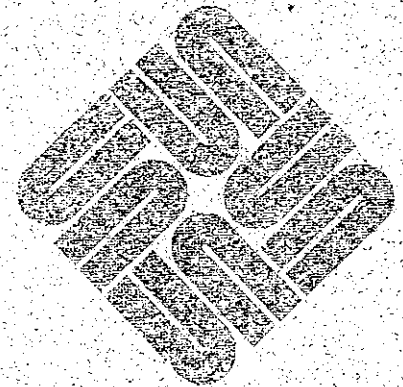




Software Technical Bulletin
July 1987

Software Information Services



Part Number 812-8701-06
Issue 1987-6
July 1987

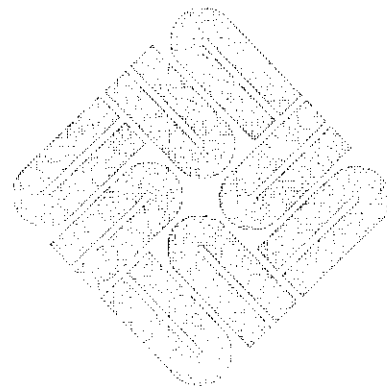




Software Technical Bulletin

July 1987

Software Information Services



Part Number 812-8701-06
Issue 1987 - 6
July 1987

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 2-34, Mountain View, CA 94043 or by electronic mail to *sun!stb-editor*. Customers who have technical questions about topics in the Bulletin should call Sun Customer Software Services AnswerLine at 800 USA-4-SUN.

UNIX, UNIX/32V, UNIX System III, and UNIX System V are trademarks of AT&T Bell Laboratories. DEC, DNA, VAX, VMS, VT100, WPS-PLUS, and Ultrix are registered trademarks of Digital Equipment Corporation. Hayes is a trademark of Hayes Microcomputer Products, Inc. PostScript and TranScript are trademarks of Adobe Systems, Inc. Sun-2, Sun-2/xxx, Sun-3, Deskside, SunStation, SunWorkstation, SunCore, DVMA, SunWindows, NFS, SunUNIFY™, SunView™, SunGKS, SunCGI, SunGuide, SunSimplify, SunLink, Sun Microsystems, and the Sun logo are trademarks of Sun Microsystems, Inc. UNIFY™ is a trademark of Unify Corporation. ENTER, PAINT, ACCELL, and RPT are trademarks of Unify Corporation. SQL™ is a trademark of International Business Machines Corporation. Applix® is a registered trademark of Applix, Inc. SunAlis™ is a trademark of Sun Microsystems, Inc. and is derived from Alis, a product marketed by Applix, Inc. SunINGRES™ is a trademark of Sun Microsystems, Inc. and is derived from INGRES, a product marketed by Relational Technology, Inc.

Copyright © 1987 by Sun Microsystems.

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

Contents

Section 1 NOTES & COMMENTS	239
Editor's Notes	239
Section 2 ARTICLES	243
Read, Write Naming Conventions	243
Back-to-Back Ethernet Packets	245
Disk Labels and Initialization	247
SunAlis Release 2.0 Uses	249
SunINGRES Release 5.0 Uses	254
Section 3 STB SHORT SUBJECTS	263
Quick SCSI Disk Test	263
SunOS Release 3.3 and Subnets	264
Mapping PF and Arrow Keys	265
nd1 Partitions and dumping	266
setup and Partition Sizes	267
NFS and mount Hints	268
sendmail and Aliases	269
rdump and Host Names	270
Incompatible Databases	271
Section 4 IN DEPTH	275
Color Maps	275
Section 5 QUESTIONS, ANSWERS, HINTS, AND TIPS	291
Q&A, and Tip of the Month	291
Errata Corrections	296
Section 6 THE HACKERS' CORNER	299
Determining Memory Size	299

Determining Devices Present	301
An Answermail Script	321

NOTES & COMMENTS

NOTES & COMMENTS	239
Editor's Notes	239



NOTES & COMMENTS

Editor's Notes

Editor's Notes

The July editor's notes for the Software Technical Bulletin (STB) include a survey on sample script and source code.

The Hackers' Corner Survey

You may have noticed past STB articles have included sample script or code to do a task described in the article. This issue contains articles with sample script or code in 'The Hackers' Corner.' Such script or code is offered for your professional interest, not as a supported Sun product. It may not work in all cases, and may not be compatible with future SunOS releases. Note that I refer you in each article to your local script or programming expert for problems or questions.

The Hackers' Corner ?

Do you want to see such script or code in future STB issues?

YES _____

NO _____

Thanks for your input. I will tabulate the results and publish them in an upcoming STB issue.



ARTICLES

ARTICLES	243
Read, Write Naming Conventions	243
Back-to-Back Ethernet Packets	245
Disk Labels and Initialization	247
SunAlis Release 2.0 Uses	249
SunINGRES Release 5.0 Uses	254



ARTICLES

Read, Write Naming Conventions

Read, Write, and Transfer Naming Conventions

This article contains a clarification of naming conventions for `read` and `write` operations between two, or among three or more agencies. Use the conventions in this article to avoid confusion about the direction of read, write, and transfer operations.

The naming conventions for these operations are slightly different, depending on the overall system configuration. One set of conventions apply at the system level in a computing system having a single Central Processing Unit (CPU). Another set applies in the rare case of a multi-node matrix in a system without a single, central controller. Such multi-node matrix arrangements have the processing load dynamically shared among several CPUs.

Single-CPU Naming Conventions

Naming conventions at the system level are straightforward when the computing system has a single CPU. In this case 'read' and 'write' are used with respect to the CPU. Read operations result in data transfers from a peripheral, for example, toward the CPU registers. Write operations result in data transfers from the CPU registers toward the peripheral device in this case.

Variations in data transfers follow these rules. For example, a controller card may be sending data toward the CPU. This operation is considered a 'read' in that the data is transferred toward the CPU. It is not considered a 'write' with respect to the controller card since it is only responding to an instruction overtly caused by the CPU. The instruction is to send data to the CPU so it can read the data.

This principle applies within hardware associated directly with the CPU. For example, data transferred from the main memory associated with the CPU to a CPU register is considered a 'read' operation. Again, this is with respect to the CPU register.

Conversely, 'write' operations are described with respect to the CPU. Write operations result in data transfers from CPU registers out to other hardware

devices. Data transfers from CPU registers outward to the main memory associated with the CPU are then write operations. Data transfers from the CPU toward peripheral device controller cards are write operations with respect to the CPU. This operation is not considered a 'read' with respect to the controller card since it is only responding to an instruction overtly caused by the CPU. The instruction in this case is to receive data from the CPU so it can write the data.

The Problem with Multiple CPUs

For the purposes of this discussion, a node is considered to be an addressable piece of hardware. Confusion in naming conventions occurs when one agency is reading, another is writing, and a third may be supervising the data transfer. Ambiguity arises when several nodes contain CPU hardware and each initiates data transfers.

The Multi-Node Naming Convention Solution

The use of 'read' and 'write' depends on where the data transfer is *initiated*.

If node[j] initiates the transfer, and node [j+1] simply receives the data, it is a write operation with respect to node[j].

If node[j+1] initiates the data transfer, and node[j] simply supplies the data, it is a read operation with respect to node[j+1].

If a third agency is involved, the operation is probably a 'transfer.' In this case the data may be sent in either direction or may change direction *en route*. However, data transfers into memory may be described as 'writes' with respect to the sending agency. Transfers from memory may then be described as 'reads' with respect to the receiving agency.

Please note that read, write, and transfer operations are not described with respect to the memory, presumed to be passive for practical purposes in these cases.

Back-to-Back Ethernet Packets

Back-to-Back Ethernet Packets

'Back-to-back' Ethernet packets are sent over the network to minimize dead-air time on the cable. This packet processing does not involve any actual negotiation, except in the case of a collision. Ignoring input issues, the logic involved is shown below.

- 1 Is an output packet command available?
If no,
go to 1.
- 2 Is an ether available, no incoming packet in progress?
If no,
go to 2.
- 3 Start packet transmission and listen for a collision-detect.
- 4 Is a collision detected?
If yes,
jam the Ethernet to force a collision-detect on all nodes,
run the backoff timer,
go to 2.
- 5 Go to 1.

Software can queue several packet transmit commands to the controller chip. The packets so transmitted in steps 1 - 3 above are within the Ethernet specification. The time interval between packets can be as little as a few hundred nanoseconds at a typical 10 MHz chip clock rate.

Back-to-back packet transmission, though within the Ethernet specification on the transmit-side, can cause problems on the receive-side when the hardware buffer size is too small or controller processing speed too slow. For example, a typical Network File System (NFS) response is frequently a file system block, 8 Kb plus protocol, which becomes six Ethernet packets. This can make back-to-back packet processing common on Sun networks. Such packets may originate from the same or from two different nodes, all transparent to network users.

Problems arise on the receive-side when Ethernet controller implementations cannot process back-to-back packets without eventually dropping one. This occasionally occurs due to a slow controller. More often it is due to inadequate buffer size to store the next packet. When a packet is dropped, a protocol layer then has to timeout and retransmit.

For example, an Ethernet controller could have buffer space sufficient for two incoming packets only. A third packet is then dropped if it arrives before the first buffer has been emptied and released. In the case of NFS servers transmitting six consecutive packets and recovery occurring at the NFS level, all six packets have to be resent. This has resulted in new mount options to reduce the transmission rate in both directions.

Sun hardware includes sufficient buffer space to make this problem unlikely. Bridge vendors, however, can increase the chances of running into this problem if either buffer memory or processing power for the retransmission are inadequate. For example, 'throughput' rates of 4-6 Mbits/second across the bridge are typical. However, groups of six back-to-back, mostly-full packets represent a peak data rate approaching 10 Mbits/second.

Disk Labels and Initialization

Disk Labeling and Initialization

This article contains information on disk labeling and partitioning. Disk users need to take care to not overwrite the disk label during usage so that the workstation will again recognize the disk upon reboot.

At boot time, the disk device driver initializes its disk- and partition-parameters. It reads the disk label residing at cylinder 0, track 0, sector 0 of the disk. The system does not recognize the disk if there is no disk label at boot time.

Disk Label Questions

Sun customers who need to format disks and set up partitions on disks have asked the following questions.

- If there is a 'normal', non-swap first partition for the label on the disk, how does `mkfs` or the system know not to access that sector?
- Should I start the partition I want to set up at the second sector?

Protecting the Disk Label: Answers

Unix file systems do not use the first 16 sectors of the partition in which they are placed. The first sector is reserved for a label, even though the label actually appears only in partitions starting at cylinder 0. The next 15 sectors are reserved for a boot track, which may be present in any partition.

- `mkfs` and `newfs` follow the convention of not using the first 16 sectors. So 'normal' partitions are safe. However, raw partitions used by unconventionally coded programs and swap partitions are not safe.
- You do not need to begin your partition on the second sector, provided you use conventionally coded programs and Unix file system calls.

A partition starts on the cylinder boundary, and extends exactly from the beginning of the cylinder to the partition end-point, wherever that may be. There is no special protection at the 'partition level' for the disk label. Note that partitions may also overlap. Any partition starting at cylinder 0 on a disk is a *potential hazard* to the disk label.

A disk label may be located in sector 0. Note whether the software which uses a particular partition is the file system or one of its associated utilities, and both do not write into sector 0. Your disk label is then safe. Then note whether the software which uses this particular partition is, for example, one of the kernel swapping routines. You are the guaranteed to destroy your disk label.

Disk Label Precautions

You may wish to take two precautions to avoid destroying your disk labels when you write software which opens a raw disk partition and then writes on it.

- Ensure that the partition does not start at cylinder zero. This is difficult and may not be possible to do within the software. You will need to ensure that all programmers who might repartition your disks know about this requirement.

- Assume that any partition may start at cylinder 0. You then need to adopt the convention of skipping sector 0 of the partitions your programmers open.

 SunAlis Release 2.0 Uses 

SunAlis Release 2.0

This article is a brief overview of SunAlis release 2.0, including performance enhancements, new features and functions, installation, and use considerations.

Introduction

SunAlis is a fully integrated office automation software system that combines applications and tools such as a document composer, graphics editors, spreadsheet, and electronic mail and message facilities.

SunAlis release 2.0 is intended for use with Sun Microsystems Operating System (SunOS) release 3.0 and subsequent releases. Note that this release will not run on Sun 100U workstations.

Performance Enhancements

The most significant improvements of SunAlis 2.0 are product reliability and performance. Performance improvements have been made in the areas of locking and batching, database buffering, and dumb terminal enhancements. Menu-based applications have been streamlined by combining them to improve response time, and by the availability of the new `prompt` mode feature to bypass menus. Local printing optimizations have been modified to be transparent to the user.

New Features and Functions

The following new features and enhancements are provided in SunAlis release 2.0.

- UNIX access for dumb terminals or non-window operation
- Document composer enhancements
- Laserwriter landscape mode
- Enhanced Mail Service capabilities
- Menu enhancements
- Printing full column and row headings with spreadsheet data
- Bulletin Board, a publicly-shared cabinet

Some highlights of these features are discussed in the following paragraphs.

Document Composer Enhancements

The Document Composer facility has been enhanced to improve document production in several ways. For example, the user is now notified if he or she attempts to create a document having the same name as an existing document, and the document listing display now includes document size information in kilobytes. Users can now add or delete entries to a user-defined spelling dictionary. Both the Document Composer and Graphics text now provide support for mathematic, scientific, and international language characters.

Mail Service

The Mail Service function now allows users to specify a sender name on all mail messages; the SunAlis system-supplied name is retained. This user-specified field is introduced with `From`. Mail Service has also been enhanced to allow the user to affix up to 255 characters of additional comments to a forwarded message.

Menu Enhancements

In addition to the new `prompt` mode feature, the menus have been enhanced as follows.

- A new facility, `Log-out`, is now available via the interrupt menu.
- A new main menu option, `Other Functions`, is now provided to enable the user to escape to the UNIX shell.
- When a user attempts to delete an object from a menu that is not retrievable from the SunAlis wastebasket, the prompt `Are you sure?` appears to confirm the delete request. The user can either proceed or cancel, as desired.

Bulletin Board

A shared cabinet facility called `Bulletin Board` can now be established for SunAlis users. This cabinet can have up to ten owners who can check documents in and out. All users can access the documents on a read-only basis. The `Bulletin Board` cabinet is uniquely named as `'Bulletin Board.'`

Installation and Use Considerations

The following are highlighted installation and use considerations for SunAlis Release 2.0. Additional information can be found in the *Read This First Notes for SunAlis Release 2.0*, part number 800-1417. Refer to the *Read This First* as well as the *SunAlis Administrator's Guide*, part number 800-1669, for further details prior to software installation.

alisverify

When installing SunAlis, you must have write access to the directory you are in when you run the `alisverify` system verification routine; if not, `alisverify` will abort. Additionally, the current system name should exist in the `/.rhosts` file. If not, the error `no entry in /.rhosts for <current system name>` occurs when running `alisverify`. This message can be ignored.

SunAlis System Space Requirements

SunAlis has the following system space requirements.

SIZE

16.7 Mb	
13.6 Mb	Executables
.6 Mb	Model File System
2.5 Mb	On-Line Help System

SWAP SPACE

10-14 Mb, plus 1 Mb per user
(More swap space required if more windows are opened)

FILE CABINET

10 Mb per user
(.7 Mb overhead)

MEMORY

4 Mb
(minimum)

Mail System Gateway Registries

When using SunAlis to send mail to and from SunAlis users and UNIX users or both, up to three different gateway registries must be created, depending on the system.

SunAlis Interdomain Gateways are created and used to send and receive between SunAlis domains through an Ethernet connection. When creating these gateway registries, the only information required is the internet address of the receiving host.

SunAlis to UNIX to SunAlis Gateways are used to send and receive mail through UNIX Bridge Gateways. When creating the gateway registry for this type of mail, name the gateway program `gtwy2`. To allow SunAlis to retrieve SunAlis mail from a user's UNIX mail box, the process `gtwy3` must be running. To enable `gtwy3`, edit the file `accaps` to change the third field in the `gtwy3` entry from `n` to `y`, as shown below. This allows the `gtwy3` process to be brought up each time `aop` is run.

accaps file before editing:

```

.
.
.
gecvt:139:n:-bc::::
gtwy1:140:n:::::
gtwy2:141:n:::::
gtwy3:142:y:::::
pdbcvt:143:n:-o::::
.
.
.

