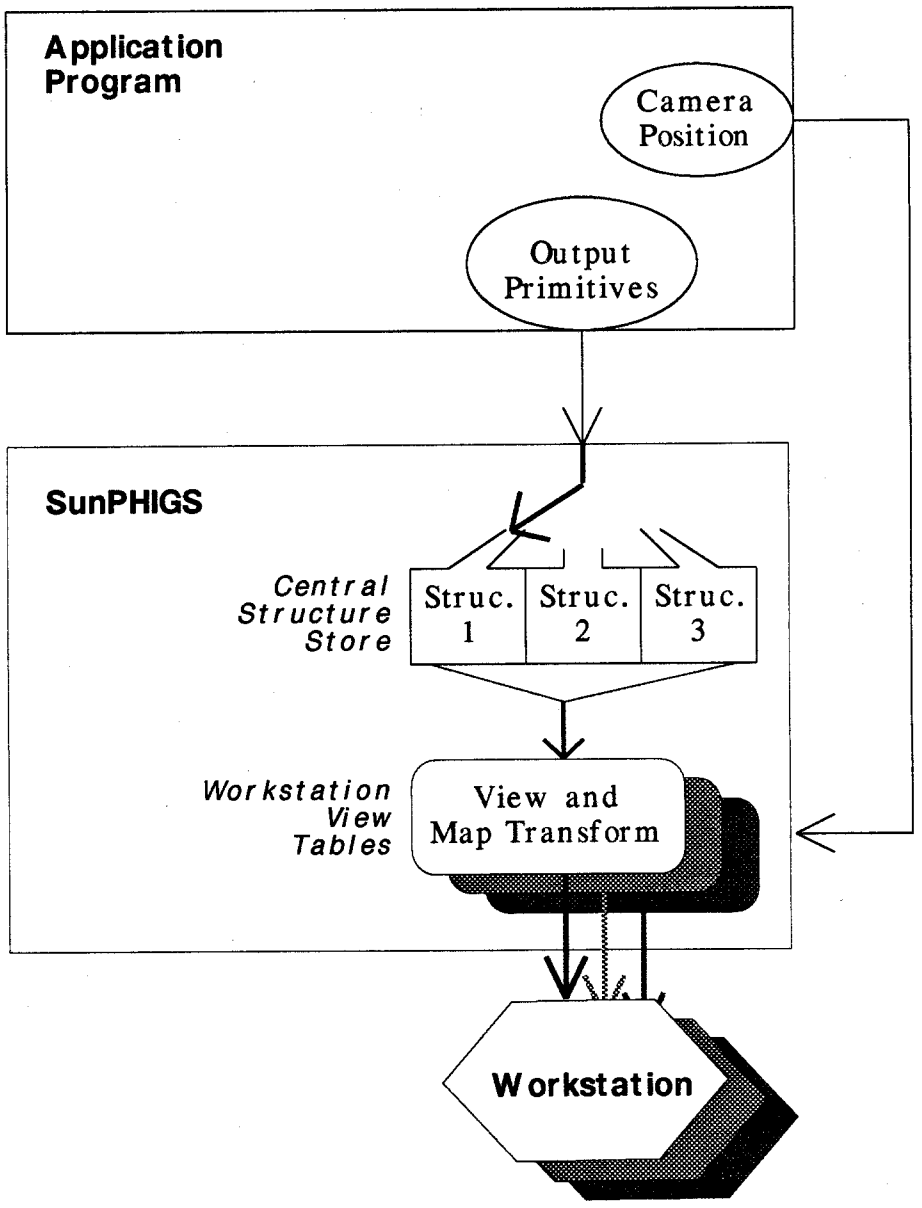


Figure 3-2: The SunPHIGS Transformation Pipeline

In PHIGS however, viewing transformations occur *after* the *central structure store* as shown in figure 3-2, 'The SunPHIGS Transformation Pipeline'. In order to orbit the SunPHIGS camera about a static object, the application program need only supply new viewing parameters. SunPHIGS *automatically updates* the display using output primitives in the central structure store. In addition, SunPHIGS application programs may interactively edit output primitives and attributes in the central structure store.

More important than its editing capabilities, PHIGS offers applications hierarchical structuring. Only when a structure is posted (using the POST STRUCTURE function) will the output primitives in the central structure store be made available for viewing.

Traversal, the execution of structures, involves sequentially passing the output primitives of posted structures through the transformation pipeline. Structures can be invoked from other structures (by placing the EXECUTE STRUCTURE function in a posted structure). A referenced structure, or one which is executed from another, is referred to as a *child* structure. Structures which execute child structures are referred to as *parent* structures.

A structure *inherits* the attributes of its parent, much as a subroutine would inherit the values passed in its parameter list. A child structure may modify its own attributes, without affecting the parent structure's attributes. Thus, the same structure invoked from different parents might inherit different attribute values. Posted structures of a structure network inherit attributes from the SunPHIGS Description Table which contains standard system default values.

For further explanation of SunPHIGS structures, refer to chapter 2 in *Getting Started with SunPHIGS.*

Workstations versus View Surfaces

A third conceptual difference between SunCore and SunPHIGS involves drawing placement. Whereas SunCore directs output to a *view surface*, SunPHIGS directs pictures to one or more *workstations*. Since SunCore uses the same viewing parameters for all active view surfaces, graphical output is rarely sent to more than one view surface. To provide three different views of the same object, a SunCore application must draw the object once for each of the three view surfaces.

SunPHIGS however, allows any structure in the central structure store to be *posted* to any workstation. Figure 3-3 shows a hierarchy providing three views of an object by only drawing the object once. Whereas SunCore takes control of an application program's UNIX window, SunPHIGS creates a new SunView window for each workstation opened.

Programming Tips

With the above concepts will you realize the full power of SunPHIGS when re-writing your SunCore applications. The following programming tips result from these concepts:

- Extensively use hierarchical structures (as opposed to SunCore's linear segments) to eliminate redundancy and to define 'object' relations. For example, when drawing an automobile only one unposted structure should contain the definition of a 'tire'. A *posted* structure would then translate and execute the child 'tire' structure four times. In addition, use hierarchical structures to define 'object' relations so that as the automobile moves, the tires move with it. Figure 3-3 shows a relational structure such that if a robot's arm is rotated, so are its wrist and hand.

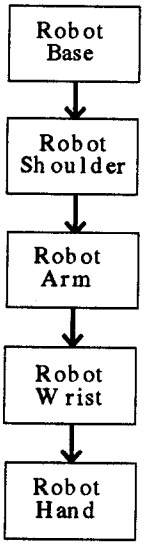


Figure 3-3: SunPHIGS Relational Structure Example

- Separate graphical code from functional code in SunPHIGS applications. For example, an initialization module should build the SunPHIGS database with all the 'objects' to be used by the application (as opposed to SunCore's build as needed method). Both you and the application benefit from the ability to perform complex operations by simple editing of the SunPHIGS database. In addition, the centralization of graphics code in an initialization module simplifies application maintenance.

- When drawing three-dimensional scenes, use multiple workstations to provide simultaneous views of the scene from different camera positions. Figure 3-4 shows such a structure.

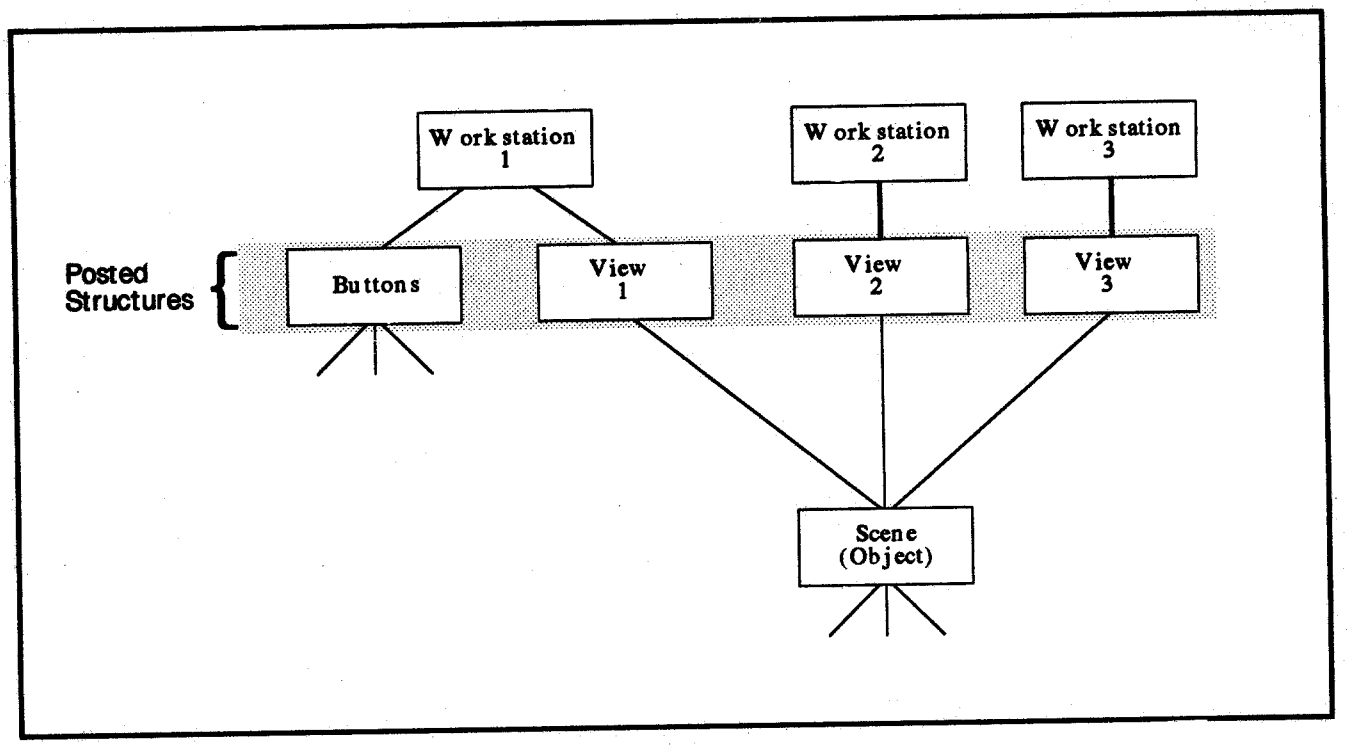


Figure 3-4: Using Multiple Workstations

Chapter 4

Chapter 4: Categorical Comparison between SunCore and SunPHIGS Equivalents

This document compliments and does not replace the SunPHIGS documentation set. Therefore, a list of differences for each of the 205 SunCore functions is beyond the scope of this In Depth feature. Instead, this chapter details an important group of function-level differences between the graphics libraries.

For situations not covered by this document or *Getting Started with SunPHIGS*, two additional references are provided. First, the source code of the wrapper library, discussed in Chapter 2, provides hundreds of programming examples showing SunCore functions emulated by SunPHIGS. Second, Appendix A of this feature lists equivalent SunPHIGS routines for each of the SunCore functions, and may be used as an 'index' into the *SunPHIGS Reference Manual*.

Viewing Operations

The most significant difference between SunCore and SunPHIGS involves viewing transformations. Specifically, the two graphics libraries utilize different coordinate systems to define viewing parameters. Table 4-1 shows the default viewing parameter values and coordinate spaces for the two graphics libraries. Refer to Appendix A and the *SunPHIGS Reference Manual* for viewing function names and parameter formats, respectively.

Parameter	*SunCore* Default		*SunPHIGS* Default	
View Reference Point	{0,0,0}	WC	**{0,0,0}**	**WC**
View Plane Normal	{0,0,−1}	RWC	**{0,0,−1}**	**RWC**
View Plane distance	0	MVRC	**0**	**VRC**
Front Plane distance	0	MVRC	**1**	**VRC**
Back Plane distance	1	MVRC	**0**	**VRC**
Type of Projection	Parallel		**Parallel**	
Projection Reference Pt.	{0,0,1}	RWC	**{0.5,0.5,1.0}**	**VRC**
Window	{0,1,0,0.75}	MVRC	**{0,1,0,1}**	**VRC**
View Up Vector	{0,1,0}	RWC	**{0,1,0}**	**RWC**
Device Space	{0,1,0,0.75,0,1}	NDC	**{0,1,0,1,0,1}**	**NPC**
Viewport	{0,1,0,0.75,0,1}	NDC	**{0,1,0,1,0,1}**	**NPC**

Table 4-1: Viewing Operation Parameters

Coordinate Systems

The following paragraphs summarize the coordinate systems used to construct graphical objects.

- *MC*

  Modeling Coordinate spaces are used to construct graphical objects from output primitives. The relative positions of the modeling coordinate systems are mapped into a single world coordinate space by modeling transformations.

  In SunCore the modeling transformation is set with the `set_world_co-ordinate_matrix_2` and `set_world_coordinate_matrix_3` functions. In SunPHIGS, the modeling transformation is set with the `psetlocaltran`, `psetlocaltran3`, `psetglobaltran`, and `psetglobaltran3` functions.

- *WC*

  World Coordinate space is a device-independent, *virtual* viewing space which defines the relative positions of objects built in modeling coordinates.

- *RWC*

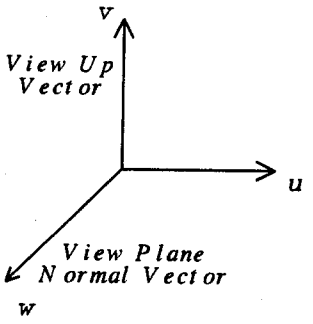
  Relative World Coordinates define positions in world coordinates relative to the *view reference point*. SunCore uses relative world coordinates to define vector directions, and plane distances from the view reference point. For example, if the view reference point is set to `( 2, -3, 1) WC`, then the point `( 4, 3, 2) RWC` is equivalent to the point `( 6, 0, 3) WC`.

- *VRC*

  View Reference Coordinates are used by SunPHIGS to define a virtual camera's viewing position of the world coordinate space. It is with this camera's viewpoint that objects are transformed to device coordinate space for drawing. In SunPHIGS it is customary for the view plane normal vector to point towards the virtual camera, otherwise a mirror image of world space results.

  The view reference coordinate system (also called the *uvw* coordinate system) is a right-handed Cartesian system with three orthogonal axes: *u*, *v*, and *w*. The *v* axis is defined by the view up vector. The *w* axis points in the direction of the view plane normal vector. Since the *uvw* system is right-handed, the *u* axis is directed 90 degrees clockwise from the *v* axis, as viewed with the *w* axis pointing towards you. The view reference point defines the origin of this *uvw* system.

  For example, a *positive* view plane distance parameter positions the clipping plane in front of the view reference point.

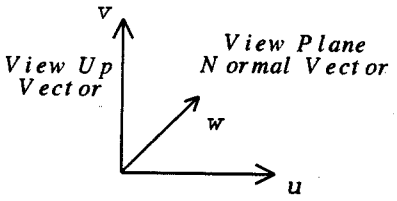


PHIGS' VRC

- *MVRC*

  Modified View Reference Coordinate space is SunCore's left-handed version of the view reference coordinate space. With SunCore's left-handed system, it is customary for the view plane normal vector (and thus the *w* axis) to point *away* from the virtual camera.

  Unlike with VRC coordinates, a *negative* view plane distance parameter positions the clipping plane in front of the view reference point.



Core's MVRC

- *NDC*

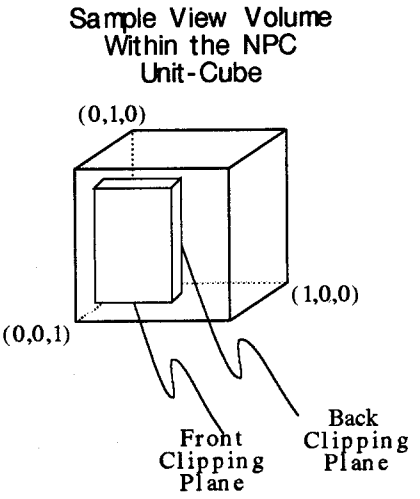
  SunCore's Normalized Device Coordinate space is a fixed-coordinate system which is independent of physical output devices. NDC space is a cube with each side of unit length. Primitives in world coordinates are transformed to NDC space for clipping and viewing by the view orientation and view mapping transformations. Each physical output device driver then transforms from NDC space to the physical device coordinates for each view surface.

- *NPC*

  Normalized Projection Coordinate space is equivalent to NDC space but for SunPHIGS workstations. All output primitives outside of the unit-cube NPC volume are clipped, even if the parameters `xy_clip`, `front_clip`, and `back_clip` are set to `PNOCLIP`. Imagine the front clipping plane, the back clipping plane, and the projection vectors

as defining a volume of world coordinate space which is compressed to fit somewhere within the NPC unit-cube. Clipping will occur either at the boundaries of the compressed volume, or at the boundaries of the NPC cube. Thus setting the `back_clip` parameter to `PNOCLIP` does not allow an infinitely far view in the direction of the view plane normal vector, since clipping will occur at the back of the NPC unit-cube. Setting SunPHIGS' back clipping plane at a large negative distance has the same effect as turning SunCore's back clipping off.

For proper *z*-buffer operation, SunPHIGS' front and back clipping planes should be chosen to enclose the smallest volume of world coordinate space that encompasses the objects being viewed.



Left-Handed World Space

Unlike SunCore, SunPHIGS only supports a right-handed coordinate system. SunCore's optional left-handed system may be emulated in SunPHIGS by multiplying all *z* coordinates by negative one (-1).

Projection Reference Point

The projection reference point orients the projectors that define the sides of the view volume. If the *projection type* is `PARALLEL`, the projectors are all parallel to a vector defined by the projection reference point. The view volume is therefore a parallelpiped.

SunCore and SunPHIGS differ in their definition of the `PARALLEL` projection vector. In SunCore, the projection vector starts at the view reference point and extends to the projection reference point (in relative world coordinates). In SunPHIGS however, the projection vector starts at the *center of the view window* and extends to the projection reference point (in view reference coordinates).

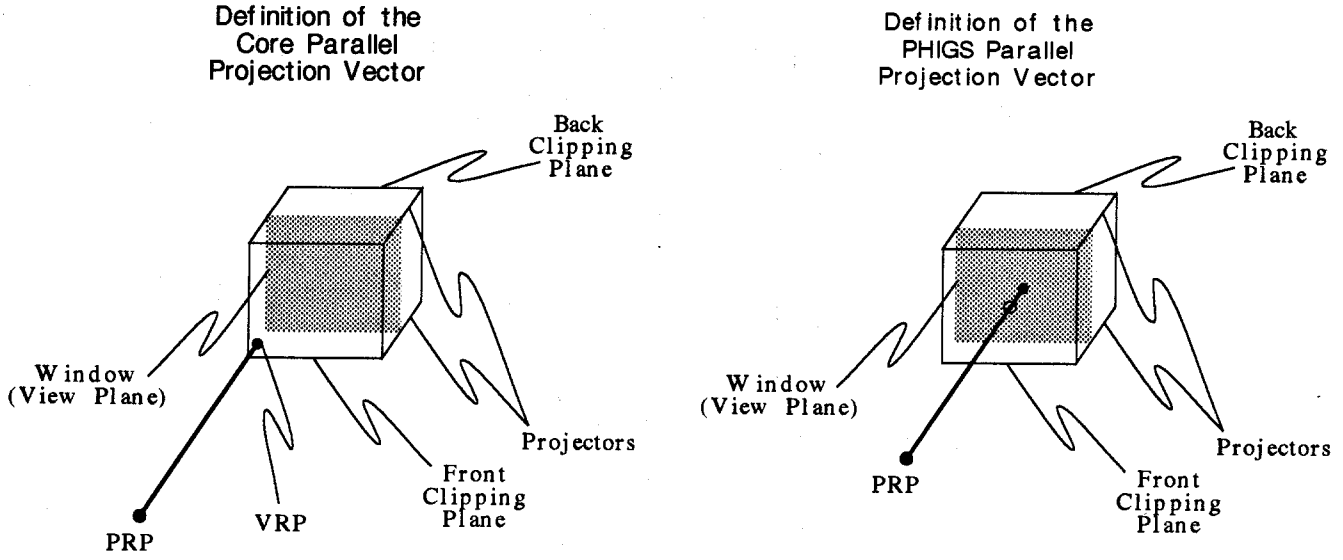


Figure 4-1: Sample Core and PHIGS View Volumes and Parallel Projectors

Viewing Example

Figure 4-2 shows a SunCore program fragment that defines a view volume for a created segment. Figure 4-3 shows how this same view volume would be assigned to a SunPHIGS structure. Numbers in **bold** represent parameters that differ between the graphics libraries.