```

accaps file after editing:

```

.
.
.
gecvt:139:n:-bc::::
gtwy1:140:n:::::
gtwy2:141:n:::::
gtwy3:142:y:::::
pdbcvt:143:n:-o::::
.
.
.

```

SunAlis to UNIX Gateways are used to send one-way mail from SunAlis to UNIX users. When creating the gateway registry for this type of mail, name the gateway program `gtwy1`.

SunAlis and Sun Windows

When several SunAlis windows are opened simultaneously outside Sun Windows, closing the SunAlis windows using the `click close` option destroys the window displays. To correct the displays, select `set-aside`, then reopen the windows. This can also be used when the SunAlis window does not get proportionately resized as a result of resizing SunTool windows.

The `screenblank` function blanks out the screen when running SunAlis outside of Sun Windows. To retrieve the screen display, move the mouse around on the mouse tablet. If you are using SunAlis from a dumb terminal, use `rlogin` to log into the host from another terminal, then use `kill` to kill the `screenblank` process.



Document Composer

Large document manipulation requires that at least the same amount of free space be left in the SunAlis file system as the size of the document being manipulated. Performance can be improved by limiting document size, or by dividing a large document into several smaller sections, as practical.

Two methods can be used to print a document with headers and footers on a document that resides in a shared cabinet: the user can either transfer the document to a personal cabinet before printing, or the user can specify two or more documents to be printed at one time.

SunINGRES Release 5.0 Uses

SunINGRES Release 5.0

This article is a brief overview of SunINGRES release 5.0, including areas of significant performance enhancement, new features and functions, installation, and upgrade considerations.

Introduction

SunINGRES is a Sun port of the INGRES product from Relational Technology, Incorporated, of Alameda, California. SunINGRES is a complete relational database which provides the features listed below.

- Menu-based access to the database via SunINGRES/MENU
- The Query-By-Forms (QBF) forms-based query facility
- The Report-By-Forms (RBF) report writer facility
- Database access through either C- or FORTRAN-based programs via embedded QUEL/C, embedded QUEL/FORTRAN, embedded SQL/C, and embedded SQL/FORTRAN
- The Application-By-Forms (ABF) forms-based application development subsystem, which uses Operation Specification Language (OSL), a fourth-generation language

SunINGRES release 5.0 is intended for use with Sun Microsystems Operating System (SunOS) release 3.0 and subsequent releases, and requires about 23 Mb of space on the database server for the executables, plus additional space for each database.

Performance Enhancements

Enhancements made to SunINGRES internals have resulted in performance improvements ranging from 80% up to 100% in short query benchmarks, such as TPI, to more than twice the performance improvement in aggregate processing and multi- field joins. Performance improvements result from changes made in the query optimizer, query processing overhead, and locking. The performance of applications making use of `btrees` will benefit from additional, major improvements to this storage structure.

Reduced query processing overhead has been achieved through the following two changes. First, a new buffer management system implements a local buffer manager as a module. The module is used by SunINGRES access methods to manipulate pages. Second, a local buffer cache manager is used to manage local sets of pages (buffers) at a rate of one set per user. The local buffer manager provides larger buffer pools and gives preference to system catalogs and index pages. The user can set the size of the buffer. Additional improvements to

frontend and backend communications, and to the interpreter have also resulted in reduced CPU overhead.

Optimizer Performance Improvements

The following query optimizer elements have been enhanced to improve performance.

- Multi-field keys
- btrees
- Lookups in zopt catalogs
- Unique keys
- Aggregate queries
- Single variable queries
- Index data
- Joins on functions and different datatypes
- optimizedb support for date and money via set commands

New and Changed Features and Functions

Highlights of new and changed features and functions affecting the Forms System, Query-By-Forms (QBF), the new SET CACHE feature, Applications-By-Forms/Operations Specification Language (ABF/OSL), and EQUQL/ESQL are included below. For additional information, refer to *SunINGRES Release 5.0 Release Notes*, part number 800-1906, and *SunINGRES Release 5.0 Read This First*, part number 800-1999.

Forms System Enhancements

Highlights of changes to the SunINGRES Forms System are as follows.

- Program Function (PF) keys can now be used.
- The HELP operation has been extended to present a submenu of operations for various types of help the user may desire. The text can be scrolled and organized in a hierarchical fashion.
- Menu items of the same name will now have the same meaning throughout the menu system. Most menu items will have unique prefixes.
- All error messages now request that the user strike the <RETURN> key as an acknowledgement.

- VIFRED (VISual FoRms EDitor) and RBF have been modified to display the catalogs frame listing existing frames or reports. Users can then proceed to the layout frame to design a new object, or edit an existing object.
- The dollar signs appearing at the end of objects being moved in VIFRED and RBF have been removed. The object being moved now blinks when the move command is in effect.

Query-By-Forms

Significant enhancements and changes to QBF (Query-By-Forms) in release 5.0 are as follows.

When creating a default form, the user can now choose to use a `tablefield` or `simple field`. When working with forms, note that the `delete` operation within `update` mode has been changed to display a submenu of choices.

Query mode has been modified to include any combination of 'ands' and 'ors' to selection criteria. Upon entering the last character in a field, the first character scrolls to the left, out of the screen display. When the user tabs out of the field, the character reappears. This occurs in the QBF `retrieve` and `update` modes. When specifying queries, values can be entered in a field by left- or right-scrolling or both of the individual fields. Sorts made from the query form can be made on any combination of keys, in ascending or descending order.

The SET CACHE Command

Release 5.0 includes a new local cache, used with the new `SET CACHE` command to increase the size of local caches. Because SunINGRES manages each user's local cache with locks, and each page of local cache consumes one lock, consideration should be given to the effect of increasing the size of local caches prior to using the `SET CACHE` command. Refer to the *SunINGRES/QUEL Release 5.0 Reference Manual* portion of the *SunINGRES Manual Set*, part number 800-1644, for complete information on the `SET CACHE` command.

Applications-By-Forms/Operations System Language (ABF/OSL)

Highlights of the ABF/OSL features and functions which have been changed or added are discussed below.

In SunINGRES release 3.0, program statements inside an `UNLOADTABLE` loop could only refer to `tablefield` data copied into hidden fields. If an explicit reference was made to a `table.column`, it was interpreted by OSL to mean 'for the row the cursor was on before the `UNLOADTABLE` began.'

In SunINGRES release 5.0, however, the data need not be copied into hidden fields for use inside the `UNLOADTABLE` loop, and an explicit reference to a `table.column` is interpreted by OSL to mean 'for the row currently being unloaded.' Refer to the `UNLOADTABLE` section in Appendix A.2 of the

SunINGRES/Applications: Applications-By-Forms (SQL) User's Guide portion of the *SunINGRES Manual Set*, part number 800-1664.

All `EQUEL` procedures called from `OSL` must now be defined explicitly as procedures to `ABF`. A source file for this procedure must also be specified, although that file can be empty. For example, assume a user is calling a procedure named 'checkdate,' which is an object library listed in the linker options file, pointed to by the `SunINGRES` name `ING_ABFOPT1`. The user defines a procedure to `ABF` named 'checkdate' and specifies an existing (but empty) source file. The application will link correctly, even though the source code for the procedure is not in the application's source code directory.

In `SunINGRES` release 5.0, expressions can now be used where previously only constants were allowed; for example, expressions are now valid in messages, subsystem calls, prompts, and so forth. In the `QUEL` or `SQL` part of the `OSL` language, a string constant, id, or field name can now be used in place of a name. The name of the field is preceded by a colon (:), as in '`fieldname`.' Note that the structure is fixed, and cannot be built dynamically, as in an `EQUEL` param statement.

When passing arguments, the default is by value. The user can optionally pass arguments by reference. Arguments can be passed to frames, host language procedures, and `OSL` procedures. This allows the user to return more than a single value to the calling frame.

New `OSL` Keyword `byref`

The default method of passing string variables from `OSL` to `EQUEL/C` procedures is by value. Any changes made to such variables in the `EQUEL/C` procedure is not reflected in the calling frame. In order to pass an `OSL` variable by reference to a procedure written in `EQUEL/C` or any other `EQUEL` language, the procedure call must include the new `OSL` keyword `byref`. Refer to Appendix C.3 of the *SunINGRES/Applications: Applications-By-Forms (SQL) User's Guide* portion of the *SunINGRES Manual Set*, part number 800-1664.

`EQUEL`, `ESQL`, and the UNIX Allocator

In `SunINGRES` release 3.0, the storage allocator in `compatlib` used by `EQUEL` and `ESQL` programs conflicted with the `malloc/free` allocator in UNIX `libc.a`. The different allocators would hand the same piece of memory to different clients, thus causing subtle, hard-to-locate problems.

To solve this problem, release 5.0 now provides versions of `malloc(3)`, `calloc(3)`, `realloc(3)`, `free(3)`, and `cfree(3)`, written in terms of the `SunINGRES` allocator. On BSD systems, `valloc()` and `memalign()` are also provided. These routines behave according to the specification in both the UNIX manual and the System V Interface definition, but may not duplicate undocumented behavior of the `libc` allocator.

The return value of `malloc(0)` is not defined. The UNIX `libc.a` `malloc` returns a pointer to 'somewhere', and the `malloc(3X)` version returns NULL. The user should not be relying on either behavior. The current `compatlib` returns a pointer to a non-zero length object.

Using UNIX Signals in an EQUEL or ESQLE Program

The user can catch any UNIX signal, then use it to alter the control flow of his or her program. If this altered control flow also affects SunINGRES, the proper EQUEL or ESQLE instructions must be issued to abort the current SunINGRES operation.

For example, if a program is in a retrieve loop, and it should break out of the loop upon receipt of a signal, then continue with other operations, the command line `## endretrieve` must be issued after the signal is received and before continuing the program. This notifies the SunINGRES backend (the process performing the actual DBMS operations) that the retrieve has been aborted, and prevents results from being returned for the aborted query.

For additional information regarding the use of UNIX signals, refer to the *SunINGRES 5.0 Release Notes*, part number 800-1906.

Installation and Upgrade Considerations

The following are installation and upgrade considerations for SunINGRES release 5.0. Additional information can be found in the *Read This First--Notes for SunINGRES Release 5.0*, part number 800-1999; *SunINGRES 5.0 Release Notes*, part number 800-1906; and Chapter 2 of the *SunINGRES Installation Guide*, part number 800-1681.

Installation

Most of the steps involved in the SunINGRES installation procedure have been automated by a new installation shell script. Refer to Chapters 2 and 3 of the *SunINGRES Installation Guide*, part number 800-1681, for complete instructions on how to read the installation tape and run the installation script.

Due to fixes made in release 5.0, users who have `btree` primary storage structures or `btree` secondary indexes should re-modify them to `btree` after installing release 5.0.

Upgrading from SunINGRES Release 3.0 to 5.0

When upgrading your system from release 3.0 to 5.0, keep in mind the following.

- Release 3.0 databases need not be converted to run under release 5.0.
- The SunINGRES lock manager must be re-installed. For details, refer to Chapter 3 of the *SunINGRES Installation Guide*, part number 800-1681.
- Due to a change in SunINGRES internal lock naming conventions, do not attempt to run release 5.0 simultaneously with release 3.0 on the

same installation. Release 5.0 can run on a release 3.0 database, as long as no release 3.0 user is simultaneously working in the same installation.

- The `II_INSTALLATION` variable must be set for each installation running on a given system. `II_INSTALLATION` must be a unique two-character code for each installation, and is set during the installation script procedure.
- Release 3.0 EQUERL and ABF applications do not need to be recompiled or relinked when release 5.0 is installed.

SunINGRES System Space Requirements

SunINGRES has the following system space requirements.

SIZE

23Mb

Executables, plus size of database

MEMORY

4 Mb

(minimum)



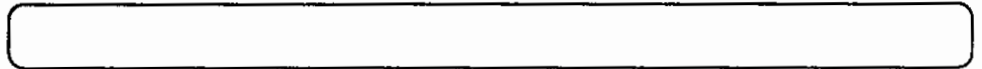
STB SHORT SUBJECTS

STB SHORT SUBJECTS	263
Quick SCSI Disk Test	263
SunOS Release 3.3 and Subnets	264
Mapping PF and Arrow Keys	265
nd1 Partitions and dumping	266
setup and Partition Sizes	267
NFS and mount Hints	268
sendmail and Aliases	269
rdump and Host Names	270
Incompatible Databases	271



STB SHORT SUBJECTS

Quick SCSI Disk Test



Quick SCSI Disk Test

If your Small Computing System Interface (SCSI) disk appears to be responding slowly and you suspect a disk malfunction, this article contains a quick check you can perform without having to halt UNIX.

Use the `dd(1)` command invoked from the shell as shown below to test your SCSI disk for correct reading speed. The `dd(1)` command reads a one-megabyte file. See the `time(1)` and `dd(1)` manual pages in the *Commands Reference Manual*, part number 800-1295, for details.

This test should take about six seconds on a Sun-2 workstation, or about four seconds on a Sun-3/100 series workstation. If it takes appreciably longer, you may have a problem with your disk or disk controller. Of course, your machine should not be too busy when this test is performed.

You need to have read permission on the raw disk device to do this test. Usually only `root` has read permission. Note that the command you enter is shown below in **bold**.

```
# time dd if=/dev/rsd0a of=/dev/null bs=16b count=128
128+0 records in
128+0 records out
0.0u 0.4s 0:04 9% 1+1k 1+3io 0pf+0w
#
```

SunOS Release 3.3 and Subnets



SunOS Release 3.3 and Subnets

Several Sun customers have called about enabling subnets while running SunOS release 3.3. The customers expect their broadcast messages to be 'subnetted broadcast' messages. One example is expecting to broadcast to 100.50.5.0 instead of 100.0.0.0. However, the customers are finding many 100.0.0.0-style broadcasts going out over the wire.

The reason for 'subnetted broadcast' messages not appearing as expected is that some of the programs that process broadcasting (ypbind, rup, umount, rwho, and the like) are not included on the SunOS release 3.3 tape. Thus, the `ifconfig` option may be present to place the proper subnet address into the kernel. However, the SunOS release 3.2 programs ignore the kernel value and always broadcast to net.0.0. New programs that broadcast using the kernel value are included in SunOS release 3.4.

SunOS release 3.3 includes `in.routed` broadcasts.

Please note that `rwhod` does not read the broadcast address from the kernel when running SunOS release 3.4. We are looking into this capability for inclusion in SunOS at a future time.

Mapping PF and Arrow Keys

Mapping PF and Arrow Keys

Problems arise when Sun2/100U customers using VT100-style keyboards try to change a key function. VT100-style keyboards share this problem with Sun2 and Sun3 keyboards: they map to something that you cannot specify in `ttyswrc`.

In this case, customers need to use `KIOCSETKEY` to change the key to something you can specify. See *kbd(5)* for information on keys PF1-PF4. Under the VT100-style keyboard table you will see, in part, the following code.

```

/* Unshifted keyboard table for "VT100 style" */

static struct keymap keytab_vt_lc = {
/* 0 */ HOLE,   BUCKYBITS+SYSTEMBIT,
                HOLE,   HOLE,   HOLE,   HOLE,   HOLE,   HOLE,
/* 8 */ HOLE,   HOLE,   STRING+UPARROW,
                STRING+DOWNARROW,
                STRING+LEFTARROW,
                STRING+RIGHTARROW,
                HOLE,   TF(1),
/* 16 */ TF(2), TF(3), TF(4), c('[', '1', '2', '3', '4'

```

PF1-PF4 are treated as TF1-TF4 (TF signifying Top Function, as opposed to Right, Left, and Bottom). So they are the equivalent to F1-F4 on your Sun-3 keyboard.

Please note that F1 is the SunView capslock key.

nd1 Partitions and dumping

Using nd1 Partitions in
Dump Procedures

When dumping a set of clients, dump the actual nd1 partitions. This is preferred to dumping nd partitions because of how the nd partitions are handled by dump. For example, assume a set of nd partitions exists in the same disk partition. nd essentially creates a new set of mass storage devices that 'secretly' exist on the disk. When dump is used to dump the disk partition, dump opens the partition, locates the superblock, and then dumps the file system. The superblock located by dump is the superblock for the first nd partition. dump will dump this partition and then stop. If the first nd partition is a swap partition, dump fails without dumping any partition.