As seen in the example, SunPHIGS viewing parameters *must* be set as a bundle (called a *view representation*). The `pevalvieworientationmatrix` and `pevalviewmappingmatrix` functions create view representation matrices that may then be passed to `psetviewrep`. The `psetviewrep` function places this bundled data into the SunPHIGS workstation state list with an assigned *view index*. Structures may then choose a specific view representation with the `psetviewind` function.

```
set_view_reference_point( 3.0, 1.0, 4.0);
set_view_plane_normal( -2.0, 0.0, -1.0);
set_view_up_3( -1.0, 0.0, 2.0);
set_view_plane_distance( 2.0);
set_view_depth( -1.0, 9.0);
set_projection( PERSPECTIVE, 3.5777, 0.0, 1.7888);
set_window( -20.0, 20.0, -10.0, 10.0);
set_viewport_3( 0.2, 0.7, 0.1, 0.6, 0.0, 1.0);
set_front_plane_clipping(TRUE); set_back_plane_clipping(TRUE);
set_window_clipping(TRUE);
create_retained_segment(5);
    ...
```

Figure 4-2: Sample SunCore Viewing Operation

```
Ppoint3        View_Ref_Pt = { 3.0, 1.0, 4.0 };
Pvector3       Normal_Vec  = { 2.0,  0.0, 1.0  };
Pvector3       Up_Vector   = {-1.0, 0.0, 2.0 };
Pviewmapping3  Mapping     = {  {-20.0, 20.0, -10.0, 10.0},
   /*    viewport    */          { 0.2, 0.7, 0.1, 0.6, 0.0, 1.0},
   /*   proj. type   */            PPERSPECTIVE ,
   /* Proj. Ref. Pt. */          { 0.0, 0.0, 4.0 },
   /* View Pln. Dist.*/           -2.0 ,
   /* Back Pln. Dist.*/           -9.0 ,
   /* Front Pl. Dist.*/           1.0                                  };
Pviewrep3   rep;

pevalvieworientationmatrix3( &View_Ref_Pt,
                             &Normal_Vec,
                             &Up_Vector,
                            &err,
                             rep.orientation_matrix);
pevalviewmappingmatrix3(     &Mapping,
                             &err,
                             rep.mapping_matrix);

rep.clip_limit = Mapping.viewport;
rep.clip_xy = PCLIP;
rep.clip_back = PCLIP;
rep.clip_front = PCLIP;

psetviewrep3( 1, 1, &rep);

popenstruct( 5);
    psetviewind( 1);
        ...

ppoststruct( 1, 5, 0.0);
        ...
```

Figure 4-3: Sample SunPHIGS Viewing Operation

Segmentation and Naming

PHIGS' structures have several advantages over Core's segments. Attributes and output primitives are stored in SunCore segments in a concise, device-readable form. Although this allows the screen to be refreshed quickly, the compressed segment data can not be edited.

Structure Elements

The powerful SunPHIGS structure, however, is a database entry that stores:

1. Untransformed Output Primitives
2. Primitive Attributes
3. View Selections

4. Modelling (local and global) Transformations
5. References (execution links) to other Structures
6. Labels to aid editing of Structure elements
7. Pick Identifiers
8. Application data
9. Name sets

Batching of Updates

Since SunCore supports incremental updates (new output primitives may be inserted without redrawing the entire display), most applications rarely use the `begin_batch_of_updates` function. In SunPHIGS, there are two reasons to commonly batch updates.

- First, by default SunPHIGS does not preform incremental updates. Each time an application makes a change to a structure (by drawing a line, for example), the entire display is erased and redrawn. Hierarchical structuring necessitates such time-consuming updates since a modified attribute in one structure may be inherited by several other structures.
- Second, SunPHIGS' lack of stored primitives in device coordinates motivates batching of updates. Since SunPHIGS structures store output primitives in modelling coordinates (for editing purposes), time-consuming transformation and clipping operations occur with each screen refresh.

Both performance problems are solved with strategic use of the `psetdisplayupdatest` function.

```
psetdisplayupdatest( 1,PWAIT,PNIVE)
```

has the same effect as the SunCore `begin_batch_of_updates` function, and

```
psetdisplayupdatest( 1,PASAP,PUWOR)
```

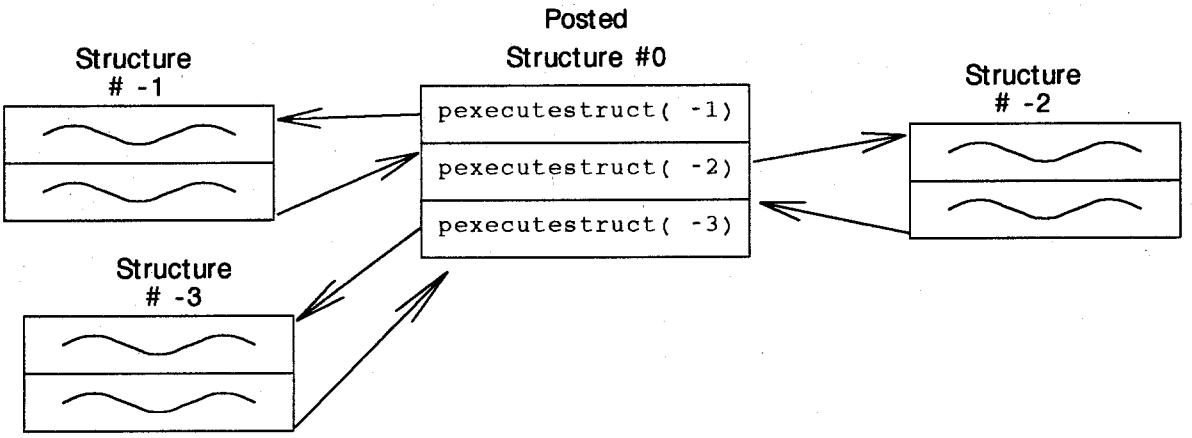
emulates the `end_batch_of_updates` function. Furthermore, SunPHIGS can be told to simulate incremental updates by using `PUQUM` (in place of `PNIVE`), along with `PWAIT` when beginning a batch of updates. Refer to the *SunPHIGS Reference Manual* (Set Display Update State) for a list of supported quick update methods.

Structure Posting

In SunCore, the `select_view_surface` function is used to direct the output of subsequently created segments to particular view surfaces. In SunPHIGS however, the `ppoststruct` function is used to attach structures to a workstation. Structures that are not explicitly posted to a workstation (unless executed by another posted structure), will not be drawn.

Temporary Segments

Unlike SunCore, SunPHIGS does not directly support temporary structures. The following figure shows how SunPHIGS' hierarchical structuring can be used to imitate temporary segments:




```
Pint    number_of_temp_structs = 0;
int temp_struct_open = 0;

create_temporary_segment()
{
  popenstruct( 0);
  ppoststruct( 1, 0, 0.0);
  pexecutestruct( --number_of_temp_structs);
  pclosestruct();
  popenstruct( number_of_temp_structs);
  temp_struct_open = 1;
}

new_frame()
{
  number_of_temp_structs = 0;
  pdelstructnet( 0, PDELETE);
  if (temp_struct_open) create_temporary_structure();
}

close_temporary_segment()
{
  pclosestruct();
  temp_struct_open = 0;
}
```

Figure 4-4: Imitation of Temporary Segments

This figure shows that SunPHIGS structures, unlike SunCore segments, can have negative identifiers. When the screen is refreshed, the posted structure #0 in the figure is executed, which in turn calls each negatively named, 'temporary' structure. Lastly, the `pdelstructnet` function emulates the SunCore

`new_frame` function by deleting all of the 'temporary' structures referenced by structure #0.

Chapter 2 in *Getting Started with SunPHIGS* explains SunPHIGS structures further.

Output Primitives

Of all the graphical functions, the output primitives are the most similar between the graphics libraries. Table 4-2 shows the relation of output primitives between SunCore and SunPHIGS.

SunCore	*SunPHIGS*
Move	
Line	Polyline
Polyline	
Polygon	Fill Area
	Fill Area Set
Text	Text
Marker	Polymarker
Polymarker	
Raster	Cellarray (minimal support [6]
	Generalized Drawing Primitive

Table 4-2: Output Primitives

Current Insertion Point

Unlike SunCore, SunPHIGS does not maintain a current drawing point. Instead, SunPHIGS output primitives receive absolutely-positioned modelling coordinates in their parameter lists. Thus, SunPHIGS has no `move_abs` or `move_rel` statements.

Relative Positioning

Many SunCore application programs have subroutines that draw specific objects relative to the current insertion point. In order to move an object, such applications continually erase the screen, set a new insertion point, and then re-call a drawing subroutine. SunPHIGS applications however, need only draw a complex object once in modelling coordinates into an unposted structure. A posted structure is then created and filled with a local transformation, and an execution link to the object's structure. Movement is achieved by simply editing the local transformation, thereby freeing the application of redrawing a complex object. Figures 4-5 and 4-6 demonstrate how SunCore's relative movements can be accomplished by SunPHIGS' local transformations.

[6] SunPHIGS release 1.0 does not correctly support cellarray operations. This will be corrected in a future version of SunPHIGS.

```
{
create_retained_segment(2);
    set_line_index(63);  /* red  */
    move_abs_2( 0.2, 0.5);
    draw_box();
    set_line_index(191); /* blue */
    move_abs_2( 0.6, 0.5);
    draw_box();
close_retained_segment();
}

draw_box()
{
  line_rel(0.2,0.0);   line_rel(0.0,0.2);
  line_rel(-0.2,0.0);  line_rel(0.0,-0.2);
}
```

Figure 4-5: Sample SunCore Polyline Routine

```
static Ppoint   box[] = { 0.0,0.0,   0.2,0.0,
                                0.2,0.2,   0.0,0.2,
                0.0,0.0      };
static Pvector  first_trans = { 0.2, 0.5 };
static Pvector  second_trns = { 0.6, 0.5 };

{
Pmatrix trans_matrix;
Pint    err;

popenstruct(-1);
        ppolyline( 5, box);
pclosestruct();
popenstruct(2);
        ptranslate( &first_trans, &err, trans_matrix);
        psetlocaltran( trans_matrix, PREPLACE);
        psetlinecolourind( 2);  /* red */
        pexecutestruct( -1);
        ptranslate( &second_trns, &err, trans_matrix);
        psetlocaltran( trans_matrix, PREPLACE);
        psetlinecolourind( 4);  /* blue */
        pexecutestruct( -1);
pclosestruct();
ppoststruct( 1, 2, 0.0);
}
```

Figure 4-6: Sample SunPHIGS Polyline Routine

Point Arrays

Another difference between SunCore and SunPHIGS is the format in which output primitive functions receive points. SunCore functions require a separate array for each dimension involved (an array of x-values, an array of y-values, and an array of z-values). However, SunPHIGS functions require a single array of point *structures* (an array of x, y pairs, or x, y, z triples).

Once again, refer to Appendix A and the *SunPHIGS Reference Manual.*

Attributes

Of the large number of SunCore attribute functions, almost all have a direct equivalent in SunPHIGS. Appendix A equates each SunCore attribute with its SunPHIGS counterpart, which may then be researched in the *SunPHIGS Reference Manual.* The following list details special attribute differences between the graphics systems:

- SunCore lines are approximately 4.4 times wider than their SunPHIGS counterparts.
- SunPHIGS 1.0 does not allow XORing of primitives to display memory. Therefore, SunPHIGS has no equivalent of the SunCore `set_rasterop` function.
- SunPHIGS 1.0[7] does not shade polygons. SunCore Phong or Gouraud shading may be emulated by splitting a region into a mesh of individually colored polygons. Shading algorithms are discussed in *Fundamentals of Interactive Computer Graphics*, Foley, J.D. and Van Dam, A., Addison-Wesley, Reading, Massachusetts, 1982.
- SunPHIGS (as discussed in Chapter 2) does not support image transformations. Instead, the `psetglobaltran` function may be used to transform Modelling coordinate space.
- SunCore and SunPHIGS initialize the color table differently. The default color indices for an eight bit-plane device are shown in table 4-3.

[7] Shading models are not part of the PHIGS standard. SunPHIGS+ will provide additional features, including lighting and shading support.

Index	*SunCore*			*SunPHIGS*		
	Red	Green	Blue	Red	Green	Blue
0	0.5	0.5	0.5	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	1.0	1.0
2	0.016	0.0	0.0	1.0	0.0	0.0
3		...		0.0	1.0	0.0
4				0.0	0.0	1.0
5				1.0	1.0	0.0
6	Intensity Color Ramp in Red			0.0	1.0	1.0
7				1.0	0.0	1.0
8				1.0	1.0	1.0
...		...			...	
63	1.0	0.0	0.0			
64	0.0	0.016	0.0			
...	Intensity Color Ramp in Green	...				
127	0.0	1.0	0.0			
128	0.0	0.0	0.016			
...	Intensity Color Ramp in Blue	...			White	
191	0.0	0.0	1.0			
192	0.016	0.016	0.0			
...	Intensity Color Ramp in Yellow (Red + green)	...				
255	1.0	1.0	0.0	1.0	1.0	1.0

Table 4-3: Default Color Tables

Input Primitives

Table 4-4 compares the input primitives of the two graphics libraries. For a complete introduction to SunPHIGS' input routines, refer to *Getting Started with SunPHIGS.*

Core Primitive	Value Returned	PHIGS Primitive	Value Returned
Pick	(int)	Pick	PickID (int)
Keyboard	Null-terminated string	String	Null-terminated string
Button	Mouse 1,2,3 (int)	Choice	Integer from 1 to N
Stroke	(x,y) points in NDC	Stroke	(x,y) or (x,y,z) points in WC
Locator	(x,y) point in NDC	Locator	(x,y) or (x,y,z) point in WC
Valuator	float	Valuator	float

Table 4-4: Input Primitives

Appendix A

Appendix A: SunCore C Functions with Related SunPHIGS Functions

This appendix connects each SunCore function with related functions in the *SunPHIGS Reference Manual.* In most cases, SunPHIGS' functions require different parameters than their SunCore counterparts. SunPHIGS programs written in C must contain the following statement at the start of each SunPHIGS source file.

```
#include <phigs.h>
```

Most PHIGS attributes may be set either individually, or as a bundle. SunPHIGS functions with a `-rep` or `-ind` suffix operate on bundled attributes. Whether a structure uses individual or bundled attributes is controlled by the SunPHIGS `psetindivasf` function.

SunCore	*SunPHIGS*
allocate_raster	**pcellarray** **pcellarray3**
await_any_button	**pawaitevent** **pgetchoice** **preqchoice** **psamplechoice**
await_any_button_get_locator_2	**pawaitevent** **pgetloc** **pgetloc3** **preqloc** **preqloc3** **psampleloc** **psampleloc3**
await_any_button_get_valuator	**pawaitevent** **pgetval** **preqval** **psampleval**
await_keyboard	**pawaitevent** **pgetstring** **preqstring** **psamplestring**
await_pick	**pawaitevent** **pgetpick** **preqpick** **psamplepick**
await_stroke_2	**pawaitevent** **pgetstroke** **pgetstroke3**

— Continued

SunCore	*SunPHIGS*
	preqstroke **preqstroke3** **psamplestroke** **psamplestroke3**
begin_batch_of_updates	**psetdisplayupdatest**
close_retained_segment	**pclosestruct**
close_temporary_segment	**pclosestruct**
create_retained_segment	**popenstruct** **ppoststruct**
create_temporary_segment	**popenstruct** **ppoststruct**
define_color_indices	**psetcolourrep**
delete_all_retained_segments	**pdelstructnet** **pdelallstruct**
delete_retained_segment	**pdelstruct**
deselect_view_surface	*No direct equivalent*
end_batch_of_updates	**pupdatews** **psetdisplayupdatest** **predrawallstruct**
file_to_raster	**popenarfile** **pretrievestruct** **pclosearfile**
free_raster	**pcellarray** **pcellarray3**
get_mouse_state	**psamplechoice** **psampleloc3** **psampleloc** **psamplestroke3** **psamplestroke** **psampleval** **psamplepick** **psamplestring**
get_raster	**pcellarray** **pcellarray3**
get_view_surface	**phigs_ws_type_get**
initialize_core	**popenphigs** **psethlhsrid** **psethlhsrmode**
initialize_device	**pinitchoice** **pinitchoice3** **psetchoicemode** **pinitloc** **pinitloc3**

— Continued

SunCore	*SunPHIGS*
	psetlocmode **pinitpick** **pinitpick3** **psetpickmode** **pinitstring** **pinitstring3** **psetstringmode** **pinitstroke** **pinitstroke3** **psetstrokemode** **pinitval** **pinitval3** **psetvalmode**
initialize_view_surface	**popenws** **phigs_ws_type_create** **phigs_ws_type_set**
inquire_charjust	**pinqtextrep** **pinqpredtextrep** **pinqcurelemcontent** **pinqelemcontent**
inquire_charpath_2	*See inquire_charjust*
inquire_charpath_3	*See inquire_charjust*
inquire_charprecision	*See inquire_charjust*
inquire_charsize	*See inquire_charjust*
inquire_charspace	*See inquire_charjust*
inquire_charup_2	*See inquire_charjust*
inquire_charup_3	*See inquire_charjust*
inquire_color_indices	**pinqcolourind**
inquire_current_position_2	*No direct equivalent*
inquire_current_position_3	*No direct equivalent*
inquire_detectability	**pinqelemcontent** **pinqcurelemcontent**
inquire_echo	**pinqchoicest** **pinqchoicest3** **pinqdefchoicedata** **pinqdefchoicedata3** **pinqlocst** **pinqlocst3** **pinqdeflocdata** **pinqdeflocdata3** **pinqpickst** **pinqpickst3** **pinqdefpickdata** **pinqdefpickdata3**

— Continued

SunCore	*SunPHIGS*
	pinqstringst **pinqstringst3** **pinqdefstringdata** **pinqdefstringdata3** **pinqstrokest** **pinqstrokest3** **pinqdefstrokedata** **pinqdefstrokedata3** **pinqvalst** **pinqvalst3** **pinqdefvaldata** **pinqdefvaldata3**
inquire_echo_position	*See inquire_echo*
inquire_echo_surface	*See inquire_echo*
inquire_fill_index	**pinqintrep** **pinqpredintrep** **pinqcurelemcontent** **pinqelemcontent**
inquire_font	*See inquire_charjust*
inquire_highlighting	**pinqelemcontent** **pinqcurelemcontent**
inquire_image_transformation_2	*No direct equivalent*
inquire_image_transformation_3	*No direct equivalent*
inquire_image_transformation_type	*No direct equivalent*
inquire_image_translate_2	*No direct equivalent*
inquire_image_translate_3	*No direct equivalent*
inquire_inverse_composite_matrix	*No direct equivalent*
inquire_keyboard	**pinqstringst** **pinqstringst3** **pinqdefstringdata** **pinqdefstringdata3**
inquire_line_index	**pinqlinerep** **pinqlineind** **pinqcurelemcontent** **pinqelemcontent**
inquire_linestyle	**pinqlinerep** **pinqlineind** **pinqcurelemcontent** **pinqelemcontent**
inquire_linewidth	**pinqlinerep** **pinqlineind** **pinqcurelemcontent** **pinqelemcontent**

— Continued

SunCore	*SunPHIGS*
inquire_locator_2	**pinqlocst** **pinqlocst3**
inquire_marker_symbol	**pinqmarkerrep** **pinqmarkerind** **pinqmarkerfacil** **pinqpredmarkerrep** **pinqcurelemcontent** **pinqelemcontent**
inquire_ndc_space_2	**pinqwstran**
inquire_ndc_space_3	**pinqwstran3**
inquire_open_retained_segment	**pinqopenstruct**
inquire_open_temporary_segment	**pinqopenstruct**
inquire_pen	*Not implemented in SunCore*
inquire_pick_id	**pinqpickst** **pinqpickst3** **pinqdefpickdata** **pinqdefpickdata3** **pinqelemcontent** **pinqcurelemcontent**
inquire_polygon_edge_style	**pinqedgerep** **pinqedgeind** **pinqedgefacil** **pinqprededgerep** **pinqcurelemcontent** **pinqelemcontent**
inquire_polygon_interior_style	**pinqintrep** **pinqintind** **pinqintfacil** **pinqpredintrep** **pinqcurelemcontent** **pinqelemcontent**
inquire_primitive_attributes	**pinqlinerep** **pinqlineind** **pinqintrep** **pinqintind** **pinqtextrep** **pinqtextind** **pinqedgerep** **pinqedgeind** **pinqmarkerrep** **pinqmarkerind** **pinqcurelemcontent** **pinqelemcontent**
inquire_projection	**pinqviewrep**

— Continued

SunCore	*SunPHIGS*
	pinqviewind **pinqcurelemcontent** **pinqelemcontent**
inquire_rasterop	*No direct equivalent*
inquire_retained_segment_names	**pinqstructids**
inquire_retained_segment_surfaces	**pinqsetopenws** **pinqsetwsposted**
inquire_segment_detectability	**pinqelemcontent** **pinqcurelemcontent**
inquire_segment_highlighting	**pinqhilightfilter** **pinqelemcontent** **pinqcurelemcontent**
inquire_segment_image_transformation_2	*No direct equivalent*
inquire_segment_image_transformation_3	*No direct equivalent*
inquire_segment_image_translate_2	*No direct equivalent*
inquire_segment_image_translate_3	*No direct equivalent*
inquire_segment_visibility	**pinqpostedstruct** **pinqinvisfilter**
inquire_stroke	**pinqstrokest** **pinqstrokest3**
inquire_text_extent_2	**pinqtextextent**
inquire_text_extent_3	**pinqtextextent**
inquire_text_index	**pinqtextrep** **pinqpredtextrep** **pinqtextind** **pinqcurelemcontent** **pinqelemcontent**
inquire_valuator	**pinqvalst** **pinqvalst3**
inquire_view_depth	**pinqviewrep** **pinqviewind** **pinqcurelemcontent** **pinqelemcontent**
inquire_view_plane_distance	*See inquire_view_depth*
inquire_view_plane_normal	*See inquire_view_depth*
inquire_view_reference_point	*See inquire_view_depth*
inquire_view_up_2	*See inuquire_view_depth*
inquire_view_up_3	*See inquire_view_depth*
inquire_viewing_control_parameters	*See inquire_view_depth*
inquire_viewing_parameters	*See inquire_view_depth*
inquire_viewport_2	*See inquire_view_depth*
inquire_viewport_3	*See inquire_view_depth*