Dumping the nd1 Partitions
Separately

The correct way to dump a set of clients is to dump the nd1 partitions separately. Essentially, each nd1 partition acts as a complete and separate disk partition. When you need to restore the nd1 partitions, run mkfs first to make a file system, and then run restore -r.

Using nd1 Partitions with
newfs

nd1 partitions differ from other disk partitions in that nd1 partitions do not have labels, thus the newfs utility cannot be used. When newfs is used, it first reads a partition label, makes a file system to fill the partition, and then writes a boot block onto track 0 of the partition. Because the nd1 partitions do not have labels, newfs fails.

It is possible to mimic the newfs functions for nd1 partitions by performing the following steps.

1. Read np.local.
2. Run mkfs to make a file system of the appropriate size on the nd1 device.
3. The usual case is to boot from /pub. To boot directly from the client partition, additionally run installboot.

setup and Partition Sizes

Determining Actual Partition Size with `/etc/mkfs`

When partitioning disks, `setup` appears to return the wrong figures for disk sizes. The figures displayed are consistently lower than they should be, and do not match the formatted size returned by the `diag` and `dkinfo` utilities.

There are several explanations for this size difference. For example, due to file system overhead (which includes super-block, inode list, and free list), the file system size displayed by `setup` is actually the remaining amount of usable space that is available to the user. For further information on file system layout, refer to Appendix A, File System Check Program, in the *System Administration for the Sun Workstation* manual, part number 800-1323.

Another possibility is that most utilities appear to occupy about 10% more disk space than is actually used. This is because the last 10% of the disk is reserved for use only by the superuser when in `setuid` or `root` or both. The extra space allows `root` room to maneuver on the disk. Because of this, a file system can appear to occupy 111% of the disk using `setup`.

It is also possible that the figures displayed by `setup` include the amount of space lost when `newfs` is run.

How to Verify Actual Partition Size

The actual partition size can be verified by entering the size of the partition in question as values to `/etc/mkfs`, as in the example below.

```
mkfs /dev/null (partition size) 67 20 8192 1024 16 10 60
```

This returns the partition size and the superblock backups for an Eagle XP (M2361), and will not write to disk. This is also a useful way to locate alternate superblocks.

NFS and mount Hints



NFS and mount

In a typical Sun network, several machines can be interdependent for services such as YP, mail, and NFS. Many systems utilize an older machine for mail, print, or another dedicated service typically short on disk space requirements, such as a Sun 2/120 with 70Mb disk capacity. In situations where the NFS server goes down (for example, if another machine had been hard mounted to write some files), mail eventually stops, and returns the following error message shortly after login.

```
NFS server not responding ...
```

This error message also appears if the NFS server goes down, and the other machines have no way to continue. The problem is that the shell tries to build a hash table of commands based on the root login path. If any NFS directories are encountered in the root login path, or if any NFS mount points exist in the directories in the path, the system hangs for hard mounts and times out for soft mounts. Since the `mount(8)` command is now in the `rc.boot` file, the system cannot boot with the `-b` switch to avoid the NFS problem.

Helpful Hints

The best preventive measure is to keep the root login path free of NFS directories. If this is not possible, the following hints will help avoid problems.

- Mount partitions with the `bg` (background) option. This allows boot to continue while waiting for the mount to succeed or fail.
- Use the `intr` (interrupt) option when mounting partitions, to allow an operation on a hard partition to be killed.
- Move the `mount -vat nfs` to the end of the `rc.local` file, or after `nfsd 4`. This helps alleviate problems with machine interdependencies.

sendmail and Aliases

sendmail and Postmaster Aliases

When a flag for a particular mailer is set, `sendmail` maps the recipient name to lower case for that mailer. This flag is set for the 'local delivery' mailer, so that mail directed to 'Postmaster', 'postmaster', 'POSTMASTER', and so on is sent to 'postmaster'.

When `sendmail` reads the `/usr/lib/aliases` file, it maps the left side of the alias definition to lower case, as shown below.

Alias definition as entered in the `/usr/lib/aliases` file:

```
Postmaster:      user@sun
```

Alias definition as mapped by `sendmail`:

```
postmaster:     user@sun
```

Names on the right side of the alias definition, however, are not mapped to lower case; neither at the time the `/usr/lib/aliases` file is read, nor when testing to see whether any names to which an alias expands are themselves aliases. `sendmail` cannot tell which mailer will deliver mail to a particular name until `sendmail` has checked the name against its rewriting rules. This prevents mail from being delivered by a mailer that is case-specific, in which case the name mapping would be incorrect.

For example, consider the following alias definition.

```
MAILER-DAEMON:  Postmaster
```

When mail is sent to 'MAILER-DAEMON', `sendmail` first translates 'MAILER-DAEMON' to 'mailer-daemon'. Thus, the translated alias definition appears as follows.

```
mailer-daemon:  Postmaster
```

When `sendmail` locates the alias definition, it expands 'MAILER-DAEMON' to 'Postmaster'. `sendmail` then looks up 'Postmaster', not 'postmaster'. The problem is that the alias definition shown in 'Postmaster: user@sun' will have been translated to lower case, as in 'postmaster: user@sun'. This translated alias definition will not be able to expand 'Postmaster'.

The Workarounds

Currently, the way to ensure problem-free alias definitions is to specify all local addresses appearing at the right side of an alias definition in lower case. A further preventive measure is to never specify 'Postmaster' to the right of a colon in the `/usr/lib/aliases` file.

rdump and Host Names



Using rdump with Host Names Containing Periods

When `rdump` is used to perform a remote dump, it essentially performs the equivalent of `rsh` on a local machine. `rdump` does a call to the `rcmd` library routine to start up the `/etc/rmt` daemon on the remote machine.

The `rcmd` library routine can be used only if the user can remotely log in as a user without a password. `dump`, by default, runs `/etc/rmt` with the same user name as the user's name on the local machine. An `rdump` therefore cannot be performed successfully unless the user can either `rlogin` as root on the remote machine without a password, or if another userid is specified with `dump`.

Since it is not always possible to remotely log in as root without a password, the usual approach is to specify another userid with `dump`, as shown in the example below.

```
dump 0f <host>.<user>:<tape> . . .
```

Note that a period is used to separate the host and user names. Because of this syntax, problems result when using `rdump` to perform a remote dump with host names containing periods, as in the example below.

```
rdump 9ufdsb sun.training.ctr:/dev/rst0 1600 2200 10 /dev/rsd0g
```

When `rdump` is run, the following error message is returned.

```
sun.training: unknown host
```

At first glance, it appears that the last element of the host name is being deliberately stripped off. What actually happens, however, is that the use of periods in the host name conflicts with the use of the period as a domain component separator in the command syntax.

Two Workarounds


Two workarounds can be used when dealing with host names containing periods. One approach is to put another name in the `hosts` file which does not include the periods, as in the example below.

```
192.9.200.xxx sun.training.ctr suntrainingctr
```


The name not containing periods can then be used as the name for the dump host.

Another workaround is to enter either root or the user name of the person performing the dump at the end of the remote host name. This entry is then preceded by a period, as in the example below.

```
rdump 9ufdsb sun.training.ctr.root:/dev/rst0 1600 2200 10 /dev/rsd0g
```

Incompatible Databases



Incompatibility: SunUNIFY, SunSimplify, and ACCELL

SunUNIFY and SunSimplify users should be aware that these products and their associated databases are incompatible with releases of Unify Corporation's ACCELL™ product and its associated databases, and is therefore an unsupported product combination.

When used with ACCELL, the SunUNIFY or SunSimplify databases might become corrupted. The corrupted databases must then be restored from a backup. Additionally, users should not run SunUNIFY or SunSimplify on ACCELL databases.

As a workaround, SunUNIFY users can convert their databases to ACCELL databases. Once converted, ACCELL can be used safely.



IN DEPTH

IN DEPTH 275

Color Maps 275



Color Maps

Sun Color Applications

In this in-depth article, we discuss the use of color on the Sun Workstation.

- Overview to Color
- Hardware Frame Buffers
- Using `suntools`
- SunView
- Sun CGI
- SunCore
- Prism System - Sun Model 3/110

An Overview to Color

The Sun windows and graphics packages support the use of colors. The two Sun windows packages are `suntools` and SunView. Two Sun graphics packages are SunCGI and SunCore. The graphics packages may use colors whether they are running inside or outside the window system.

Customer applications may each create their own colormap, or share a colormap created by a previous application. The colormap is created by defining red, green, and blue color arrays. A user sets the intensity of the color in each array as desired. The colormap index is defined as the index into the red, green, and blue color arrays. For example, colormap color index 3 displays the color defined by the combined contents of `red[3]`, `green[3]`, and `blue[3]`. The user application changes the colormaps on the screen, by changing the contents of the red, green, and blue color arrays; *not* by changing the index.

A total of $256 (2^8)$ colors may be chosen from a palette containing 16 million (256^3) colors. This limitation is imposed by the color frame

buffer hardware. The memory allocated for the color frame buffer has as many colormaps loaded as possible, not to exceed the 256 total colors restriction. If the user applications that are running exceed 256 colors, some colormaps will be swapped out. The system determines which colormaps are loaded, by the user placement of the mouse.

If you run a combination of applications whose colormaps cannot all be held in the frame buffer at the same time, the colormap for the window containing the mouse is placed in the frame buffer so that it fits into a contiguous number of positions. The other windows on your screen may then be displayed in spurious colors, or may be black.

Hardware Color Frame Buffers

The color frame buffers that Sun supports are described below.

`/dev/cgone0`

for Sun 2 color, multibus systems

`/dev/cgtwo0`

for Sun 2 systems with the VME bus, and Sun 3 systems

`/dev/cgfour0`

the Prism frame buffer for Sun model 3/110 systems

`/dev/gpone`

the graphics processor, used with `/dev/cgtwo0` on the Sun model 3/160C and 3/260C systems when this optional hardware board is installed.

Viewing Surfaces

The Sun graphics packages distinguish between the inside and outside of the window system. While running under `sunttools`, the graphics packages use different window devices as the view surfaces.

The color view surfaces for the window system are the 'color graphics pixwins,' called 'cgpixwindds.' The graphics packages while running inside the window system use 'cgpixwindds' or 'gppixwindds.' While running outside the window system, the graphics packages use the raw frame buffer. These devices are known as `cg1dd`, `cg2dd`, `cg4dd`, and `gp1dd`. These devices are described below.

`CG1DD`

the Sun 1 color frame buffer device; and for Sun 2 multibus systems when running outside `sunttools`, in 'console mode'

`CG2DD`

the Sun 2 and Sun 3 color frame buffer device when running outside `sunttools`, in 'console mode'

`CG4DD`

the Sun 3/110 color frame buffer device when running outside `sunttools`, in 'console mode'

- CGPIXWINDD a color window, when running the application in a `suntools` window; or in a SunView canvas subwindow
- GP1DD the graphics processor when running the application outside `suntools`, in 'console mode' (SunCore only)
- GP1PIXWINDD the graphics processor when running the application in a `suntools` window, or in a SunView canvas subwindow (SunCore only)

See the *SunCore Reference Manual*, part number 800-1257; and the *SunCGI Reference Manual*, part number 800-1256, for further information. The 'cg' devices run on the Sun 2 and 3 frame buffers. The 'gp' devices run on the graphics processor in conjunction with the color frame buffer.

The graphics processor is Sun's hardware graphics accelerator. This is a single-board option. An additional graphics buffer board option may also be added at a later time.

Sharing Colormaps

The Unix kernel manages the window system and its colormaps. It gives a name to and then stores a colormap when defined. Any particular colormap may be shared among applications by using the same colormap name. This is true across Sun products such as SunView, SunCore, and SunCGI.

Applications may share colormaps by calling `pw_setcmsname()` with the arguments being the `pixwin` for that window, the colormap name already defined, followed by `pw_putcolormap()`. `pw_putcolormap()` sets the colormap size and points to the colormap's red, green, and blue arrays. When sharing colormaps, the application which defines the colormap must be running before the windows for other applications may attach to that colormap.

The colors of a window may be changed 'on the fly' by modifying the red, green, and blue color arrays; and then by calling `pw_putcolormap()` to recolor the window. The `pw_putcolormap()` call changes the values in the colormap table. The CRT then redisplay the same pixel values. The color index is the same; however, a different color is projected on the screen. The pixels on the screen contain the index number into the colormap. The color which the index represents may be changed by modifying what the colormap represents. This is done by changing the contents of the red, green, and blue arrays. It is faster to change the colormap than to redisplay the screen.

`suntools`

The sun windowing system which is invoked by calling the command `suntools` with the `-f` and `-b` options, sets the default frame buffer colors. These colors are inherited by tools and other applications running inside the window system unless otherwise specified. Without these options, all applications receive the default colormap name 'monochrome' which defines white (red = 255, green = 255, blue = 255) as the background color, and black (red = 0, green = 0, blue = 0) as the foreground color.

The user may invoke tools such as `shelltool`, `textedit`, `cmdtool`, and `gfxtool`, with different foreground and background colors by specifying the `-wf` and `-wb` options. For example, the following `shelltool` is displayed with a red border and blue namestripe. The inside of the tool remains black and white.

```
shelltool -wf 0 0 255 -wb 255 0 0
```

Following the `-wf` and `-wb` options are the red, green, and blue color intensities. For example, the following tool has the foreground color purple and a light blue background, for not only the frame border and namestripe, but also for the window inside the tool. This is accomplished by adding the `-wg` option as the example below shows.

```
shelltool -wf 185 000 184 -wb 102 250 247 -wg
```

SunView

SunView applications may define their own colormaps or share colormaps up to the maximum 256 frame buffer colors. First, define the colormap size for each application. Second, set up the red, green, and blue arrays. Color number 0 is the color defined by `red[0]`, `green[0]`, and `blue[0]`; color number 1 is the color defined by `red[1]`, `green[1]`, and `blue[1]`; and so forth. The size of each colormap must be a power of 2, i.e. {2,4,8,16,32,64,128, or 256}.

A minimum value of 0 is used for no intensity of that color. A maximum value of 255 is used for full intensity of that color. For example, `red[15] = 200`, sets the 16th element of the red array with a very strong red. `Blue[10] = 15`, sets the 11th element of the blue array with a light intensity of blue. The intensity determines how intensely the monitor Cathode Ray Tube (CRT) displays the color.

For a given `pixwin`, use `pw_setcmsname()` to set the colormap name. The arguments are the `pixwin` for the window, and a character string. To bind the colormap to the `pixwin`, use `pw_putcolormap()`. The arguments are the `pixwin` for the window; the starting entry into the colormap; the number of colors; and the red, green, and blue color arrays. Again, the number of colors must be a power of 2.

A SunView example follows.


```

/*
 * SunView color example
 * Draws color lines in a canvas
 */

#include <suntool/sunview.h>
#include <suntool/canvas.h>

#define NCOLORS      4

Frame          frame;
Canvas         canvas;
Pixwin         *pw;

main()
{
    int          i;

    u_char  red[NCOLORS], green[NCOLORS], blue[NCOLORS];

    frame = window_create( NULL, FRAME,
        0);
    canvas = window_create( frame, CANVAS,
        0);

    /*
     *      Set up the red[], green[], and blue[] arrays.
     */

    red[0] = 255;    green[0] = 0;    blue[0] = 0;    /*red */
    red[1] = 0;     green[1] = 255;  blue[1] = 0;    /*green*/
    red[2] = 0;     green[2] = 0;    blue[2] = 255; /*blue */
    red[3] = 208;   green[3] = 173;  blue[3] = 203; /*pink */

    /*
     *      Get the canvas pixwin, initialize the colormap,
     *      and put the colormap into the canvas window.
     */

    pw = canvas_pixwin( canvas);
    pw_setcmsname( pw, "Four Colors"); /*kernel now has this name*/
    pw_putcolormap( pw, 0, NCOLORS, red, green, blue);

    /*
     *      Draw lines in the canvas.
     */

    for (i=1; i < NCOLORS; i++) {

```

```
/*      The PIX_SRC|PIX_COLOR raster op adds the color index
*      to the source pixel value for display.
*/

    pw_rop(pw, 10, i*20, 300, i*20, PIX_SRC|PIX_COLOR(i),0);
}

window_main_loop(frame);
}
```

SunCGI

SunCGI must define its own colormap by creating a new colormap or using a shared colormap whether the SunCGI application is running inside or outside the window system.

The SunCGI color intensity scheme is the same as SunView, ranging from 0 to 255.

In SunCGI, first set the `dd` element of the view surface structure to be the frame buffer type {CG1DD, CG2DD, CG4DD, GP1DD, or CGPIXWINDD}. In the window environment, the graphics processor is accessed through CGPIXWINDD. If the graphics processor is available, the CGPIXWINDD uses that device for transformation calculations. Within the view surface structure, set the `cmapsize` element to the colormap size, and the `cmapname` element to the string that names the colormap. When the view surface is opened, and the red, green, and blue color arrays are initialized; the colormap array of type `Ccentry` points to the red, green, and blue color arrays.

A SunCGI example follows.

```

/*
 *   A SunCGI "C" program
 *   Draws colored lines
 *   Run in a gfxtool
 */

#include <cgidefs.h>
#include <stdio.h>

#define NCOLORS 64
#define MIN     .0
#define MAX     10000

static Ccoor  vpll   = { MIN, MIN }; /* lower left corner */
static Ccoor  vpur   = { MAX, MAX }; /* upper right corner */

main()
{
    int      name;          /* view surface name */
    Cvwsurf device;        /* view surface device */
    Ccoorlist line;        /* line coordinate list */
    Ccoor  points[2];      /* point list */
    int     i;             /* position counter */
    Ccentry clist;         /* color map list */
    u_char  red[NCOLORS];  /* red color map */
    u_char  green[NCOLORS]; /* green color map */
    u_char  blue[NCOLORS]; /* blue color map */

    /* start cgi */
    device.dd = CGPIXWINDD;          /* select output device */
    open_cgi();                       /* initialize cgi */
    open_vws(&name,&device);          /* open view surface */
    vdc_extent(&vpll,&vpur);         /* reset vdc space */

    /* set the line attributes */
    line_width_specification_mode(ABSOLUTE);
    line_width(1.0);

    /* set up the color map */
    for(i=0; i<NCOLORS; i++) {
        red[i] = (i*3);
        green[i] = 64;
        blue[i] = 128;
    }
    clist.n = NCOLORS;
    clist.ra = red;
    clist.ga = green;
    clist.ba = blue;
    color_table(0,&clist);

    /* draw colored lines */

```

```

line.n = 2;
line.ptlist = points;

for (i = 0; i < NCOLORS; i++) {
    line_color(i);
    points[0].y = MIN;
    points[0].x = (i*1000);
    points[1].y = MAX;
    points[1].x = (i*1000);
    polyline(&line);
}
sleep(3);

/* end cgi */
close_vws(name);
close_cgi();
}

```

SunCore

SunCore applications running on the console window, in a `gfxtool`, a `shelltool`, or in a SunView canvas window must define their own colormaps. SunCore color devices are `CG1DD`, `CG2DD`, `CG4DD`, `CGPIXWINDD`, `GP1DD`, and `GP1PIXWINDD`.

SunCore colors do not range from 0 to 255. Instead they range as 256 possible values between 0 and 0.99. Most users familiar with the SunView model may assign their colors as shown in the example below.

```

float red[256];

for (i=0; i<256; i++) {
    red [i] = (float)i * ( (float)1 / (float)256 );
}

```

In SunCore, the view surface `cmapsize` must be set to the size of the color table, and `cmaname` must be set to the colormap name. These must be set before the `initialize_view_surface()` call. After the viewport and window are set up, you may define the color indices for text, line, and fill operations using the `define_color_indices()` call.

A SunCore C-language program example follows.

```

/*
 * SunCore example written in "C"
 * Writes a string in color
 * Run this in a gfxtool
 */
#include      <usercore.h>

#define NCOLORS  8

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

main()
{

    float  red[NCOLORS], green[NCOLORS], blue[NCOLORS];
    float  x, y;
    int    i;

    vwsurf.cmapsize = NCOLORS;
    strcpy (vwsurf.cmapname , "Colormap");

    red  [0] = 0.99;
    green[0] = 0.99;
    blue [0] = 0.99;

    for ( i=1; i < NCOLORS; i++) {
        red  [i] = (float)i * (1.0 / (float)NCOLORS);
        green[i] = (float)i * (1.0 / (float)NCOLORS);
        blue [i] = (float)i * (1.0 / (float)NCOLORS);
    }

    if (initialize_core(BASIC,NOINPUT,TWOD))
        exit(1);
    if (initialize_view_surface(&vwsurf,FALSE))
        exit(2);
    if (select_view_surface(&vwsurf))
        exit(3);
    set_viewport_2 (0.0,1.0,0.0,.75);
    set_window (-100.0,100.0,-100.0,100.0);

    /*
     * SunCore - You pass NCOLORS-1 since SunCore wants to
     * to know where the last value is, not the colormap size.
     */

    define_color_indices(&vwsurf,0,NCOLORS-1,red,green,blue);
    create_temporary_segment();

    x = -100.0;
    y = 90.0;

```

```
for (i = 1; i < NCOLORS; i++ ) {  
    /*  
    * Set line index = color index.  
    */  
  
    set_line_index( i);  
    move_abs_2(x, y);  
    y = y - 20.0;  
    line_abs_2( x, y);  
    x = x + 20.0;  
}  
  
sleep( 5);  
close_temporary_segment();  
deselect_view_surface(&vwsurf);  
terminate_core();  
}
```

When writing SunCore programs in FORTRAN, the programmer must set up the view surface structure array elements as shown in the example that follows. The rest of the code is similar to the C-language example above; in setting up the red, green, and blue arrays; assigning colors with intensities from 0 to 0.99; and applying the colors to lines, text, and fill operations.

A SunCore FORTRAN program example follows.

```

c
c SunCore program in FORTRAN
c Draws two lines
c Run in a gfxtool
c
    include '/usr/include/f77/usercore77.h'

    integer vsurf(VWSURFSIZE)

c
c Initialization of view surface structure.
c
    character *20 screenname, windowname, cmapname
    integer cmapsize
    equivalence (vsurf(1), screenname)
    equivalence (vsurf(6), windowname)
    equivalence (vsurf(14), cmapsize)
    equivalence (vsurf(15), cmapname)

c
c Declarations of all color devices.
c
    integer cg1dd, cg2dd, cgpixwindd
    external cg1dd, cg2dd, cgpixwindd

c
c Create color arrays.
c
    real red(4), green(4), blue(4)

    integer InitializeCore, InitializeVwsurf, SelectVwsurf

c
c
    data vsurf /VWSURFSIZE*0/

    vsurf(DDINDEX) = loc(cgpixwindd)
    if (InitializeCore(BASIC, NOINPUT, TWOD) .ne. 0) call exit(1)

c
c Display current vsurf information.
c
    print *, 'initializecore:'
    print *, 'screenname: ', screenname, 'windowname: ', windowname
    print *, 'cmapname: ', cmapname, 'cmapsize: ', cmapsize

c
c Initialize colormap.
c
    cmapsize = 4

    red(1)          = 0.0
    green(1)        = 0.5
    blue(1)         = 0.0

    red(2)          = 1.0

```

```
green(2)      = 0.5
blue(2)       = 0.0
```

```
red(3)        = 0.0
green(3)      = 1.0
blue(3)       = 0.5
```

```
red(4)        = 1.0
green(4)      = 0.0
blue(4)       = 0.5
```

```
print *, 'red: ', red(1), red(2), red(3)
print *, 'green: ', green(1), green(2), green(3)
print *, 'blue: ', blue(1), blue(2), blue(3)
```

c
c
c

Initialize view surface and window.

```
if (InitializeVwsurf(vsurf, FALSE) .ne. 0) call exit(2)
if (SelectVwsurf(vsurf) .ne. 0) call exit(3)
call SetViewPort2(0.125, 0.875, 0.125, 0.75)
call SetWindow(-50.0, 50.0, -10.0, 80.0)
print *, 'initialization done'
```

c
c
c

First line is drawn on screen after setting colors.

```
call DefColorIndices(vsurf, 1, 4, red, green, blue)
call CreateTempSeg()
print *, 'temporary segment created'
call SetLineIndex(1)
call MoveAbs2(0.0, 0.0)
call LineAbs2(-100.0, 0.0)
print *, 'first line created'
call sleep(2)
```

c
c
c

Second line is drawn on screen after changing colors.

```
call SetLineIndex(2)
call MoveAbs2(0.0, 12.0)
call LineAbs2(-0.0, 100.0)
```

c
c
c

Display current vsurf info.

```
print *, 'my color map:'
print *, 'screenname: ', screenname, 'windowname: ', windowname
print *, 'cmapname: ', cmapname, 'cmapsize: ', cmapsize
```

c
c
c

Close down SunCore.

```
call CloseTempSeg()
call sleep(5)
call DeselectVwsurf(vsurf)
call TerminateCore()
end
```


Prism System - Sun Model 3/110

The model Sun 3/110 systems, known as 'Prism' systems, have frame buffer architecture that is unique among Sun workstations. Such systems have a 10-bit frame buffer that emulates both a 'monochrome' and a color frame buffer. This frame buffer is called `/dev/cgfour0`.

The architecture for the 10-bit deep frame buffer is shown below.

```
1 plane    - enable plane
1 plane    - overlay plane group, monochrome, (/dev/bwtwo0)
8 planes   - color plane group, color, (/dev/cgfour0)
```

where the enable plane bit at a pixel location is set to 0 = color, 1 = monochrome. This is the plane group visible at that pixel location.

The overlay plane group is the black and white plane, the other plane group for color. On the Sun model 3/110 as on all other Sun 2 and 3 color machines, a maximum of 256 colors are visible.

When `suntools` is invoked with no options, all monochrome tools or applications appear from the overlay plane. All color tools or applications appear from the color planes. Tools that are invoked with color options appear from the color plane, and the other tools appear from the overlay plane.

The most common usage of `suntools` on Sun model 3/110 systems is to then run different applications in the different planes. This allows the user to use the model 3/110 as if there were two monitors, like `adjacentscreens` allows. The user invokes `suntools` in the color planes first. Then, from a `shelltool`, invoke a separate `suntools` in the overlay plane. When `suntools` is invoked this way, a `shelltool` with no `-wf` or `-wb` options is running in the overlay or monochrome planes belonging to the `suntools` from which it was invoked. From both the color and overlay planes, `switcher` is run, so that you may toggle between the color and the overlay planes.

An example follows, taken from the `switcher(1)` man page.

>From the console.

```
"suntools -8bit_color_only -toggle_enable"
```

>You are now in the color planes. From a shelltool.

```
"suntools -d /dev/bwtwo0 -toggle_enable -n &"  
<suntools has been started in the overlay plane>
```

```
"switcher -d /dev/bwtwo0 -s i &"
```

```
<a switcher icon should appear to allow you to switch to the overlay  
plane>
```

>Click on the switcher icon, you are now in the overlay plane.

>From a shelltool.

```
"switcher -s o &"
```

```
<a switcher icon should appear to allow you to switch to the color  
planes>
```

On Sun model 3/110 systems, the default frame buffer, /dev/fb, refers to the color portion of the frame buffer. Applications such as screendump(1) and screenload(1) access /dev/fb by default. Thus screendump is equivalent to screendump -f /dev/fb, which is equivalent to screendump -f /dev/cgfour0.

For more information on Sun model 3/110 systems, see the *Release 3.2 Manual*, part number 800-1364.

QUESTIONS, ANSWERS, HINTS, AND TIPS

QUESTIONS, ANSWERS, HINTS, AND TIPS	291
Q&A, and Tip of the Month	291
Errata Corrections	296



QUESTIONS, ANSWERS, HINTS, AND TIPS

Q&A, and Tip of the Month

Hints & Tips #4

This is the fourth in a continuing series of this column which I have created for two purposes.⁵ First, some questions are asked regularly on the AnswerLine. I feel everyone can benefit from distributing discussions of these problems as widely as possible. Second, a large and constantly growing body of information, hints, and tips are not documented anywhere.

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

If you have a question that you would like answered in this column, please mail your question to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Avenue, M/S 2-34, Mountain View, CA 94043. You can also send in your question by electronic mail to *sun!stb-editor*. U. S. customers can call Sun Customer Software Services AnswerLine at 800 USA-4-SUN for technical questions on this column or any other article in this bulletin. I look forward to hearing from you!

Yellow Pages and Mail Aliases

This month we are going to look at the Yellow Pages and how they relate to mail aliases on your system, and then look at the rest of the `.cshrc` aliases I promised last month.

Mail Aliasing

When you send electronic mail, the system uses a set of alias files to allow people to set up mailing lists and make sure that mail gets to the right person. The three places where aliases can be set up are listed below.

⁵ This continuing column is submitted by Chuq Von Rospach, Customer Software Services.

`user/.mailrc`

This file is read by the `/usr/ucb/Mail` program, and allows a user to set up private aliases or mailing lists without administration intervention.

`/usr/lib/aliases`

This file is read by `sendmail`, and can be used to set up mailing lists or aliases global to the entire machine.

`mail.aliases`

This is a yellow pages map found in SunOS releases 3.0 and subsequent releases. This allows an administrator to set up aliases in one location. The aliases are then accessible to all machines within the domain.

Two processes need to be clearly understood: when these files are accessed, and in what order. When you send a piece of mail, the mail system processes it as described below.

Every address is checked against your list of aliases in your `.mailrc` file. If there is a match, the alias is replaced with the list of entries in the `.mailrc` file. One example is shown below.

```
alias friend chuq rx      [.mailrc file]
```

```
% mail friend
```

This becomes translated to the entry that follows.

```
% mail chuq rx
```

In this example, the aliasing is quite limited, simply a straightforward string substitution that is space delimited.

The message is then sent to `sendmail`, which inspects it and checks each mailing address in the order shown below.

```
local aliases
yellow page aliases
```

Note that a match on the local alias does *not* keep the system from trying to rematch on the global alias. If the local alias file changes `'rx'` to `'rx@suntoo'` then the global alias file will not change it again to `'rx@ray.'` However, if you alias something locally that matches something in the yellow page alias, it will get aliased again.

After this aliasing, the mail is sent to the machine that does local delivery. On the local machine, a check is made for a `.forward` file. If it exists, the current mail address is replaced with the address in the `.forward` file. `sendmail` then aliases the address again. This process applies for multiple mail addresses as well. The `.mailrc` is not checked after the first round, though. When both rounds of aliasing are completed, `sendmail` drops the mail in your mailbox and all is done.

Pitfalls to Avoid

There are a few points to remember to ensure proper mail processing. First, it is possible to set up degenerative alias loops using the `mail.alias` file. The most common problem is to set up your aliases using the 'bang' format (as in *machine!user*) instead of using the 'at' format (as in *user@machine*). The 'bang' format worked prior to Sun OS release 3.0. However, with the implementation the yellow pages map, mail then transfers between a client and mailhost until it fails due to too many transfers. 'Bang' addressing should be used only for machines that are outside the local network, and then only when using UUCP for communication.

Second, it is possible to set up forwarding loops with a `.forward` file that will cause mail to be destroyed. In general, you should avoid using the `.forward` file and put your aliases into the yellow pages map instead.

Tip of the Month (TOM)

This month completes the lengthy `.cshrc`. This file includes a lot of miscellaneous aliases, shorthand command names, and other things to make working with Unix a little easier, nicer, and hopefully give you some ideas on customizing your environment so it works best for you. If you have some favorite aliases you want added to the next generation of the monster `.cshrc`, mail them to *sun!stb-editor*.