— Continued

SunCore	*SunPHIGS*
inquire_visibility	**pinqpostedstruct** **pinqinvisfilter**
inquire_window	*See inquire_view_depth*
inquire_world_coordinate_matrix_2	**pinqelemcontent** **pinqcurelemcontent**
inquire_world_coordinate_matrix_3	**pinqelemcontent** **pinqcurelemcontent**
line_abs_2	**ppolyline** **psetlineind** **pinqlinefacil**
line_abs_3	**ppolyline3** **psetlineind** **pinqlinefacil**
line_rel_2	**ppolyline** **psetlineind** **pinqlinefacil**
line_rel_3	**ppolyline3** **psetlineind** **pinqlinefacil**
map_ndc_to_world_2	**ptranpt**
map_ndc_to_world_3	**ptranpt3**
map_world_to_ndc_2	**ptranpt**
map_world_to_ndc_3	**ptranpt3**
marker_abs_2	**ppolymarker** **psetmarkerind** **pannotationtextrelative** **ptext** **psettextind**
marker_abs_3	**ppolymarker3** **psetmarkerind** **pannotationtextrelative3** **ptext3** **psettextind**
marker_rel_2	**ppolymarker** **psetmarkerind** **pannotationtextrelative** **ptext** **psettextind**
marker_rel_3	**ppolymarker3** **psetmarkerind** **pannotationtextrelative3** **ptext3** **psettextind**

— Continued

SunCore	*SunPHIGS*
move_abs_2	*No direct equivalent*
move_abs_3	*No direct equivalent*
move_rel_2	*No direct equivalent*
move_rel_3	*No direct equivalent*
new_frame	**pdelstruct** **pupdatews** **predrawallstruct**
polygon_abs_2	**pfillarea** **pfillareaset** **psetintind** **pinqintfacil**
polygon_abs_3	**pfillarea3** **pfillareaset** **psetintind** **pinqintfacil**
polygon_rel_2	**pfillarea** **pfillareaset** **psetintind** **pinqintfacil**
polygon_rel_3	**pfillarea3** **pfillareaset3** **psetintind** **pinqintfacil**
polyline_abs_2	**ppolyline** **psetlineind** **pinqlinefacil**
polyline_abs_3	**ppolyline3** **psetlineind** **pinqlinefacil**
polyline_rel_2	**ppolyline** **psetlineind** **pinqlinefacil**
polyline_rel_3	**ppolyline3** **psetlineind** **pinqlinefacil**
polymarker_abs_2	**ppolymarker** **psetmarkerind** **pannotationtextrelative** **ptext** **psettextind**
polymarker_abs_3	**ppolymarker3** **psetmarkerind** **pannotationtextrelative3**

— Continued

SunCore	*SunPHIGS*
	ptext3 **psettextind**
polymarker_rel_2	**ppolymarker** **psetmarkerind** **pannotationtextrelative** **ptext** **psettextind**
polymarker_rel_3	**ppolymarker3** **psetmarkerind** **pannotationtextrelative3** **ptext3** **psettextind**
print_error	**pinqerrorhandmode** **perrorhand** **perrorlog** **pseterrorhandmode** **pemergencyclosephigs**
put_raster	**pcellarray** **pcellarray3** **psetindivasf** **psetlinetype** **psetlinewidth** **psetlinecolourind**
raster_to_file	**popenarfile** **pcellarray3** **pcellarray** **parstruct** **pclosearfile**
rename_retained_segment	**pchangestructidref**
report_most_recent_error	**pinqerrorhandmode** **perrorhand** **perrorlog** **pseterrorhandmode** **pemergencyclosephigs**
restore_segment	**popenarfile** **pretrievestructids** **pretrievestruct** **pretrievestructnet** **pretrieveallstruct** **pclosearfile** **pinqarfiles**
save_segment	**popenarfile** **parstruct** **parstructnet**

— Continued

SunCore	*SunPHIGS*
	parallstruct **pclosearfile**
select_view_surface	**ppoststruct**
set_back_plane_clipping	**psetviewrep3** **psetviewrep**
set_charjust	**psettextalign**
set_charpath_2	**psettextpath**
set_charpath_3	**psettextpath**
set_charprecision	**psettextprec** **psettextrep** **psetindivasf**
set_charsize	**psetcharheight** **psetcharexpan** **psettextrep** **psetindivasf**
set_charspace	**psetcharspace** **psettextrep** **psetindivasf**
set_charup_2	**psetcharup**
set_charup_3	**psetcharup**
set_coordinate_system_type	*No direct equivalent*
set_detectability	**psetpickfilter**
set_drag	*No direct equivalent*
set_echo	**psetchoicemode** **psetlocmode** **psetpickmode** **psetstringmode** **psetstrokemode** **psetvalmode** *See initialize_device*
set_echo_group	*See set_echo*
set_echo_position	*See initialize_device*
set_echo_surface	*See initialize_device*
set_fill_index	**psetintcolourind** **psetintrep** **psetintind** **psetindivasf**
set_font	**psettextfont** **psettextrep** **psetindivasf**
set_front_plane_clipping	**psetviewrep3** **psetviewrep**

— Continued

SunCore	*SunPHIGS*
set_highlighting	**psethilightfilter** **pgse** **paddnameset** **premovenameset**
set_image_transformation_2	**psetglobaltran** **psetlocaltran** **ptranslate** **pscale** **protate** **pbuildtran**
set_image_transformation_3	**psetglobaltran3** **psetlocaltran** **ptranslate3** **pscale3** **protatex** **protatey** **protatez** **pbuildtran3**
set_image_transformation_type	*No direct equivalent*
set_image_translate_2	**psetglobaltran** **psetlocaltran** **ptranslate** **pbuildtran**
set_image_translate_3	**psetglobaltran3** **psetlocaltran3** **ptranslate3** **pbuildtran3**
set_keyboard	**pinitstring** **pinitstring3**
set_light_direction	*No direct equivalent*
set_line_index	**psetlinecolourind** **psetlinerep** **psetindivasf**
set_linestyle	**psetlinetype** **psetlinerep** **psetindivasf**
set_linewidth	**psetlinewidth** **psetindivasf**
set_locator_2	**pinitloc** **pinitloc3**
set_marker_symbol	**psetmarkertype** **psetmarkerrep** **psetmarkersize** **psetannotationcharheight** **psetannotationcharup** **psetannotationpath**

— Continued

SunCore	*SunPHIGS*
	psetannotationalign **psetannotationstyle** **psetindivasf**
set_ndc_space_2	**psetwswindow** **psetwsviewport**
set_ndc_space_3	**psetwswindow3** **psetwsviewport3**
set_output_clipping	*No direct equivalent*
set_pen	*No implemented in SunCore*
set_pick_id	**psetpickid**
set_polygon_edge_style	**psetedgetype** **psetedgewidth** **psetedgecolourind** **psetedgeflag** **psetedgerep** **psetedgeind** **psetindivasf**
set_polygon_interior_style	**psetintstyle** **psetintrep** **psetintstyleind** **psetindivasf**
set_primitive_attributes	**psetlinecolourind** **psetlinerep** **psetintcolourind** **psetintrep** **psettextcolourind** **psetmarkercolourind** **psettextrep** **psetmarkerrep** **psetlinetype** **psetlinerep** **psetintstyle** **psetintrep** **psetintstyleind** **psetedgetype** **psetedgewidth** **psetedgecolourind** **psetedgeflag** **psetedgerep** **psetedgeind** **psetindivasf** **psetlinewidth** **psettextfont** **psetcharheight**

— Continued

SunCore	*SunPHIGS*
	psetcharexpan **psetcharup** **psettextpath** **psetcharspace** **psettextalign** **psettextprec** **psetmarkertype** **psetindivasf**
set_projection	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep** **psetviewrep3** **psetviewind**
set_rasterop	*No direct equivalent*
set_segment_detectability	**psetpickfilter**
set_segment_highlighting	**psethilightfilter** **pgse** **paddnameset** **premovenameset**
set_segment_image_transformation_2	**psetlocaltran** **psetglobaltran** **ptranslate** **pscale** **protate** **pbuildtran**
set_segment_image_translate_2	**psetlocaltran** **psetglobaltran** **ptranslate** **pbuildtran**
set_segment_image_translate_3	**psetlocaltran3** **psetglobaltran3** **ptranslate3** **pbuildtran3**
set_segment_image_transformation_3	**psetlocaltran3** **psetglobaltran3** **ptranslate3** **pscale3** **protatex** **protatey** **protatez** **pbuildtran3**
set_segment_visibility	**ppoststruct** **punpoststruct** **psetinvisfilter**

— Continued

SunCore	*SunPHIGS*
set_shading_parameters	*No direct equivalent*
set_stroke	**pinitstroke** **pinitstroke3**
set_text_index	**psettextcolourind** **psettextrep** **psetmarkercolourind** **psetmarkerrep** **psetindivasf**
set_valuator	**pinitval** **pinitval3**
set_vertex_indices	*No direct equivalent*
set_vertex_normals	*No direct equivalent*
set_view_depth	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_view_plane_distance	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_view_plane_normal	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_view_reference_point	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep** **psetviewrep3** **psetviewind**
set_view_up_2	**pevalvieworientationmatrix** **pevalviewmappingmatrix** **psetviewrep** **psetviewind**
set_view_up_3	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_viewing_parameters	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_viewport_2	**pevalvieworientationmatrix** **pevalviewmappingmatrix** **psetviewrep**

— Continued

SunCore	*SunPHIGS*
	psetviewind
set_viewport_3	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep3** **psetviewind**
set_visibility	**ppoststruct** **punpoststruct** **psetinvisfilter**
set_window	**pevalvieworientationmatrix3** **pevalviewmappingmatrix3** **psetviewrep** **psetviewrep3** **psetviewind**
set_window_clipping	**psetviewrep3** **psetviewrep**
set_world_coordinate_matrix_2	**psetglobaltran** **pbuildtran**
set_world_coordinate_matrix_3	**psetglobaltran3** **pbuildtran**
set_zbuffer_cut	*No direct equivalent*
size_raster	**pcellarray** **pcellarray3**
terminate_core	**pclosephigs**
terminate_device	*No direct equivalent*
terminate_view_surface	**pclosews**
text	**ptext** **ptext3** **psettextind**

Appendix B

Appendix B: An Example Conversion

This appendix shows an example conversion of a SunCore program to a SunPHIGS program.

```
#include <usercore.h>

                                /* Relative coordinates */
static float glassdx[] = { -10.0,   9.0,  0.0, -14.0, 30.0,
                           -14.0,   0.0,  9.0, -10.0       };
static float glassdy[] = {   0.0,   1.0, 19.0,  15.0,  0.0,
                           -15.0, -19.0, -1.0,   0.0       };

int pixwindd();
struct vwsurf vwsurf = DEFAULT_VWSURF(pixwindd);

main()
{
    initialize_core(BASIC, NOINPUT, TWOD);
    initialize_view_surface(&vwsurf, FALSE);
    select_view_surface(&vwsurf);
    set_viewport_2(0.125, 0.875, 0.125, 0.75);
    set_window(-50.0, 50.0, -10.0, 80.0);

    create_temporary_segment();
        move_abs_2(0.0, 0.0);
        polyline_rel_2(glassdx, glassdy, 9);
        move_rel_2(-12.0, 33.0);
        line_rel_2( 24.0,  0.0);
    close_temporary_segment();

    sleep(10);

    deselect_view_surface(&vwsurf);
    terminate_core();
}
```

Figure B-1: The Core_glass.c Example

The PHIGS example program appears in figure B-2 on the following pages.