The script appears on the following pages.

```
#!/bin/csh
# Monster cshrc -- everything you might ever want to do when you
# use csh.

# If we are running a script, do not source .cshrc for speed.
# Prompt is set if we are interactive.
# term is set if we have a tty attached (needed for at)
if (! $?prompt) exit
if (! $?term) exit

# Note that echo is built in and therefore much faster than
# calling pwd as a normal program. This does not work correctly
# across symlinks.
alias pwd 'echo $cwd'

# set up general variables
set history=99 # nice round number

# These aliases let you bring a job to the foreground simply by
# typing the job number. Very convenient.
alias 1 %1
alias 2 %2
alias 3 %3
alias 4 %4
alias 5 %5
alias 6 %6
alias 7 %7
alias 8 %8
alias 9 %9

# Quick pushd/popd -- pushd should really be +,
# but that requires shift.
# is shorthand for pushd $HOME, which tends to happen often.
alias - popd
alias = pushd
alias pushd

# Back up the directory tree quickly.
alias .. "cd .."

# Quick sunview compiling.
alias ccsv "cc * -lsuntool -lsunwindow -lpixrect"
alias ccsvg "cc * -g -lsuntool -lsunwindow -lpixrect"

# lint alias. Lets you look at the libc lint library and check
# a call's parameters. For example, 'check read.'
alias check "grep * /usr/lib/lint/l1ib-lc"

# Convenient shorthands.
alias pe printenv
alias h history
```



```
alias m more
alias clean 'rm *.o core a.out'
alias psa "ps axu | sort -f +0 +1n | more"

# Safety hatches.
alias cp cp -i
alias mv mv -i
alias rm rm -i

# fg brings job into foreground, bg brings job into background,
# and v restarts a stopped vi.
# j lists jobs
# k kills jobs; 'k 1' kills job 1, 'k' kills the most recent job,
# shown with a '+' in the jobs list.
alias v %vi
alias j jobs -l
alias k `kill %*`

# History editor.
# Dump the history into a file, edit it, and then source it back.
# Useful if you have a long, tedious command you do not want
# to retype.
set $hed /tmp/hed$$
alias hed history -h * > $hed; vi + $hed; source -h $hed
```

Errata Corrections



Errata Corrections

This article contains corrections to one article in the May STB issue, and to one bug in the CDB.

Routing to Standalone
Machines

Two corrections need to be made in the 'Routing to Standalone Machines' article. Make the two following corrections.

On page 52, lines 3 and 5, replace the existing network addresses with those shown below. On line 3,

```
192.9.250.6  D
```

and on line 5,

```
thirdnet  192.9.250
```

These corrections are needed since the last three components or parts of an Internet class C network number must be within the range $1 \leq \text{part} \leq 254$.

Customer Distributed BugsList:
Bug 1004564

Two corrections need to be made in the May issue CDB to bug 1004564, on page 89. Note the two `printf` commands on lines 8 and 10 of the code.

For each `printf` command, replace the 'backslash zero' with a 'backslash n', followed by a double quotation mark.

THE HACKERS' CORNER

THE HACKERS' CORNER	299
Determining Memory Size	299
Determining Devices Present	301
An Answermail Script	321



THE HACKERS' CORNER

Determining Memory Size



The Hackers' Corner: Determining Available Memory

There are times when it might be helpful to know how much memory is available at the time you boot your system. This information is available in `/usr/adm/messages` from `dmesg`, but lengthy, intelligent parsing through the file may be required.

Unfortunately, no system or library calls are available that quickly give this information. However, this article contains a program that looks through `kmem` to see what the kernel remembers about its memory.

Areas of Interest

The script or code contained in this article may be of interest to professionals, enthusiasts, or anyone having the time to key the script or code onto their system. If you email the STB editor a request for the script or code at `sun!stb-editor`, we will mail you an online copy. Please include the article name with your request.

Also, please consult your local shell script or programming expert regarding any script or code problems. The script or code is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves.

An Introduction to the Program

The kernel stores the amount of memory in a variable called `physmem`, which is initialized from the monitor Programmable Read Only Memory (PROM) when the kernel boots. The `physmem` value is the number of *pages* of physical memory minus the number of pages reserved by the monitor PROM for its functions. Note that these pages are not available to UNIX under any circumstances.

The number of memory bytes is architecture-dependent. Sun2 workstations contain 2K pages, so you need to multiply the `physmem` value by 2048 to obtain the size of Sun2 physical memory in bytes. Sun3 workstations contain 8K pages, so you then need to multiply by 8192. One of the last lines in the code uses the `getpagesize()` function. This function returns the number of bytes-per-page for your system. You then get your memory size in bytes when

the output of the program is multiplied by the value returned by `getpagesize()`.

The Memory-Size Program

The program that follows looks through `/dev/kmem` to determine how much memory (in pages) you have. It then multiplies this value by the bytes-per-page using `getpagesize()`, yielding the system memory size in bytes. This value is the total memory available from UNIX, exclusive of the Sun monitor PROM reserves.

```
#include <stdio.h>
#include <nlist.h>

struct nlist nl[] = {
    {"_physmem"},
#define X_PHYSMEM 0
    {""},
};

main()
{
    int kmem;          /* file descriptor into /dev/kmem */
    int physmem;      /* where to store obtained physmem -- an
                       * int in the kernel */

    if ((kmem = open("/dev/kmem", 0)) < 0)
    {
        perror("kmem");
        exit(1);
    }

    nlist("/vmunix", nl); /* see nlist(3) */
    if (nl[0].n_type == 0)
    {
        perror("No namelist"); /* stripped your kernel! */
        exit(1);
    }

    /* seek into kernel memory to where the variable lives */
    lseek(kmem, (long) nl[X_PHYSMEM].n_value, 0);
    /* obtain the value */
    if (read(kmem, (char *) &physmem, sizeof(physmem)) != sizeof(physmem))
    {
        perror("read");
        exit(1);
    }

    printf("Number of pages of memory on this system is: %d0, physmem);
    printf("Amount of memory on this system is: 0x%x0, getpagesize()*physmem);
}
```

 Determining Devices Present**The Hackers' Corner:
Determining Devices Present**


There are times when it might be helpful to know what devices are present and connected to your system. Again, you might not want to have to intelligently parse through a file to reinvent the wheel.

This article contains two programs, written by different programmers. Each approaches the effort to determine a machine's configuration differently. Use these programs carefully since investigating what the kernel knows is more of an arcane art than some kind of supported interface. UNIX systems export few, if any, internal kernel interfaces or data structures.

Professional Interest

The script or code contained in this article may be of interest to professionals, enthusiasts, or anyone having the time to key the script or code onto their system. If you email the STB editor a request for the script or code at *sun!stb-editor*, we will mail you an online copy. Please include the article name with your request.

Also, please consult your local shell script or programming expert regarding any script or code problems. The script or code is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves.

 Program 0

This is the first of two programs that determine what devices are present on your system. This program appears on the following pages.

```
#ifndef lint
static char *sccsid = "%Z%%M% %I% %E% SMI";
#endif

/*
 * Print system hardware configuration
 */

#include <stdio.h>
#include <sys/param.h>
#include <sys/fcntl.h>
#include <nlist.h>

#include <sys/buf.h>
#include <sundev/mbvar.h>
#include <sun/autoconf.h>
#include <machine/mmu.h>
#include <machine/cpu.h>

static char *kmemf = "/dev/kmem";
static char *nlistf = "/vmunix";
static int kvm_des;

static struct nlist nl[] = {
#define X_MBDINIT 0
    { "_mbdinit" },
#define X_CPUYPE 1
    { "_cpu" },
#define X_PHYSMEM 2
    { "_physmem" },
    { "" }
};

static int allflg;
static void usage();
static void printconf();
static int kvmread();

extern long lseek();

int
main(argc, argv)
    int argc;
    char **argv;
{
    register char *argp;

    argc--, argv++;
    while (argc > 0 && **argv == '-') {
        argp = *argv++;
        argp++;
    }
}
```



```

    argc--;
    while (*argp++)
        switch (argp[-1]) {

            case 'a':
                allflg++;
                break;

            default:
                usage();
                exit(1);
        }
    }
    if (argc > 1) {
        nlistf = argv[1];
        argv++;
        argc--;
    }
    if (argc > 1) {
        kmemf = argv[1];
        argv++;
        argc--;
    }
    if ((kvm_des = open(kmemf, O_RDONLY)) < 0) {
        (void) fprintf(stderr, "showconfig: Can't open ");
        perror(kmemf);
        exit(1);
    }
    if (nlist(nlistf, nl) < 0) {
        (void) fprintf(stderr, \
            "showconfig: Can't get at kernel namelist0);
            /* XXX - need better error message */
        exit(1);
    }
    printconf();
    return (0);
}

static void
usage()
{
    (void) fprintf(stderr, "usage: showconfig -a [system] [core]0);
    exit(1);
}

static void
printconf()
{
    unsigned long mbdptr;          /* address of mb_device tbl */
    unsigned long drvaddr = 0;    /* address of mb_driver */
    unsigned long ctlraddr = 0;   /* address of mb_ctlr */
    struct mb_driver driver;

```

```
struct mb_ctlr ctlr;
struct mb_device device;
int cpu;
int physmem;
char dname[12];
char cname[12];
char *space;
caddr_t addr;
int intpri;
unsigned long intvec;
struct vec vecs[2];
register int i;
register char *c;

if (kvmread(kvm_des, (unsigned long) nl[X_CPUATYPE].n_value,
    (char *)&cpu, sizeof cpu) < 0) {
    perror("showconfig: Can't read cpu type");
    exit(1);
}
/*      12345678901234567890123456789012345678901234567890 */
switch (cpu) {

#if defined(SUN2_ARCH)
case CPU_SUN2_120:
    (void) printf("Multibus Sun-20);
    break;

case CPU_SUN2_50:
    (void) printf("VMEbus Sun-2 or Sun-2/500);
    break;

default:
    (void) printf("Sun-2, unknown type %#4.4x0, cpu);
    break;
#endif
#if defined(SUN3_ARCH)
case CPU_SUN3_160:
    (void) printf("Sun-3/75, Sun-3/160, or Sun-3/1800);
    break;

case CPU_SUN3_50:
    (void) printf("Sun-3/50 or Sun-3/520);
    break;

case CPU_SUN3_260:
    (void) printf("Sun-3/260 or Sun-3/2800);
    break;

case CPU_SUN3_110:
    (void) printf("Sun-3/1100);
    break;

default:
```

```

        (void) printf("Sun-3, unknown type %#4.4x0, cpu);
        break;
#endif
    }

    if (kvmread(kvm_des, (unsigned long) nl[X_PHYSMEM].n_value,
        (char *)&physmem, sizeof physmem) < 0) {
        perror("showconfig: Can't read amount of physical memory");
        exit(1);
    }

    (void) printf("Physical memory = %dK0, ctob(physmem)/1024);

    mbdptr = nl[X_MBDINIT].n_value;
    (void) printf(
"    DEVICE      SPACE  HEX ADDRESS RANGE      CTRLR  SLV PRI   VEC0);
    while (1) {          /* a 'for' would be a mess here */
        /*
         * get the next mb_device entry
         */
        if (kvmread(kvm_des, mbdptr, \
            (char *)&device, sizeof device) < 0) {
            perror("showconfig: Can't read device entry");
            exit(1);
        }
        if (device.md_driver == 0)          /* end of table */
            break;
        mbdptr += sizeof device;
        /*
         * get the mb_ctrlr and mb_driver, if not current
         */
        if (drvaddr != (unsigned long) device.md_driver) {
            drvaddr = (unsigned long) device.md_driver;
            if (kvmread(kvm_des, drvaddr, (char *)&driver,
                sizeof driver) < 0) {
                perror("showconfig: Can't read driver entry");
                exit(1);
            }
            if (kvmread(kvm_des, (unsigned long)driver.mdr_dname,
                (char *)dname, sizeof dname) < 0) {
                perror("showconfig: Can't read device name");
                exit(1);
            }
        }
        if (device.md_mc != 0) {
            if (kvmread(kvm_des, \
                (unsigned long)driver.mdr_cname,
                (char *)cname, sizeof cname) < 0) {
                perror("showconfig: \
                Can't read controller name");
                exit(1);
            }
        }
    }
}

```

```

if (device.md_mc != 0 && ctrladdr != \
(unsigned long)device.md_mc) {
    ctrladdr = (unsigned long)device.md_mc;
    if (kvmread(kvm_des, ctrladdr, (char *)&ctrl,
        sizeof ctrl) < 0) {
        perror("showconfig: \
Can't read controller entry");
        exit(1);
    }
}

if (!allflg && !device.md_alive)
    continue;      /* unconfigured device */

/*
 * Consistency checking
 */
if (device.md_mc != 0) {
    if (device.md_ctrl == -1) {
        (void) printf(
"%s%-2d - Bad controller number:  md_ctrl: -10,
        dname, device.md_unit);
    }
    if (device.md_slave == -1) {
        (void) printf(
"%s%-2d - Bad slave number:  md_slave: -10,
        dname, device.md_unit);
    }
}

if (device.md_alive && (device.md_mc == 0)) {
    if (device.md_ctrl != -1) {
        (void) printf(
"%s%-2d - Controller number for unspecified controller:  md_ctrl: %d0,
        dname, device.md_unit,
        device.md_ctrl);
    }
    if (device.md_slave != -1) {
        (void) printf(
"%s%-2d - Slave number for unspecified controller:  md_slave: %d0,
        dname, device.md_unit,
        device.md_slave);
    }
}

if (device.md_mc != 0) {
    if (device.md_driver != ctrl.mc_driver) {
        (void) printf(
"%s%-2d - Driver pointer mismatch:  md_driver: %x  mc_driver: %x0,
        dname, device.md_unit,
        device.md_driver, ctrl.mc_driver);
    }
    if (device.md_ctrl != ctrl.mc_ctrl) {

```

```

        (void) printf(
"%s%-2d - Controller number mismatch: md_ctlr: %d mc_ctlr: %d0,
        dname, device.md_unit,
        device.md_ctlr, ctrlr.mc_ctlr);
    }
}

    if (device.md_mc != 0 && device.md_alive && !ctrlr.mc_alive) {
        (void) printf(
"%s%-2d - Controller not marked alive:  %s%-2d0,
        dname, device.md_unit,
        cname, device.md_ctlr);
    }

/*
 * Figure out the address space in which the device is mapped
 * Also, get the interrupt priority and vector
 */
if (device.md_mc != 0) {
    i = SP_BUSMASK & ctrlr.mc_space;
    c = "mc_";
    addr = ctrlr.mc_addr;
    intpri = ctrlr.mc_intpri;
    intvec = (unsigned long)ctrlr.mc_intr;
} else {
    i = SP_BUSMASK & device.md_space;
    c = "md_";
    addr = device.md_addr;
    intpri = device.md_intpri;
    intvec = (unsigned long)device.md_intr;
}

/* Read the first two vectors in */
if (intvec != 0) {
    if (kvmread(kvm_des, intvec, (char *)vecs,
        sizeof vecs) < 0) {
        perror("showconfig: \
        Can't read interrupt vectors");
        exit(1);
    }
}

/*
 * More consistency checking
 */
if (intpri < 0 || intpri >= 7)
    (void) printf(
        "%s%-2d - Illegal priority:  %sintpri: %d0,
        dname, device.md_unit, c, intpri);

if (intvec != 0 && vecs[0].v_func == NULL)
    (void) printf(
        "%s%-2d - Null interrupt handler:  %sintr0,

```

```
        dname, device.md_unit, c);

switch (i) {
case SP_VIRTUAL:
    space = "virtual";
    break;
case SP_OBMEM:
    space = "obmem";
    break;
case SP_OBIO:
    space = "obio";
    break;
case SP_MBMEM:
    space = "mbmem";
    break;
case SP_MBIO:
    space = "mbio";
    break;
case SP_VME16D16:
#if defined(SUN2_ARCH)
    space = "vme16";
#else
    space = "vme16d16";
#endif
    break;
case SP_VME24D16:
#if defined(SUN2_ARCH)
    space = "vme24";
#else
    space = "vme24d16";
#endif
    break;
case SP_VME32D16:
    space = "vme32d16";
    break;
case SP_VME16D32:
    space = "vme16d32";
    break;
case SP_VME24D32:
    space = "vme24d32";
    break;
case SP_VME32D32:
    space = "vme32d32";
    break;
default:
    (void) printf(
        "%s%-2d - Unknown address space: %sspace: %d0,
        dname, device.md_unit, c, i);
case 0:
    space = "????";
}
```

```

/*
 * Now (finally) print out the information
 */
(void) printf("%s", (device.md_alive ? " " : "*"));
(void) printf("%8.8s%-2d ", dname, device.md_unit);
(void) printf("%8.8s ", space);
if (addr != 0) {
    (void) printf("%08x-%08x ",
        addr, (addr + driver.mdr_size - 1));
} else
    (void) printf("    size:%#-6x ", driver.mdr_size);
if (device.md_mc != 0)
    (void) printf("%8.8s%-2d %3d ",
        cname, device.md_ctlr, device.md_slave);
else
    (void) printf("                ");

if (intpri != 0)
    (void) printf("%3d ", intpri);
else
    (void) printf("    ");

if (intvec != 0 && vecs[0].v_func != NULL) {
    for (c = ""; ; ) {
        (void) printf("%s %#x", c, vecs[0].v_vec);
        if (vecs[1].v_func == NULL)
            break;
        intvec += sizeof (struct vec);
        if (kvmread(kvm_des, intvec, (char *)vecs,
            sizeof vecs) < 0) {
            perror("showconfig: \
                Can't read interrupt vectors");
            exit(1);
        }
        c = ",";
    }
} else
    (void) printf("                ");

(void) printf("0");
}
}

static int
kvmread(fd, addr, value_ptr, value_size)
    int fd;
    unsigned long addr;
    char *value_ptr;
    int value_size;