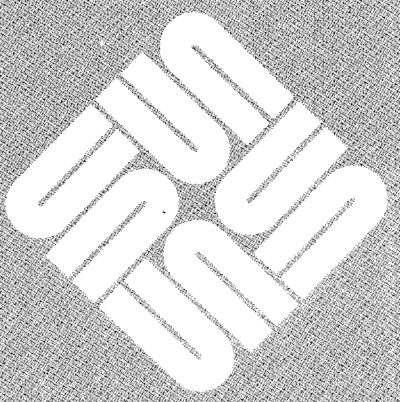
```
#include <phigs/phigs.h>

Ppoint glass_pts[] = {  {   0.0,  0.0 },
                        { -10.0,  0.0 },
                        {  -1.0,  1.0 },
  /* Absolute    */     {  -1.0, 20.0 },
  /* coordinates */     { -15.0, 35.0 },
                        {  15.0, 35.0 },
                        {   1.0, 20.0 },
                        {   1.0,  1.0 },
                        {  10.0,  0.0 },
                        {   0.0,  0.0 }  };

Ppoint rim_pts[] = {    { -12.0, 33.0 },
                        {  12.0, 33.0 }  };

Ppoint        View_Ref_Pt = { 0.0, 0.0 };
Pvector       Up_Vector   = { 0.0, 1.0 };
Pviewmapping  Mapping     = { {-50.0, 50.0, -10.0, 80.0},
   /*    viewport    */       { 0.125, 0.875, 0.125, 0.75},
                                                  };


main()
{
Pint       err;
Pupdatest  update;
Pviewrep   rep;
Pmatrix    matrix;

   popenphigs((Pchar*)NULL, (Plong)0);
   popenws(1, (Pconnid)NULL, phigs_ws_type_sun_tool);

   pevalvieworientationmatrix( &View_Ref_Pt,
                               &Up_Vector,
                               &err,
                               rep.orientation_matrix);
   pevalviewmappingmatrix(     &Mapping,
                               &err,
                               rep.mapping_matrix);

   rep.clip_limit = Mapping.viewport;
   rep.clip_xy = PCLIP;

   psetviewrep( 1, 1, &rep);
   popenstruct(1);
       psetviewind(1);
       ppolyline(10, glass_pts);
       ppolyline( 2, rim_pts  );
   pclosestruct();

   ppoststruct( 1, 1, 0.0);
```

```
    sleep(10);
    pclosews(1);
    pclosephigs();
}
```

Figure B-2: The PHIGS_glass.c Example Program

5

HINTS AND TIPS



5

HINTS AND TIPS

`cgfour` : Moving Windows

Moving Windows Faster with Sun-3s and `cgfour` Framebuffers

Customers using Sun-3 workstations with `cgfour` framebuffers may have noticed that moving windows takes longer at some times than others. This article contains a hint that works for OpenWindows (not for SunView).[8]

The Problem

At some times if you move a window from one place to another, you can watch the server copy copy the bits from the source to the destination. The window appears to 'wipe' into its new location. At other times, however, the window is copied quickly.

The Problem Defined

The slow movement of windows results from the relative alignment of the source and destination locations. Fast copying results about once every four times.

If the destination is not word-aligned with respect to the source location, the `bitblt` code then must select each word of framebuffer memory, shift and mask it around, and then store it back into the framebuffer. If the source and destination are word-aligned, the `bitblt` code copies entire words without shifting or masking. This results in greatly improved performance.

The Hint

If you are a PostScript programmer, try the code appearing in the last section of this hints and tips article in your `.startup.ps` file.

The code ensures that windows are always positioned in an x location of `0 mod 4`. This causes the rasters containing the window's bits to be in alignment relative to each other.

You will now get the fast `bitblt` processing when you move a window. You give up the ability to position windows at arbitrary locations, though.

---

[8] This article is submitted by Stuart Marks, Window Systems Group, Mountain View, California, USA.

The Code

The below hit is an example of the synergy that results from tuning the system as a whole instead of as isolated pieces.

```
UserProfile begin
    /OpenLookFrame { % name class => name class

    /move { % x y => -
        exch 2 add 4 idiv 4 mul exch
        /move super send
    } /installmethod 3 index send

    /reshape { % x y w h => -
        4 -1 roll 2 add 4 idiv 4 mul 4 1 roll
        /reshape super send
    } /installmethod 3 index send

    } def
end
```

Sun386*i* enscript

DOS Landscape Printing Hints

This article contains some hints on how to get 132-column printed output from a DOS program that outputs directly to `LPT1`.

Hint Number 1

You can get 132-column printed output using `enscript`. Redefine `LPT1` in your `~/pc/setup.pc` file. One example that has been used with success appears below.

```
LPT1: enscript -r -f Courier7
```

Hint Number 2

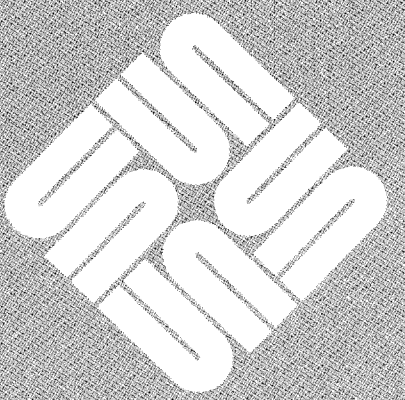
If you want to store printed output in a file, send printer output to `LPT2`. By default, output sent to this DOS 'printer' is actually appended to the `lpt-2` file in your home directory.

Hint Number 3

You can pipe DOS printer output through any combination of SunOS scripts and filters by changing the definitions in your `~/pc/setup.pc` file.

6

THE HACKERS' CORNER



6

THE HACKERS' CORNER

The `mush` Mail Utility

`mush` Mail User's Shell

The Mail User's Shell (`mush`) is an interface for sending and manipulating a database of electronic mail messages under the UNIX environment, much like other mail programs. `mush` has more commands for manipulating, finding and grouping messages, and in addition to customary user interface, also includes curses, visual, and SunView mode with a macro facility.

Included in the `mush` code are `sample.mushrc` and `advanced.mushrc` files.[9]

Initialization

When reading the initialization file, `mush` will recognize the `#` character as a comment delimiter. It may be placed anywhere in the file. When encountered as in the example below, processing of the line is discontinued to the end of the line:

```
set shell = /bin/csh # set the environment variable
```

When the `#` is enclosed in either single or double quotes, as in `'#'` or `"#"`, it is not considered a comment, as shown in the example below:

```
set prompt = "Message #%m: "  # The '#' is within quotes
```

String evaluation is allowed in `if` expressions, and the operators `==` and `!=` may be used to determine equality or inequality. Variables are compared with constants for evaluation. Note that it is not possible to compare variables to an empty string, and variables that evaluate to an empty string may cause errors.

---

[9] The `mush` code does not appear in this **Hackers' Corner** since it is over 30 pages long. For an online copy of the `mush` code, simply send email to *sun!stb-editor*. Please specify that you want the October 1989 **Hackers' Corner** code.

Environment variables are controlled by the `setenv` and `unsetenv` commands. Shell variables are controlled by the `set` and `unset` commands. When evaluating shell variables, if they are not found, the environment variables are searched.

Options can be *boolean*, in which case it is only significant to see whether or not they are set; *string*, in which case the actual value is of interest; or *numerical*, in which case the numerical value is important.

After sourcing the initialization file, `mush` will read mail in the specified folder and create a list of messages. The maximum number of messages the user can load is set to 1000 by default. If the `sort` variable is set, the messages are sorted according to the value of the variable. It may set to sort by author, date, subject, and so forth.

Each message has a number of message header lines that contain information about the sender, the subject, the date it was received, and information about the letter. This information is then compiled into a one-line summary for each message and is printed.

Arguments

The following command line arguments are understood by `mush` and can be typed at the command line prompt. Most arguments have abbreviations and can be followed by message lists. In most cases, spaces are not necessary to separate commands from message lists. For example, both `d*` and `d *` will delete all messages.

`-b bcc-list (-blindcarbon, -blind)`
The list of Blind Carbon Copy recipients is set on the command line. If more than one address or an address containing spaces is specified, the entire list should be enclosed in quotes.

`-C (-curses)`
Enter the mailer in curses mode upon startup.

`-c cc-list (-carbon, -copy)`
The list of Carbon Copy recipients is set on the command line. If more than one address or an address containing spaces is specified, the entire list should be enclosed in quotes.

`-F[!] filename (-source)`
This option allows commands that manipulate or search messages to be given. Normally, such commands will not appear in the initialization file since that file is read before the folder is scanned. The file specified by `-F` is read after the folder is scanned. The optional `'!'` argument prevents the shell from running after the file has been sourced. This is useful for managing mail automatically without having to interact with the program.

`-f [ filename ] (-folder)`
The optional filename argument specifies a folder containing mail messages. With no argument, `mbox` in the current directory is used.

`-H[:c] (-headers)`
The header option lets `mush` display mail headers without entering the shell.

`-i (-interact)`
This argument forces interactive mode when input has been redirected to the program. This is intended for remote host mail sessions but also allows the user to redirect scripts of `mush` commands.

`-N (-noheaders)`
The noheader option enters `mush` without displaying any message headers.

`-n (-noinit)`
This argument sees that no initialization is done on startup, and that the `.mushrc` or `.mailrc` files are not sourced.

`-r (-readonly)`
The read-only argument is passed on to the folder command. It initializes the folder in Read-Only mode; no modification of the folder is permitted.

`-S (-shell)`
This flag allows the user to enter the shell even if the system mailbox or specified folder is empty or does not exist.

`-s subject (-subject)`
The subject is set on the command line using this flag. If the subject has any spaces or tabs, the entire subject should be enclosed in quotes.

`-T timeout (-timeout)`
The timeout argument specifies the length of time, in seconds, to wait between each check for new mail. This applies in `suntools` mode only.

`-t (-tool)`
This flag allows the user to run `mush` using the graphics tool mode. This option is going to be deleted in the future. Currently, it must be the *first* argument on the command line if it is to be used. Otherwise, `toolmode` starts up automatically if the program name ends in 'tool' or 'view', e.g. `mushtool` or `mushview`

`-m mailbox-path (-mailbox)`
The mailbox specified in this option will be interpreted as if it were the system mailbox in place of `/usr/spool/mail/$USER`.

`-v (-verbose)`
This option is passed to the actual mail delivery subsystem internal to the user's version of UNIX. Some mailer systems, such as some System V systems, do not have a verbose option.

Files

The `/usr/spool/mail/*` files created by the program have read/write access to the owner only. Group and other permissions are not set. All other files created by the user via internal or external commands to the program have permissions set by the user's default `umask`. If the `umask` is reset within the program, the mask remains intact. The user must remember to set the variable `unix` before attempting to set the `umask` value.

```
/usr/spool/mail/*        Directory for incoming mail
```

If the system is using NFS, note that filesystems mounted for read/write access should be mounted as hard NFS mounts.