```

```
{
  if (lseek(fd, (long) addr, 0) == -1L
      || read(fd, value_ptr, value_size) != value_size)
    return (-1);
  return (0);
}
```


Program 1

A second program named `probe.c` looks into the kernel to determine what devices are present. It produces an output similar to that shown below.

```
astra% probe
si0 at obio 0x140000 pri 2
st0 at si0 slave 0
st0 at si0 slave 32
zs0 at obio 0x20000 pri 3
zs1 at obio 0x0 pri 3
le0 at obio 0x120000 pri 3
bwtwo0 at obmem 0x100000 pri 4
des0 at obio 0x1c0000 not attached
astra%
```

Again, use this program on an experimental basis. Consult your local experts to see what they have to say.... Good luck!

```
/*
 * lists all devices defined for a configuration
 * and indicates if a device has not been attached
 *
 * usage: probe [vmunix]
 *
 * mark opperman
 * sun europe
 * 20 aug 86
 */
```

```
#include <stdio.h>
#include <ctype.h>
#include <nlist.h>
#include <sys/param.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/buf.h>
#include <sys/vmmac.h>
#include <sys/dkbad.h>
#include <machine/param.h>
#include <machine/pte.h>
#include <sun/dklabel.h>
#include <sun/dkio.h>
#include <sundev/mbvar.h>
#include <sundev/screg.h>
#include <sundev/sireg.h>
#include <sundev/scsi.h>
```

```
struct nlist nl[] = {
    { "_Sysmap", 0, 0, 0, 0 },
#define NL_SYSMAP 0
    { "_mbcinit", 0, 0, 0, 0 },
#define NL_MBCINIT 1
    { "_mbdinit", 0, 0, 0, 0 },
#define NL_MBDINIT 2
    { "_scdriver", 0, 0, 0, 0 },
#define NL_SCDRIVER 3
    { "_scsi_ntype", 0, 0, 0, 0 },
#define NL_SCSINTYPE 4
    { "_scsi_unit_subr", 0, 0, 0, 0 },
#define NL_SCSIUNITSUBR 5
    { "" },
#define NL_LAST 6
};

#define physaddr(addr) (addr - KERNELBASE)

#define MAX_SCSI_DEV_NAME_LENGTH 3

int mem;
```

```

struct pte *Sysmap;
struct mb_ctlr *mbcinit;
struct mb_device *mbdinit;
struct mb_driver *scdriver;
int scsi_n_type;
struct scsi_unit_subr *scsi_unit_subr;
char *vmunix = "/vmunix";

struct pte getpte();
extern char *malloc();
extern char *calloc();

main(argc, argv)
    int argc;
    char **argv;
{
    if (argc > 2) {
        fprintf(stderr, "usage: %s [vmunix]0, argv[0]);
        exit(1);
    }

    if (argc == 2)
        vmunix = argv[1];

    if ((mem = open("/dev/mem", O_RDONLY)) < 0) {
        perror("can't open /dev/mem");
        exit(1);
    }

    getkvars();

    process();
}

getkvars()
{
    register char *devname;
    register i;

    nlist(vmunix, nl);
    if (nl[NL_SYSMAP].n_type == 0 || nl[NL_MBCINIT].n_type == 0 ||
        nl[NL_MBDINIT].n_type == 0) {
        fprintf(stderr, "no namelist0);
        exit(1);
    }

    for (i=0; i<NL_LAST; i++) {
        if (i == NL_SCDRIVER) /* need virtual address */
            continue;
        if (nl[i].n_value >= KERNELBASE)
            nl[i].n_value = physaddr(nl[i].n_value);
    }
}

```

```

Sysmap = (struct pte *) nl[NL_SYSMAP].n_value;
if (scdriver = (struct mb_driver *) nl[NL_SCDRIVER].n_value) {
    kread(mem, nl[NL_SCSINTYPE].n_value, &scsi_notype,
          sizeof(scsi_notype));
    scsi_unit_subr = (struct scsi_unit_subr *)
        calloc(scsi_notype, sizeof(struct scsi_unit_subr));
    kread(mem, nl[NL_SCSIUNITSUBR].n_value, scsi_unit_subr,
          scsi_notype * sizeof(struct scsi_unit_subr));
    for (i=0; i<scsi_notype; i++) {
        devname = malloc(MAX_SCSI_DEV_NAME_LENGTH);
        kread(mem, scsi_unit_subr[i].ss_devname, devname,
              MAX_SCSI_DEV_NAME_LENGTH);
        scsi_unit_subr[i].ss_devname = devname;
    }
}
}

process()
{
    struct mb_ctlr mb_ctlr;
    struct mb_device mb_device;
    u_int addr;
    int n;

    for (addr = nl[NL_MBCINIT].n_value, n=0; ; n++) {
        kread(mem, addr, (caddr_t) &mb_ctlr,
              sizeof(struct mb_ctlr));
        if (!mb_ctlr.mc_driver)
            break;
        addr += sizeof(struct mb_ctlr);
    }
    /*
     * Allocate one more controller than really exists and
     * mark its driver as zero to indicate the end
     * of the controllers.
     */
    mbcinit = (struct mb_ctlr *) calloc(n+1, sizeof(struct mb_ctlr));
    kread(mem, nl[NL_MBCINIT].n_value, (caddr_t) mbcinit,
          n * sizeof(struct mb_ctlr));
    mbcinit[n].mc_driver = (struct mb_driver *) 0;

    for (addr = nl[NL_MBDINIT].n_value, n=0; ; n++) {
        kread(mem, addr, (caddr_t) &mb_device,
              sizeof(struct mb_device));
        if (!mb_device.md_driver)
            break;
        addr += sizeof(struct mb_device);
    }
    /*
     * Allocate one more device than really exists and
     * mark its driver as zero to indicate the end
     * of the devices.

```

```

    */
    mbdinit = (struct mb_device *) calloc(n+1, sizeof(struct mb_device));
    kread(mem, nl[NL_MBDINIT].n_value, (caddr_t) mbdinit,
          n * sizeof(struct mb_device));
    mbdinit[n].md_driver = (struct mb_driver *) 0;

    init_ctlr_drivers();
    init_device_drivers();
    display();
}

/*
 * Emulate mapping in this process' address space.
 */
update_drivers(old, new)
register struct mb_driver *old;
register struct mb_driver *new;
{
    register struct mb_ctlr *mc;
    register struct mb_device *md;

    if (scdriver == old)
        scdriver = new;

    for (mc=mbcinit; mc->mc_driver; mc++)
        if (mc->mc_driver == old) {
            mc->mc_driver = new;
        }
    for (md=mbdinit; md->md_driver; md++)
        if (md->md_driver == old) {
            md->md_driver = new;
        }
}

/*
 * Read in structures referenced by the mb_ctlr struct
 * and change pointers.
 */
init_ctlr_drivers()
{
    struct mb_ctlr *mc;
    struct mb_driver *mdr;
    struct vec *vec;
    char buf[16];

    for (mc=mbcinit; mdr=mc->mc_driver; mc++) {
        if (mdr < (struct mb_driver *) KERNELBASE)
            goto intr;

        mdr = (struct mb_driver *) malloc(sizeof(struct mb_driver));
        kread(mem, mc->mc_driver, mdr, sizeof(struct mb_driver));
    }
}

```

```

update_drivers(mc->mc_driver, mdr);
if (mdr->mdr_dname) {
    kread(mem, mdr->mdr_dname, buf, sizeof(buf));
    mdr->mdr_dname = malloc(strlen(buf)+1);
    strcpy(mdr->mdr_dname, buf);
}
if (mdr->mdr_cname) {
    kread(mem, mdr->mdr_cname, buf, sizeof(buf));
    mdr->mdr_cname = malloc(strlen(buf)+1);
    strcpy(mdr->mdr_cname, buf);
}
intr:
if (mc->mc_intr) {
    vec = (struct vec *) malloc(sizeof(struct vec));
    kread(mem, mc->mc_intr, vec, sizeof(struct vec));
    mc->mc_intr = vec;
}
}

/*
 * Read in structures referenced by the mb_device struct
 * and change pointers.
 */
init_device_drivers()
{
    struct mb_device *md;
    struct mb_driver *mdr;
    struct vec *vec;
    char buf[16];

    for (md=mbdinit; mdr=md->md_driver; md++) {
        if (mdr < (struct mb_driver *) KERNELBASE)
            goto intr;

        mdr = (struct mb_driver *) malloc(sizeof(struct mb_driver));
        kread(mem, md->md_driver, mdr, sizeof(struct mb_driver));
        update_drivers(md->md_driver, mdr);
        if (mdr->mdr_dname) {
            kread(mem, mdr->mdr_dname, buf, sizeof(buf));
            mdr->mdr_dname = malloc(strlen(buf)+1);
            strcpy(mdr->mdr_dname, buf);
        }
        if (mdr->mdr_cname) {
            kread(mem, mdr->mdr_cname, buf, sizeof(buf));
            mdr->mdr_cname = malloc(strlen(buf)+1);
            strcpy(mdr->mdr_cname, buf);
        }
    }
intr:
    if (md->md_intr) {
        vec = (struct vec *) malloc(sizeof(struct vec));
        kread(mem, md->md_intr, vec, sizeof(struct vec));
    }
}

```

```

        md->md_intr = vec;
    }
}

/*
 * Display all controllers/devices alive or not on the system.
 * Uses info in mbcinit and mbdinit.
 */
display()
{
    register struct mb_ctlr *mc;
    register struct mb_device *md;
    register struct mb_driver *mdr;
    register char *name;

    for (mc=mbcinit; mdr=mc->mc_driver; mc++) {
        doprobe(mc->mc_addr, mc->mc_space, mdr->mdr_cname,
            mc->mc_ctlr, mc->mc_alive, mc->mc_intpri,
            mc->mc_intr);

        /*
         * Now look for devices attached to this controller
         * (even if it's not attached).
         */
        for (md=mbdinit; md->md_driver; md++) {
            if (md->md_driver != mdr ||
                md->md_driver == (struct mb_driver *) -1 ||
                md->md_ctlr != mc->mc_ctlr)
                continue;

            /*
             * SCSI devices kludge...
             */
            if (md->md_driver == scdriver) {
                mdr->mdr_dname =
                    scsi_unit_subr[TYPE(md->md_flags)].ss_devname;
            }
            printf("%s%d at %s%d slave %d ",
                mdr->mdr_dname, md->md_unit,
                mdr->mdr_cname, mc->mc_ctlr, md->md_slave);
            if (!md->md_alive)
                printf("not attached");
            putchar('0');
            /*
             * -1 indicates that info has already been displayed
             * so not redisplayed below.
             */
            md->md_driver = (struct mb_driver *) -1;
        }
    }

    for (md=mbdinit; mdr=md->md_driver; md++) {

```

```

    if (mdr == (struct mb_driver *) -1)
        continue;

    doprobe(md->md_addr, md->md_space, mdr->mdr_dname,
            md->md_unit, md->md_alive, md->md_intpri,
            md->md_intr);
}
}

doprobe(addr, space, name, num, alive, intpri, intr)
caddr_t addr;
u_int space;
char *name;
short num;
short alive;
int intpri;
struct vec *intr;
{
    char *addrspace;
    struct pte pte;

    if (alive) {
        pte = getpte(addr);
        addr = (caddr_t) ((u_int) ptob(pte.pg_pfnnum) |
                          ((u_int) addr & PGOFSET));
    }

#define SP_BUSMASK      0x0000FFFF      /* mask for bus type */
#define SP_VIRTUAL     0x00000001
#define SP_OBMEM       0x00000002
#define SP_OBIO        0x00000004
#define SP_VME16D16    0x00000100
#define SP_VME24D16    0x00000200
#define SP_VME32D16    0x00000400
#define SP_VME16D32    0x00001000
#define SP_VME24D32    0x00002000
#define SP_VME32D32    0x00004000

    switch (space & SP_BUSMASK) {
    case SP_VIRTUAL:
        addrspace = "virtual";
        break;
    case SP_OBMEM:
        addrspace = "obmem";
        break;
    case SP_OBIO:
        addrspace = "obio";
        break;
    case SP_VME16D16:
        addrspace = "vme16d16";
        if (alive)
            addr = (caddr_t) ((int) addr & 0xffff0000);

```



```

        break;
    case SP_VME24D16:
        addrspace = "vme24d16";
        if (alive)
            addr = (caddr_t) ((int) addr & 0xff000000);
        break;
    case SP_VME32D16:
        addrspace = "vme32d16";
        break;
    case SP_VME16D32:
        addrspace = "vme16d32";
        if (alive)
            addr = (caddr_t) ((int) addr & 0xffff0000);
        break;
    case SP_VME24D32:
        addrspace = "vme24d32";
        if (alive)
            addr = (caddr_t) ((int) addr & 0xff000000);
        break;
    case SP_VME32D32:
        addrspace = "vme32d32";
        break;
    default:
        addrspace = "unknown";
        break;
}

printf("%s%d at %s 0x%x ", name, num, addrspace, addr);

if (alive) {
    if (intpri) {
        if (intr == (struct vec *) 0)
            printf("pri %d ", intpri);
        else
            printf("vec 0x%x ", intr->v_vec);
    }
}
else {
    printf("not attached");
}
putchar('0');
}

kread(fd, off, into, size)
int fd;
long off;
caddr_t into;
u_int size;
{
    if (off >= KERNELBASE)
        off = physaddr(off);
}

```

```
lseek(fd, off, 0);
if (read(fd, into, size) != size) {
    perror("kread: read failed");
    exit(1);
}
}

struct pte
getpte(a)
u_int a;
{
    u_int v;
    struct pte pte;

    v = btop(physaddr(a));
    kread(mem, (long) (Sysmap + v), &pte, sizeof(struct pte));
    return(pte);
}
```

An Answermail Script

The Hackers' Corner: An Answermail Script

The script contained in this article may be used to respond to incoming mail messages when you are away from your workstation. It is particularly useful during training programs, time off, vacations, and the like.

Professional Interest

This script may be of interest to professionals, enthusiasts, or anyone having the time to key the script or code onto their system. If you email the STB editor a request for the script or code at *sun!stb-editor*, we will mail you an online copy. Please include the article name with your request.

Also, please consult your local shell script or programming expert regarding any script or code problems. This script or code is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves.

Some things that you will be able to do with the answermail script include defining infrequently-changing answermail parameters (for example, your backup contact person) with environment variables. You will also be able to set other parameters using command-line arguments.

Use the `at (1)` command to install your answermail script. You will be able to set up the answermail script installation as soon as you know when you will be away from your workstation, avoiding any last minute rush or the chance to forget.

Installation: General

To activate your mail answering machine, install the script, for example, as `/usr/local/answermail`. To make it executable, use the command shown below. Note that your response to the machine prompt is shown in bold.

```
% chmod +x /usr/local/answermail
```

If the filesystem where the script is installed is an NFS filesystem, it should be one that is hard-mounted on any workstation using the answermail script. Otherwise, incoming mail may return to the sender with confusing error messages, if the server is not responding at the moment the mail arrives.

Please note the directory pathname where the shell script is installed must be entirely in lower case. The script will verify that this is true.

Installation: Specific User

To install the automatic answering machine for a particular user, log in as that user and run `answermail -i`. The script will prompt for some information and then create a file named `.forward`. The file will be located in that user's home directory. Once the `.forward` file is created, answermail is 'on' for that user.

The `.forward` file contains one line as shown below.

```
| /usr/local/answermail -a mail_dir
```

The first character is a vertical bar, the pipe symbol. The flag `-a` indicates that you want to run in answer mode. `mail_dir` specifies where you want to store the saved mail.