Unique `mush` Features

The following are features unique to `mush` that are not found in regular mail.

- `csh`-*style Interface*

  `mush` supports `history`, command line aliases, and command 'piping' just as `csh` does. Those familiar with `csh` will find using `mush` quite simple.

- *Programmable Interface for Curses Mode*

  By default, `mush` looks and acts like `vi`, but can be configured to work like `emacs`. You can build your own macros to do any number of simple or complicated `mush` commands.

- *Automatic Scanning for New Mail*

  No longer do you need to quit `mail` and reenter it just to get the new messages. `mush` automatically scans for new mail between each command.

- *Sorting Mail*

  You can sort messages by author, date, subject, and the like.

- *Searching for Specific Messages*

  Using the `pick` command, you can search for all messages from particular authors, by date, between dates, by subject, and so forth. Examples are shown below.

  ```
  pick -f argv
  ```

  Picks all messages from `argv`.

  ```
  pick -t junk | d
  ```

  Picks all messages that are addressed To: junk and deletes them.

  ```
  pick -s Status Report | save +status | delete
  ```

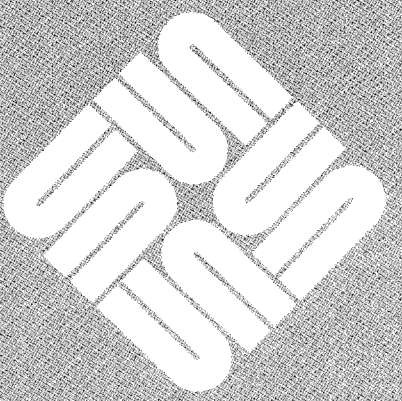
  Picks all messages that have the Subject: Status Report in them, saves them in your folder `directory/status` file, and then deletes the messages.

- *Editable Headers*

  You can edit your outgoing message headers. If you `reply-all` to a message and the list is very large, but contains the name of someone you wish to remove, then you can edit the list using your editor.

7

HARDWARE, CONFIGURATIONS, & UPGRADES



7

HARDWARE, CONFIGURATIONS, & UPGRADES

Sun386*i* Serial Port Voltages

Sun386*i* Serial Port Voltages and Performance

Customers may notice odd voltages on unconnected input pins of the Sun386*i* serial communication port (-3.5V on pin 3, `ReceiveData`, for example). They may think these odd voltages are the cause of problems such as poor throughput.[10]

The Hardware Design

The hardware design and these odd voltages do not result in poor throughput. The voltages one might measure on input pins when no device is connected to the serial port result from internal pull-up resistors.

The pull-up resistors are included in the circuit to prevent electrical noise getting into an unconnected port. The odd voltages disappear when a device is connected to the port.

The Problem: Cable Causes Poor Serial Port Performance

Poor serial port throughput may result from using an RS-232 cable with *too many wires* in it. Most modems do not have connections to all RS-232 pins. For example, modems intended for use only with leased lines probably do not have a connection to pin 21, `RingIndicator`.

If there are wires in the cable attached to unused pins, they act as antennas. They pick up signals from the other wires and may cause thousands of spurious interrupts on control pins. This problem is most severe with synchronous modems, where pins 15 and 17 carry clocking signals.

The Solution

If your RS-232 cable has wires that do not connect to a pin on your modem, ground the unused cable wires at the modem end.

Terminating unused wires in an RS-232 cable is a good practice for *all* serial communications ports, not just those on a Sun386*i*.

---

[10] This hardware article is submitted by Chuck Kollars, Marketing Engineering Support, Boston Development Center.

Sun386*i* Parallel Port & ATs

Sun386*i* Parallel Port Compatibility

The Sun386*i* parallel port is fully hardware IBM-AT compatible. Like the AT, the Sun386*i* parallel port is out-only. This arrangement is *not* IBM PS-2 compatible. PS-2 parallel ports are fully 8-bit bidirectional.

Hardware Reference

See the IBM Personal Computer Hardware Reference Library manuals listed below for information on the AT and for options and adapters used to connect to parallel port devices.

- *Technical Reference Options and Adapters, Volume 1*, part number 6137804
- *Technical Reference Options and Adapters, Volume 2*, part number 6137806
- *Technical Reference Personal Computer AT*, part number 6139362

Software Reference

For a listing of the driver, see pages 435 through 443, Section E.5, 'Sun386*i* Parallel Port Driver', in Appendix E of the *Writing Device Drivers* manual, part number 800-1780.

Software Release Levels

As of August 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
Cross Compilers (SunOS 4.x, Sun-3,4)	3.0
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	3.0
SunPHIGS	1.1
Sun58TE	1.0

Unbundled Applications

Product Name	Current Release
SunSimplify	1.1
SunTrac (Sun-2)	1.2
SunTrac (Sun-3,4,386*i*)	1.3
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.

Product Dependency Tables

System Hardware and Operating System Dependencies

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

Effective with this issue of the Hardware/Software release dependency tables, information is presented as follows:

- Sun-4 and SPARCStation1 Product Dependency Tables

  This series of tables lists the Sun-4 and SPARCstation1 product dependencies under the **Sun-4 kernel** (Sun-4 products) and **Sun-4c kernel** (SPARCstation1 product) headings.

- Sun-3 Product Dependency Tables

  This series of tables lists the Sun-3 product dependencies under the **Sun-3 kernel** (Motorola 68020-based products) and **Sun-3x kernel** (Motorola 68030-based products) headings.

- Sun-386*i* Product Dependency Tables

  This series of tables lists the Sun-386*i* product dependencies.

- Sun-2 Product Dependency Tables

  This series of tables includes the Sun-2 product dependencies.

These tables are updated and published in the STB on a quarterly basis.

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

Sun-4 Dependencies

System Hardware and Operating System Dependencies

FEATURE	Sun-4 Kernel		Sun-4c Kernel
	Release 4.0.3	Release 4-3.2	Release 4.0.3
System Hardware Architecture:			
Sun-4/110 and Sun-4/2xx	x	x	
Sun-4/3xx	x		
System Hardware Features:			
ALM2	x	x	
32 MB memory board	x		
900 MB disk	x	x	
327 MB SCSI	x	x	
Double buffering	x		
Operating System Installation:			
Remote tape installation	x	x	
Diskless Sun-3 and Sun-4 installation on a Sun-4 Server*	x	x	
Diskless Sun-2, Sun-3, and Sun-4 installation on Sun-2**, Sun-3, Sun-4 Servers	x		
Sunupgrade from 4.0 to 4.0.3	x		

* Sun-4 servers running Sys4-3.2 can support Sun-3 clients running SunOS Release3.5.

**Sun-2 servers strongly discouraged.

Bug Fixes and Improvements

FEATURE	Sun-4 Kernel		Sun-4c Kernel
	Release 4.0.3	Release 4-3.2	Release 4.0.3
QIC-24 Distribution Media (Sun-3/Sun-4)	x	x	
SunPro `make`	x	x	
`filemerge`	x		
Subnets	x		
SCSI Disconnect/Reconnect	x		
SunOS Rel. 3.3 bug fixes	x		
SunOS Rel. 3.4 kernel bug fixes	x		
SunOS Rel. 3.4 SunView bug fixes	x	x	
SunOS Rel. 3.5 bug fixes	x		

Bundled Software Products

FEATURE	Sun-4 Kernel		Sun-4c Kernel
	Release 4.0.3	Release 4-3.2	Release 4.0.3
SunView Rel. 1.7		x	
SunView Rel. 1.75	x		

Unbundled Software Products

FEATURE	Sun-4 Kernel		Sun-4c Kernel
	Release 4.0.3	Release 4-3.2	Release 4.0.3
NeWS Rel. 1.1	x	x	
NSE Rel. 1.1		x	
Cross-Compilers Rel. 2.0:			
Sun-4 to Sun-2 or Sun-3		x	
Cross-Compilers Rel. 3.0:			
Cross-Compilers 3.0 - C	x		x
Cross-Compilers 3.0 - FORTRAN	x		x
Cross-Compilers 3.0 - Pascal	x		x
Sun FORTRAN Rel. 1.05		x	
Sun FORTRAN Rel. 1.1	x		
Sun FORTRAN Rel. 1.2	x		x
Sun Pascal Rel. 1.05		x	
Sun Pascal Rel. 1.1	x		
SunINGRES Rel.5.1	x		
SunUNIFY Rel. 3.0	x	x	
SunSimplify Rel. 1.1	x	x	
SunGKS Rel. 2.2.1*	x		
SunGKS Rel. 3.0	x		x
Sun58TE Rel. 1.0	x		
SunPHIGS Rel. 1.0*	x		
SunPHIGS Rel. 1.1	x		x
Modula-2 Rel. 2.1	x		
PC-NFS Rel. 3.0	x	x	x
PC-NFS Toolkit		x	
SunIPC Rel. 1.2	x		
Sun Common Lisp Rel. 2.1	x	x	
Sun Common Lisp Rel. 3.0	x	x	
SPE Rel. 1.0 for SCLisp Rel. 2.1	x	x	
TranScript Rel 2.1	x		
SunTrac Rel. 1.3	x		x
DOSWindows Rel. 1.0	x		

* Applications produce graphics only under `sunview(1)` in Rel. 4.0.3. NeWS is not currently supported.

**These releases run under SunOS 4.0.3 in a "compatibility" mode.

SunLink Communications Software Products

FEATURE	Sun-4 Kernel		Sun-4c Kernel
	Release 4.0.3	Release 4-3.2	Release 4.0.3
IBM Connectivity Products:			
BSCRJE Rel. 6.0	x		
SCA Rel. 6.0	x		
Local 3270 Rel. 6.1	x		
SNA 3270 Rel. 5.1		x	
SNA Peer-to-Peer Rel. 6.0	x		
SNA 3270 Rel. 6.1	x		
CG3270 Rel. 6.1	x		
DEC Connectivity Products:			
DNI Release 5.1		x	
TE100 Rel. 5.1		x	
TE100 Rel. 6.0	x		
TE3278	x		
DNI Rel. 6.0	x		
Standards:			
X.25 Rel. 5.1		x	
X.25 Rel. 6.0	x		
Wide Area Networks:			
IR Rel. 6.0	x		
MCP Rel. 6.0	x		

Sun-3 Dependencies

System Hardware and Operating System Dependencies

FEATURE	Sun-3 Kernel					Sun-3x Kernel
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2	Release 4.0.3
System Hardware Architecture:						
Sun-3/60	x	x	BT			
Sun-3/E	x	x	BT			
Sun-3/50, 3/75, 3/1xx, Sun-3/2xx	x	x	x	x	x	
Sun-3/80, Sun-3/4xx					x	
System Hardware Features:						
ALM2	x	x				x
32 MB memory board						x
900 MB disk	x	x				x
327 MB SCSI	x	x				x
Double buffering	x	x				x
Operating System Installation:						
Remote tape installation	x	x	x	x	x	x
Diskless Sun-2 and Sun-3 installation on a Sun-3 Server	x	x	x	x	x	x
Diskless Sun-3 and Sun-4 installation on a Sun-4 Server*	x					x
Diskless Sun-2, Sun-3, and Sun-4 installation on Sun-2**, Sun-3, Sun-4 Servers	x					x
Sunupgrade from 4.0 to 4.0.3	x					

* Sun-4 servers running Sys4-3.2 can support Sun-3 clients running SunOS Release3.5.