Again, please note that the directory pathname of the user's mail must be entirely in lower case. The script will verify that this is true.

Reading Mail When You Return

When you return from your absence, log in and run `answermail -r`. This will remove the `.forward` file, the log of to whom the answering machine sent notices, and will put you in `mail`, reading the log of mail that arrived while you were away.

The only mail reader program supported by this `answermail` script is Berkeley Mail. It cannot accommodate Emacs or Mh, for two examples. To use other mail reader programs, use the SunView program `mailtool` to read the folder. Exit Berkeley Mail using `exit` or `x` and invoke `Mailtool` on the folder `answermail-maillog`.

Installation Details

The `answermail` script checks the environment variable `ANS_MAILDIR` upon installation, and prompts for your mail directory pathname if the environment variable is not set. This information is *required!* `ANS_MAILDIR` is the directory (full UNIX pathname) where all `answermail` parameter files and your messages will be stored. Note that the directory name must be in lower case, for compatibility with older versions of the `sendmail` program.

The following paragraphs show the questions that `answermail` asks; the corresponding shell variable, if any; and an explanation of your response needed. The `answermail` script creates a file in the user's mail directory called `answermail-parameters` that contains lines that initialize these shell variables.

Your Return Day or Date? (`PHONE_BACKUP_MAILADDRESS`)

This is the optional date to incorporate in the message sent to those who send you mail. The default is for no return date in the reply.

Explanation for Your Absence? (`ABSENCE_MSG`)

This is the optional explanation of why you are away from your workstation. The explanation should be a complete sentence, since it will appear on a line by itself in the response message. The default is for no explanation in the reply.

Mail Address for Your Backup Contact Person

(`PHONE_BACKUP_MAILADDRESS`) This is the optional mail address to which to forward your phone messages while

you are away from your workstation. An example is *jdoe@sun*. 'Phone messages' are defined as any mail that contains the word 'phone' in the subject line (case-insensitive). For example, you may want to forward your phone messages to a receptionist to a department administrator. The default is for no special treatment of phone messages.

Backup Contact Person (BACKUP_CONTACT_DETAIL)

This is the optional contact direction for your backup contact person. This is included in the message sent to people who send you mail. For example, *Jane Doe, department administrator, (415) 987-1234*. The default is for no backup contact person information in the reply.

Respond to Mailing List Messages?

This is an optional check to determine whether the message was sent to you personally, or to a mailing list that includes you. The default is not to respond to messages sent to mailing lists. You should answer 'yes' only in those cases where you want to send the 'I'm away' reply to *every* message you receive, including junk mail message senders and all other mailing lists.

Other Aliases to Check? (NAMELIST)

This is for any optional alternate names in the mail address that you have on the system. The default is to include your login name only.

Installation Shortcuts

Several of these parameters may be predefined as environment variables. In this case, the answermail script uses the environment variables and does not prompt interactively for the values. This allows a more rapid and easier automatic answermail installation using the UNIX `at(1)` command. See the *Commands Reference Manual*, part number 800-1295, for details.

Parameters that may be defined in environment variables are shown below.

```
ANS_MAILDIR
ANS_PHONE_BACKUP_MAILADDRESS
ANS_BACKUP_CONTACT_DETAIL
ANS_NAMELIST
```

When defining these environment variables, for example, in your `.login` file, be sure that multi-word values are enclosed in double quotation marks. This is shown in the examples that follow. Note also that the `NAMELIST` syntax is an `egrep`-style regular expression. Each alias is separated by a vertical bar, `|`, and the entire expression must be enclosed in double quotation marks. An example is `john|jsmith`.

To further aid automatic installation, your return date and message explaining your absence may be specified as command-line arguments when running `answermail -i`. See the example shown below.

```
answermail -i "Tuesday, March 19" "I have gone to Europe."
```

If you supply only one argument, it will be used as the return date. Third and following arguments will cause an error. Be sure to place multi-word argument between double quotation marks.

An example of using the `at(1)` command to automatically set up the `answermail` script at a future date is shown below. Note again that your responses to the machine prompts are shown in **bold** in the following example.

```
machine% at 1:00A May 28  
at> answermail -i "Monday, June 14" "I have gone to Europe."  
at> ^D  
machine%
```

The Answermail Script

The `answermail` script appears on the following pages. Use it as an experiment, along with any advice from your local shell script or programming expert. Good luck!

```

#!/bin/sh -u
#
# Automatic mail answering machine
#
#   static char sccsid [] = "@(#)answermail 1.21 87/06/26 SMI";
#
#
BASENAME=/usr/bin/basename
LS=/bin/ls
CAT=/bin/cat
GREP=/bin/grep
RM=/bin/rm
PWD=/bin/pwd
AWK=/bin/awk
SED=/bin/sed
TOUCH=/usr/bin/touch
ECHO=/bin/echo
ERROR=/bin/echo
DATE=/bin/date
MAIL=/usr/ucb/Mail
CHMOD=/bin/chmod
HOSTNAME=/bin/hostname
TR=/usr/bin/tr
LOWERCASE="$TR ' [A-Z]' ' [a-z]'"

#
# NOTE:
#
# Some of these filenames must be all in lower case.
# It is good coding practice to make them all in lower case.
#
ANSWERMAIL_SUBJECT_LINE="Away from my mail"
HEARDFROM_FILENAME=answermail-heardfrom
MAILLOG_FILENAME=answermail-maillog
PARAMS=answermail-parameters
LOCKFILE=/tmp/answermail-lock

SCRIPTNAME=`$BASENAME $0`
SYNTAX="$SCRIPTNAME -a|-i|-r [ Mail_Dir ]"
case $# in
0 )
    # With no args at all, out of here
    $ERROR 1>&2 $SCRIPTNAME: Syntax: $SYNTAX
    exit 1
    ;;
esac
case ${HOME-"No Home?"} in
"No Home?" )
    $ERROR 1>&2 $SCRIPTNAME: Must have home directory in environment.
    exit 1

```

```

;;
esac
#
# Make sure that files will be readable by the sendmail daemon
# later (it runs as root initially), even if the .forward file
# and others are being created in a directory in an NFS filesystem.
#
umask 133
case "$1" in
-a )
    # Continue below with automatic answer procedure.
    shift
    ;;
-i )
    # Invoke installation procedure.
    case $# in
    1 )
        # No command line arguments - have to prompt for them.
        ;;
    2 )
        # Second argument is return day/date.
        RETURN_DATE="$2"
        ;;
    3 )
        # Second argument is return day/date;
        # third argument is explanation of absence.
        RETURN_DATE="$2"
        ABSENCE_MSG="$3"
        ;;
    * )
        $ERROR 1>&2 $SCRIPTNAME: Syntax: $SYNTAX
        exit 1
        ;;
    esac
    $ECHO $SCRIPTNAME: Begin installation procedure...
    $ECHO ""
    #
    # Home directory must be publicly executable, required so that
    # sendmail can read the .forward file, even if the home directory
    # directory is in an NFS filesystem.
    #
    $LS -ll $HOME | $GREP -s "^d.....x"
    case $? in
    0 )
        # Is publicly readable - continue
        ;;
    * )
        $ERROR 1>&2 $SCRIPTNAME: Your home directory must have public
        $ERROR 1>&2 $SCRIPTNAME: execute permission for sendmail to read
        $ERROR 1>&2 $SCRIPTNAME: your .forward file. If that is not
        $ERROR 1>&2 $SCRIPTNAME: acceptable, you cannot use $SCRIPTNAME.
        $ERROR 1>&2 $SCRIPTNAME: If it is acceptable, type the command
        $ERROR 1>&2 $SCRIPTNAME: "      " chmod o+x $HOME

```



```

        $ERROR 1>&2 $SCRIPTNAME: and run answermail again.
        exit 1
        ;;
    esac
    case "$0" in
    /* )
        SCRIPTPATH=$0
        ;;
    /*/* )
        SCRIPTPATH=`$PWD`/$0
        ;;
    * )
        $ECHO "(Required) Directory where this script lives?"
        $ECHO -n "=> "
        read SCRIPTHOME
        if [ ! -r "$SCRIPTHOME"/$SCRIPTNAME ]
        then
            $ERROR 1>&2 $SCRIPTNAME: I do not exist there.
            exit 1
        fi
        SCRIPTPATH="$SCRIPTHOME"/$SCRIPTNAME
        if [ "$SCRIPTPATH" != ` $ECHO $SCRIPTPATH | $LOWERCASE ` ]
        then
            $ERROR 1>&2 $SCRIPTNAME: Script pathname must be lower case.
            exit 1
        fi
        ;;
    esac
    case ${ANS_MAILDIR-"Not set"} in
    "Not set" )
        $ECHO "(Required) Full pathname of your mail directory?"
        $ECHO -n "=> "
        read MAILDIR
        ;;
    * )
        MAILDIR="$ANS_MAILDIR"
        ;;
    esac
    if [ ! -d "$MAILDIR" ]
    then
        $ERROR 1>&2 $SCRIPTNAME: Mail directory "$MAILDIR" does not exist.
        exit 1
    fi
    if [ "$MAILDIR" != ` $ECHO $MAILDIR | $LOWERCASE ` ]
    then
        $ERROR 1>&2 $SCRIPTNAME: Mail directory "$MAILDIR" must be lower case.
        exit 1
    fi
    if [ -r $MAILDIR/$PARAMS ]
    then
        $ECHO $SCRIPTNAME: Already installed in your mail directory.
        $ECHO -n "$SCRIPTNAME: Proceed anyway? "
        read RESPONSE
    fi

```

```

case "$RESPONSE" in
y | yes | Y | YES )
    ;;
* )
    $ERROR 1>&2 $SCRIPTNAME: Installation aborted.
    exit 1
    ;;
esac
fi
$RM -f $HOME/.forward $MAILDIR/$PARAMS \
    $MAILDIR/$HEARDFROM_FILENAME \
    $MAILDIR/$MAILLOG_FILENAME
case ${RETURN_DATE-"Not set"} in
"Not set" )
    $ECHO "(Optional) Your return day/date?"
    $ECHO -n "=> "
    read RETURN_DATE
    ;;
esac
case "$RETURN_DATE" in
"" )
    RETURN_DATE=none
    ;;
esac
case ${ABSENCE_MSG-"Not set"} in
"Not set" )
    $ECHO "(Optional) Explanation for your absence?"
    $ECHO -n "=> "
    read ABSENCE_MSG
    ;;
esac
case "$ABSENCE_MSG" in
"" )
    ABSENCE_MSG=none
    ;;
esac
case ${ANS_PHONE_BACKUP_MAILADDRESS-"Not set"} in
"Not set" )
    $ECHO "(Optional) Mail address for phone message backup?"
    $ECHO -n "=> "
    read PHONE_BACKUP_MAILADDRESS
    ;;
* )
    PHONE_BACKUP_MAILADDRESS="$ANS_PHONE_BACKUP_MAILADDRESS"
    ;;
esac
case "$PHONE_BACKUP_MAILADDRESS" in
"" )
    PHONE_BACKUP_MAILADDRESS=none
    ;;
* )
    $ECHO ` $DATE ` "    $PHONE_BACKUP_MAILADDRESS" \
        >> $MAILDIR/$HEARDFROM_FILENAME

```



```

                ;;
            * )
                NAMELIST="$NAMELIST|$ALIAS"
                ;;
        esac
    done
    ;;
esac
;;
* )
    NAMELIST="$ANS_NAMELIST"
    ;;
esac
NAMELIST=`$ECHO $NAMELIST | $SED -e "/'/s/'/'\"'/g"`
$ECHO "NAMELIST=' $NAMELIST' "      >>$MAILDIR/$PARAMS
#
# All parameters are set in the parameter file,
# now install the '.forward' command to invoke the script.
#
$ECHO "
$ECHO ""
$ECHO $$SCRIPTNAME: Automatic mail answering service installed.
exit 0
;;
-r )
# Invoke removal and read procedure.
# Complain if extraneous arguments.
case $# in
1 )
    # Continue
    ;;
* )
    $ERROR 1>&2 $$SCRIPTNAME: Syntax: $$SYNTAX
    exit 1
    ;;
esac
$RM -f $HOME/.forward
case ${ANS_MAILDIR-"Not set"} in
"Not set" )
    $ECHO $$SCRIPTNAME: Pathname of your mail directory?
    $ECHO -n "=> "
    read MAILDIR
    ;;
* )
    MAILDIR="$ANS_MAILDIR"
    ;;
esac
if [ ! -d "$MAILDIR" ]
then
    $ERROR 1>&2 $$SCRIPTNAME: Mail directory "$MAILDIR" does not exist.
    exit 1
fi
$RM -f $MAILDIR/$HEARDFROM_FILENAME $MAILDIR/$PARAMS

```

```

$ECHO $$SCRIPTNAME: Automatic mail answering service removed.
if [ -s $MAILDIR/$MAILLOG_FILENAME ]
then
    $MAIL -f $MAILDIR/$MAILLOG_FILENAME
    exit $?
fi
$RM -f $MAILDIR/$MAILLOG_FILENAME
$ECHO $$SCRIPTNAME: No answermail backlog in ${MAILDIR}.
exit 0
;;
* )
    # Bogus switch, report syntax error
    $ERROR 1>&2 $$SCRIPTNAME: Syntax: $$SYNTAX
    exit 1
    ;;
esac

#
# ===== Automatic response section =====
#

case $# in
0 )
    # Without at least a place to put the mail, we can not continue.
    $ERROR 1>&2 $$SCRIPTNAME: Missing mail log directory?
    exit 1
    ;;
1 )
    MAILDIR=$1
    ;;
* )
    # BUG - (Or someone edited their .forward file.)
    $ERROR 1>&2 $$SCRIPTNAME: Too many args "$#" for "'-a'".
    exit 1
    ;;
esac
HEARDFROM=$MAILDIR/$HEARDFROM_FILENAME
MAILLOG=$MAILDIR/$MAILLOG_FILENAME
MSGTMP=/tmp/answermail-tmp_$$
MSGTMP1=/tmp/answermail-tmp1_$$
MSGDATA=/tmp/answermail-data_$$
#
# Clean up interrupts.
#
trap "$RM -f $MSGTMP $MSGTMP1 $MSGDATA; $ERROR 1>&2 Interrupted.; exit 1" 1 2

$RM -f $MSGTMP $MSGTMP1 $MSGDATA          # Make sure temp files clear.
$CAT > $MSGTMP                             # Save incoming message.
if [ $? != 0 -o ! -s $MSGTMP ]
then
    #
    # The script will not actually die at this point from
    # losing the incoming message. But if the message is

```

```

# lost, we may as well stop now.
#
$ERROR 1>&2 $SCRIPTNAME: Sorry - error while receiving your message.
$ERROR 1>&2 $SCRIPTNAME: /tmp filesystem on '$HOSTNAME' probably full!
$RM -f $MSGTMP $LOCKFILE
exit 1
fi
#
#
# Instantiate the parameters created at install time and check validity.
#
NAMELIST=none
PHONE_BACKUP_MAILADDRESS=none
RETURN_DATE=none
ABSENCE_MSG=none
BACKUP_CONTACT_DETAIL=none
case "$NAMELIST" in
none )
# This is the last parameter put in the file - something is wrong.
$ERROR 1>&2 $SCRIPTNAME: Invalid parameter file from installation.
$RM -f $MSGTMP $LOCKFILE
exit 1
;;
esac
#
# Parse the mail message for sender, addressee, subject, etc.
#
# This awk script attempts to reliably parse portions of
# the mail header. The idea is to look for the first 'From:'
# line in the message, the first 'Subject:' line, if any, and
# your own name in the 'To:' or 'Cc:' lines' if any. With
# this information we can avoid answering mail to mailing
# lists or blind copies.
#
# But first, protect any single-quote characters in the
# input message in case they are in, for example, the subject
# field and would cause problems later on when we
# instantiate the values of the key fields. To do so,
# change all single-quotes (') to (''').
#
# Interpret the following command as
#  /'/s/'/' " ' " '/g
$SED -e "/'/s/'/'''''''''''''''''/g" $MSGTMP > $MSGTMP1
if [ $? != 0 -o ! -s $MSGTMP1 ]
then
$ERROR 1>&2 $SCRIPTNAME: Sorry - error while processing your message.
$ERROR 1>&2 $SCRIPTNAME: /tmp filesystem on '$HOSTNAME' probably full!
$RM -f $MSGTMP $MSGTMP1 $LOCKFILE
exit 1
fi
$AWK -f - $MSGTMP1 << !End!Awk!Script! > $MSGDATA
BEGIN
    { sender = ""; subject = "";
      to = ""; into = 0; cc = ""; incc = 0 }