**Sun-2 servers strongly discouraged.

Bug Fixes and Improvements

FEATURE	Sun-3 Kernel					Sun-3x Kernel
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2	Release 4.0.3
QIC-24 Distribution Media (Sun-3/Sun-4)	x					
SunPro `make`	x	x	x			
`filemerge`	x	x	x			
Subnets	x	x	x	x		
SCSI Disconnect/Reconnect	x	x	x	x		
SunOS Rel. 3.3 bug fixes	x	x	x	x	x	
SunOS Rel. 3.4 kernel bug fixes	x	x	x			
SunOS Rel. 3.4 SunView bug fixes	x	x	x			
SunOS Rel. 3.5 bug fixes	x	x				

Bundled Software Products

FEATURE	Sun-3 Kernel					Sun-3x Kernel
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2	Release 4.0.3
FORTRAN-77		x	x	x*	x	
pc (Pascal)		x	x	x*	x	
SunView Rel. 1.7		x	x			
SunView Rel. 1.75	x					

* Uses the version for SunOS 3.2

Unbundled Software Products

FEATURE	Sun-3 Kernel					Sun-3x Kernel
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2	Release 4.0.3
NeWS Rel. 1.1	x	x	x	x	x	x
NSE Rel. 1.1		x	x	x		
Cross-Compilers Rel. 2.0:						
Sun-3 to Sun-2 or Sun-4		x	x	x	x	
Cross-Compilers Rel. 3.0:						
Cross-Compilers Rel. 3.0 - C	x					x
Cross-Compilers Rel. 3.0 - FORTRAN	x					x
Cross-Compilers Rel. 3.0 - Pascal	x					x
Sun FORTRAN Rel. 1.0		x	x	x	x	
Sun FORTRAN Rel. 1.1	x					
Sun FORTRAN Rel. 1.2	x					x
Sun Pascal Rel. 1.1	x					x
SunINGRES Rel.5.1	x	x	x	x	x	
SunUNIFY Rel. 3.0	x	x	x	x	x	
SunSimplify Rel. 1.1	x	x	x	x	x	
SunAlis Rel. 2.1		x	x	x	x	
SunGKS Rel. 2.2.1*	x	x	x	x	x	x
SunGKS Rel. 3.0	x					x
Sun58TE Rel. 1.0	x					x
SunPHIGS Rel. 1.0*	x	x				x
SunPHIGS Rel. 1.1	x					x
Modula-2 Rel.2.0		x	x	x	x	
Modula-2 Rel. 2.1	x					x
SunPaint Rel. 1.0**	x	x	x			x
SunWrite Rel. 1.0**	x	x	x			x
SunDraw Rel. 1.0**	x	x	x			x
PC-NFS Rel. 3.0	x	x	x	x	x	
PC-NFS Toolkit		x	x	x	x	
SunIPC Rel. 1.2		x	x	x	x	x
Sun Common Lisp Rel. 2.1	x	x	x	x	x	
Sun Common Lisp Rel. 3.0	x	x	x	x	x	
SPE Rel. 1.0 for SCLisp Rel. 2.1	x	x	x	x	x	
TranScript Rel 2.1	x	x	x	x	x	x
SunTrac Rel. 1.3	x					x
DOSWindows Rel. 1.0	x					x

* Applications produce graphics only under `sunview(1)` in Rel. 4.0.3. NeWS is not currently supported.
**These releases run under SunOS 4.0.3 in a "compatibility" mode.

SunLink Communications Software Products

FEATURE	Sun-3 Kernel					Sun-3x Kernel
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2	Release 4.0.3
IBM Connectivity Products:						
BSC3270 Rel. 3.0		x	x		x	
BSC3270 Rel. 6.1	x					x
BSCRJE Rel. 5.0		x	x		x	
BSCRJE Rel. 6.0	x					x
SCA Rel. 5.0		x	x		x	
SCA Rel. 6.0	x					
Local 3270 Rel. 5.0		x	x		x	
Local 3270 Rel. 6.1	x					
SNA 3270 Rel. 5.0		x	x		x	
SNA Peer-to-Peer Rel. 5.0		x	x		x	
SNA Peer-to-Peer Rel. 6.0	x					x
SNA 3270 Rel. 6.1	x					
CG3270 Rel. 6.1	x					x
DEC Connectivity Products:						
DNI Release 5.0		x	x		x	
TE100 Rel. 4.0		x	x		x	
TE100 Rel. 6.0	x					x
TE3278	x					x
DNI Rel. 6.0	x					x
Standards:						
DDN Rel. 5.0		x	x		x	
OSI Rel. 5.2		x	x		x	
MHS Rel. 5.2		x	x		x	
X.25 Rel. 5.2		x	x		x	
X.25 Rel. 6.0	x					x
Wide Area Networks:						
IR Rel. 5.0		x	x		x	
IR Rel. 6.0	x					x
MCP Rel. 5.0		x	x		x	
MCP Rel. 6.0	x					x*
* Not supported on Sun-3/80.						

Sun386*i* Dependencies

Unbundled Software Products

FEATURE	Release 4.0.2
Sun FORTRAN Rel. 1.1R	x
Sun Pascal Rel. 1.1	x
SunGKS Rel. 3.0	x
SunUNIFY Rel. 3.0	x
SunSimplify Rel. 1.1	x
Modula-2 Rel. 2.1	x
SunTrac Rel. 1.3	x
TranScript Rel 2.1	x
DOS Windows Rel. 1.0	x

SunLink Communications Software Products

FEATURE	Release 4.0.2
IBM Connectivity Products:	
SNA 3270 Rel. 6.1	x
DEC Connectivity Products:	
TE100 Rel. 6.0	x
DNI Rel. 6.0	x
Standards:	
X.25 Rel. 6.0	x
Wide Area Networks:	
IR Rel. 6.0	x

Sun-2 Dependencies

System Hardware and Operating System Dependencies

FEATURE	Sun-2 Kernel				
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2
System Hardware Architecture:					
Sun-2/xxx	x	x	x	x	x
System Hardware Features:					
Double buffering	x	x			
Operating System Installation:					
Remote tape installation	x	x	x	x	
Diskless Sun-2 and Sun-3 installation on a Sun-3 Server	x	x	x	x	
SunUpgrade from 4.0 to 4.0.3	x				

Bug Fixes and Improvements

FEATURE	Sun-2 Kernel				
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2
SunPro `make`	x	x	x		
Subnets	x	x	x	x	
SCSI Disconnect/Reconnect*	x	x	x	x	
SunOS Rel. 3.3 bug fixes	x	x	x	x	
SunOS Rel. 3.4 kernel bug fixes	x	x	x		
SunOS Rel. 3.4 SunView bug fixes	x	x	x		
SunOS Rel. 3.5 bug fixes	x	x			

* No SCSI-3 support on Sun-2 systems.

Bundled Software Products

FEATURE	Sun-2 Kernel				
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2
FORTRAN-77		x	x	x*	x
pc (Pascal)		x	x	x*	x
SunView Rel. 1.5					x
SunView Rel. 1.7		x	x		
SunView Rel. 1.75	x				

* Uses the version for SunOS 3.2

Unbundled Software Products

FEATURE	Sun-2 Kernel				
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2
NeWS Rel. 1.1	x	x	x	x	
NSE Rel. 1.1		x	x	x	
Cross-Compilers Rel. 2.0:					
Sun-2 to Sun-3 or Sun-4		x	x	x	x
Sun FORTRAN Rel. 1.0		x	x	x	x
Sun FORTRAN Rel. 1.1	x				
Sun Pascal Rel. 1.1	x				
SunINGRES Rel.5.1	x	x	x	x	x
SunUNIFY Rel. 3.0	x	x	x	x	x
SunSimplify Rel. 1.1	x	x	x	x	x
SunAlis Rel. 2.1		x	x	x	x
SunGKS Rel. 2.2.1*	x	x	x	x	x
SunPHIGS Rel. 1.0*	x	x			
Modula-2 Rel.2.0		x	x	x	x
SunPaint Rel. 1.0**	x	x	x		
SunWrite Rel. 1.0**	x	x	x		
SunDraw Rel. 1.0**	x	x	x		
SunIPC Rel. 1.2		x	x	x	x
Sun Common Lisp Rel. 2.1	x	x	x	x	x
TranScript Rel 2.1		x	x	x	x
SunTrac Rel. 1.2	x	x	x	x	x
DOS Windows Rel. 1.0	x				

* Applications produce graphics only under `sunview(1)` in Rel. 4.0.3. NeWS is not currently supported.

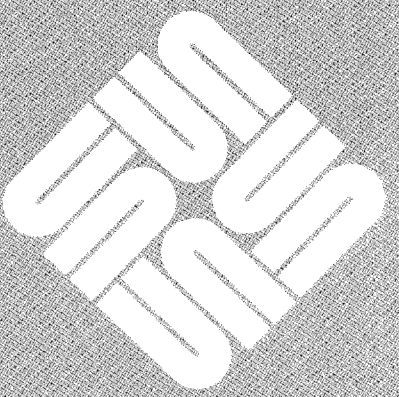
**These releases run under SunOS 4.0.3 in a "compatibility" mode.

SunLink Communications Software Products

FEATURE	Sun-2 Kernel				
	Release 4.0.3	Release 3.5	Release 3.4	Release 3.3	Release 3.2
IBM Connectivity Products:					
BSC3270 Rel. 3.0		x	x		x
BSC3270 Rel. 6.1	x				
BSCRJE Rel. 5.0		x	x		x
BSCRJE Rel. 6.0	x				
Local 3270 Rel. 5.0		x	x		x
SNA Peer-to-Peer Rel. 6.0	x				
CG3270 Rel. 6.1	x				
DEC Connectivity Products:					
DNI Release 5.0		x	x		x
TE100 Rel. 6.0	x				
TE3278	x				
DNI Rel. 6.0	x				
Standards:					
DDN Rel. 5.0		x	x		x
OSI Rel. 5.2		x	x		x
MHS Rel. 5.2		x	x		x
X.25 Rel. 5.2		x	x		x
X.25 Rel. 6.0	x				
Wide Area Networks:					
IR Rel. 5.0		x	x		x
IR Rel. 6.0	x				
MCP Rel. 5.0		x	x		x
MCP Rel. 6.0	x				

8

CUMULATIVE INDEX: 1989



8

CUMULATIVE INDEX: 1989

Index

D

E

K

L

M

N

O

S

W

X

Y

Z

Revision History

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

BULK RATE
U.S. POSTAGE
PAID
Racine, WI
Permit No. 957

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