```

```

(sender==" " && == "From:") { sender = ;
    printf("SENDER=' ');
    for (i=2; i < NF; i++) {
        printf("%s ", );
    }
    printf("%s'0, F);
    next;
}
(to==" " && == "To:") { into = 1 }
(into!=0 && /$NAMELIST/) { to = "[You are on the list]" }
(into!=0 && substr(,length(),1)!=":" && != "To:")\
    { into = 0 }
(subject==" " && == "Subject:")\
    { subject = ;
    printf("SUBJECT=' ');
    for (i=2; i < NF; i++) {
        printf("%s ", );
    }
    printf("%s'0, F);
    next;
}
(to==" " && == "Cc:") { incc = 1 }
(incc!=0 && /$NAMELIST/) { to = "[You are on the list]" }
(incc!=0 && substr(,length(),1)!=":" && != "Cc:")\
    { incc = 0 }
(NF==0 || == "Status:") { printf("ENVELOPE_END_LINE=%d0, NR);
    exit }
END { if (subject == "") {
    subject="?-no subject line in your message-?";
    printf("SUBJECT=' (%s)'0, subject);
    }
    if (sender == "" ) {
        sender="No sender!?"
        printf("SENDER=' (%s)'0, sender);
    }
    if (to == "") {
        to = "[Not sent to you]";
    }
    printf("TO=' %s'0, to);
}
!End!Awk!Script!
if [ $? != 0 -o ! -s $MSGDATA ]
then
    $ERROR 1>&2 $$SCRIPTNAME: Sorry - error while processing your message.
    $ERROR 1>&2 $$SCRIPTNAME: /tmp filesystem on '$HOSTNAME' probably full!
    $RM -f $MSGTMP $MSGTMP1 $MSGDATA $LOCKFILE
    exit 1
fi
#
# Instantiate the environment variable settings created by the awk script.
#
$RM -f $MSGDATA $MSGTMP1
#

```

```

# Rework address of sender, if necessary, for the mailer.
# Example field that causes the problem:
#   From: John Smith (UNIX Expert) <js'ith@nowhere>
# Mailer will try to reply to 'John', 'Smith', and 'jsmith@nowhere'
# due to input handling conventions. See routine 'skin' in
# 'src/ucb/Mail/aux.c' for details on how to strip such addresses.
#
# NOTE: Given a message with multiple senders on the 'From:' line
# (if that is even legal), the reworking here will reply only
# to one sender, with precedence to the rightmost sender in
# '<...>'s. But implementing fully correct RFC822 stripping
# would be difficult.
#
SENDER='$ECHO $SENDER | $SED -e 's/.*<>.*</>/'`
#
# Append incoming message to mail log.
#
# Note: It is up to us to ensure that the mail log file is formatted properly
# for the mail reader. That job is not done by the software that sent
# the outgoing mail, since such formatting is a local mail-handler
# property.
#
# In the case of the UNIX mail readers, proper formatting of a mail
# folder means that the 'message boundary' is defined as a blank line
# followed by a line starting with the word 'From' in the left margin.
# So we must ensure that:
#   - there is a blank line at the end of every message
#     in the log file,
#   - any lines beginning with the word 'From' in the left
#     margin, that are part of the body of the message, have
#     that word protected by prepending an '>'. This is the
#     conventional mechanism.
#
# Make sure that the mail log file will not be publicly readable so that
# incoming mail remains private.
#
$TOUCH $MAILLOG           # Ensure logfile exists.
$CHMOD 600 $MAILLOG      # Make private read-write only.
$SED -e "$ENVELOPE_END_LINE,/^From />From /" $MSGTMP >> $MAILLOG
$ECHO "" >> $MAILLOG     # Ensure message separator.
#
# Special cases:
# Ignore messages from certain 'people'.
#   MAILER-DAEMON, ---!MAILER-DAEMON, Mailer-Daemon, and the like.
# Ignore messages from answermail itself.
#
$ECHO "$SENDER" | $GREP -s -i "mailer-daemon"# Note pattern in lower case.
case $? in
0 )
  $RM -f $MSGTMP $LOCKFILE
  exit 0
;;
esac

```



```

case "$SUBJECT" in
"$ANSWERMAIL_SUBJECT_LINE" )
    $RM -f $MSGTMP $LOCKFILE
    exit 0
    ;;
esac
#
# Check for phone messages and send them to the backup contact person,
# as long as there is a backup contact person and the message
# did not come from that person in the first place.
#
$ECHO "$SUBJECT" | $GREP -s -i "phone" # Note pattern in lower case!
case $? in
0 )
    IS_PHONE_MESSAGE=true
    case "$PHONE_BACKUP_MAILADDRESS" in
    none )
        # Fall through.
        ;;
    **$SENDER** )
        # Do not forward a message from our backup contact person
        # back to him or her.
        #
        # Note match for 'pattern contained in', not 'exact'.
        # This handles cases where people add 'comments' to their
        # mail address, for example, their full name or title.
        #
        # This will fail to forward phone messages in the
        # case that someone else sending phone messages has the
        # backup contact person's address embedded in it.
        ;;
    * )
        ($ECHO This phone message has been logged, but ; \
        $ECHO please follow up on it if it looks urgent ;\
        $ECHO "";. \
        $ECHO " :unset record"; \
        $SED -e "1,/^d" -e "s/^/**> /" < $MSGTMP) |\
        $MAIL -s "Forwarded *** PHONE MESSAGE ***" \
            "$PHONE_BACKUP_MAILADDRESS"
        ;;
    esac
    ;;
1 )
    IS_PHONE_MESSAGE=false
    # Not obviously a phone message, so fall through.
    ;;
* )
    IS_PHONE_MESSAGE=false
    # grep error of some sort, just fall through.
    ;;
esac
#
# Now decide whether to notify the person

```

```

# who sent the message about your absence.
#
case "$SENDER" in
"No sender!?" )
    # If there was no 'From:' line then there is not a lot more we can do.
    $RM -f $MSGTMP $LOCKFILE
    exit 0
    ;;
esac
case "$NAMELIST" in
nobody )
    # Do not check whether the mail was really sent to us.
    ;;
* )
    case "$TO" in
    "[Not sent to you]" )
        # We must have been on a mailing list, or Bcc'ed.
        $RM -f $MSGTMP $LOCKFILE
        exit 0
        ;;
    * )
        # The mail was addressed to us, or we were copied.
        # - Fall Through -
        ;;
    esac
    ;;
esac
$TOUCH $HEARDFROM          # Make sure the list exists.
#
#
$GREP -s " $SENDER" $HEARDFROM    # Have we already notified?
case $? in
0 )
    # We have already sent a warning to the sender of
    # this message, so there is nothing more to do.
    ;;
1 )
    # This is the first message from this sender.
    # Add the sender to the log and send a return notice.
    $ECHO '$DATE' " $SENDER" >> $HEARDFROM
    case "$RETURN_DATE" in
    none )
        RETURN_MSG=""
        ;;
    * )
        RETURN_MSG="I'll be back on $RETURN_DATE."
        ;;
    esac
    case "$BACKUP_CONTACT_DETAIL" in
    none )
        BACKUP_MSG1=""
        BACKUP_MSG2=""
        ;;

```

```
* )
  BACKUP_MSG1="If your message is too urgent to wait, please contact"
  BACKUP_MSG2="$BACKUP_CONTACT_DETAIL."
  ;;
esac
$MAIL -s "$ANSWERMAIL_SUBJECT_LINE" "$SENDER" << !End!Mail! \
      >/dev/null 2>&1

:unset record
Hi. I'm away from the office for a few days.
$ABSENCE_MSG
$RETURN_MSG

The mail you just sent me concerning "$SUBJECT"
has been saved, and I'll read it when I return.
$BACKUP_MSG1
$BACKUP_MSG2

Thanks for your patience.

!End!Mail!
  ;; # NOTE: The blank line at the end of the message
  #   text shown above is necessary.
* )
  # There was some sort of error from grep.
  ;;
esac

$RM -f $MSGTMP $LOCKFILE
exit 0
```



Index

Special Characters

- .cshrc
 - at usage, 211
 - slow, 67
 - with interactive shell, 68
- .login, 67
- /dev
 - ownership, 54
- /etc/group
 - searches, 26
 - YP master server, 27
- /etc/hosts
 - INR, 51

A

- ACCELL
 - databases, 271
- address
 - device drivers, 195
- address mask, 74
- aliases
 - mail, 291
 - namestripes, 220
 - sendmail, 269
- AnswerLine, 5, 26, 67, 219, 291
- answermail
 - script, 321, 324
 - script installation, 321
- architecture
 - Prism, 287
- arrow keys
 - mapping, 265
- asm
 - with C source, 215
- assembly code
 - with C source, 215
- at
 - used with .cshrc, 211
- at (1)
 - answermail script, 321

B

- back-to-back packets, 245
- bind
 - port numbers, 213
- blocking
 - using select (), 62

- buffer
 - Ethernet, 245
- buffers
 - color frame, 276
- bug
 - 3/50 CPU board, 189
 - reporting, 206
- bugs
 - assembler, 83
 - Bourne shell, 140
 - bsc3270, 103
 - bscrje, 103
 - C compiler, 84
 - C shell, 140
 - cgi, 120
 - compiler library, 99
 - compiler utilities, 102
 - compilers, 83
 - Datacomm, 103
 - debugger, 90
 - demo, 123
 - diagnostics, 109
 - dna, 105
 - documentation, 111
 - formatter, 156
 - FORTRAN compiler, 93
 - gp, 123
 - graphics, 120
 - kernel, 128
 - lint, 100
 - LISP, 170
 - mail, 158
 - make, 158
 - Modula 2, 171
 - network, 135
 - network library, 135
 - network program, 137
 - nfs, 135
 - optimizer, 100
 - pixrect, 123
 - printer, 159
 - protocol, 138
 - shell, 140
 - sna3270, 108
 - SunAlis, 167
 - SunCORE, 124
 - SunINGRES, 168
 - SunUNIFY, 172

bugs, *continued*
 SunView, 142
 system administration, 148
 utilities, 156
 utility programs, 160
 uucp, 165
 vt100tool, 108
 yellow pages, 139
 Bulletin Board, 250

C

canvas
 colormaps, 282
 CDB
 errata, 296
 child processes
 dbxtool, 192
 PID, 192
 chip
 83586, 188
 client
 sample programs, 13
 stream socket, 12
 color, 275
 maps, 276
 compilers
 assembler bugs, 83
 bugs, 83
 C compiler bugs, 84
 compiler library bugs, 99
 debugger bugs, 90
 FORTRAN compiler bugs, 93
 lint bugs, 100
 optimizer bugs, 100
 utility bugs, 102
 controller
 Ethernet, 245
 corrections
 April TOM, 224
 routing, 296
 CPU
 multiple, 244
 CR LF
 end-of-line, 44
 Customer Software Services, 5, 39

D

DARPA, 73
 databases
 incompatible, 271
 Datacomm
 bsc3270 bugs, 103
 bscrje bugs, 103
 bugs, 103
 dna bugs, 105
 sna3270 bugs, 108
 vt100tool bugs, 108
 daylight savings time
 kernel, 24
 dbxtool
 child processes, 192
 dd (1)

dd (1), *continued*
 slow disk test, 263
 device drivers
 Consulting Services, 194
 device addresses, 195
 devices
 ones present, 301
 diagnostics
 bugs, 109
 disk
 slow test, 263
 disks
 size using mkfs, 267
 size using setup, 267
 DMA, 194
 documentation
 bugs, 111
 DoD, 73
 DST, 24
 Australia, 24
 Europe, 24
 rules table, 25
 dump
 nd1 partitions, 266
 with host names, 270
 DVMA, 194

E

end-of-line
 definitions, 44
 environment
 answermail variables, 322
 errata
 April TOM, 224
 May CDB, 296
 routing, 296
 errno
 EWOULDBLOCK, 64
 errors
 1e0, 21
 Ethernet
 back-to-back packets, 245
 buffer, 245
 controller, 245
 throughput, 246
 experiment
 answermail script, 321
 devices present, 301

F

fork ()
 child processes, 192
 ftime, 24

G

gateway, 74
 getpagesize (), 300
 gettimeofday, 24
 gettytab
 modem entries, 209
 GMT, 24

graphics

- bugs, 120
- cgi bugs, 120
- demo bugs, 123
- gp bugs, 123
- pixrect bugs, 123
- SunCORE bugs, 124

grpck

- YP map problems, 27

H

Hackers' Corner

- answermail script, 321
- devices present, 301
- memory size, 299
- survey, 239

hardware

- color frame buffers, 276

Hayes-Compatible, 219

host names

- with dump, 270
- with fdump, 270

hotline@sun.COM

- reporting bugs, 206

I

I/O

- sockets, 9

ie0 spurious interrupt

- SunOS 3.2, 187

incompatibility

- databases, 271

INR, 51

- requirements for, 53

K

kernel

- bugs, 128
- daylight savings time, 24
- swap space, 232
- time zones, 23

keys

- mapping, 265

L

LANCE, 21

- packets, 21

le0

- errors, 21

line speeds

- uucp, 214

LISP

- bugs, 170

localtime, 25

lpr

- flow control, 43

M

mail

- aliases, 291
- formats, 293
- pitfalls, 293

Mail Service, 250

manuals

- proprietary, 197

maps

- color, 276
- YP, 34

mask

- address, 74

memory

- size, 299
- SunAlis requirements, 251
- SunINGRES requirements, 259

mkfs

- disk sizes, 267

modems

- gettytab entries, 209

Modula 2

- bugs, 171

N

namestripes

- aliases, 220
- reprogramming, 27

naming convention

- read, 243
- transfer, 243
- write, 243

ND

- swap space, 229

ndl

- dumping partitions, 266

network

- bugs, 135
- library bugs, 135
- nfs bugs, 135
- program bugs, 137
- protocol bugs, 138
- yellow pages bugs, 139

newfs

- dumping partitions, 266

NFS

- partitions, 57

nodes

- multiple, 244

O

out-of-band data

- sockets, 9

P

packets

- back-to-back, 245
- LANCE, 21

panic: iechkcca, 187

partition

- calculating size, 230
- swap space, 229

partitions

- dumping ndl, 266
- read protection, 57

Personal AnswerLine, 5

PF keys

- PF keys, *continued*
 - mapping, 265
- physmem, 299
- PID
 - child processes, 192
- port number
 - assignment of, 213
- PostScript
 - pscat output, 198
 - setlinewidth, 208
- pounds sterling
 - symbol printing, 49
- Prism
 - windows, 287
- proprietary manuals, 197
- pscat
 - PostScript, 198
- ptroff
 - pounds sterling, 49
- pty
 - ownership, 54
- R**
- rdump
 - with host names, 270
- read
 - naming convention, 243
 - read optimization, 59
 - write permission, 59
- read protection
 - NFS, 57
- register
 - saving D2, 46
- release level
 - SunOS, 205
- reporting bugs, 206
- RETRN
 - end-of-line, 44
- root
 - file permissions, 57
 - read permissions, 57
- S**
- SCB, 188
- screendump
 - color windows, 288
- script
 - answermail, 324
- SCSI
 - slow disk test, 263
- seek
 - read optimization, 59
- select ()
 - exceptions, 64
 - non-blocking mode, 62
- sendmail
 - aliases, 269
- server
 - stream socket, 10
- setlinewidth, 208
- setup
 - continued*
 - disk sizes, 267
- shell
 - Bourne shell bugs, 140
 - bugs, 140
 - C shell bugs, 140
- SIGIO, 9
- SIGPIPE
 - server, 10
- SIGQUIT
 - server, 10
- SIGURG, 9
- sleep, 43
- sockets
 - example programs, 10
 - out-of-band data, 9, 15
 - programming examples, 9
 - servers, 10
- Software Information Services, 1, 39
- STB
 - duplication of, 181
- stdio
 - read optimization, 59
- subnets
 - address mask, 74
 - definition, 73
 - enabling, 77
 - Exterior Gateway Protocol, 73
 - limitations, 75
 - SunOS release 3.3, 264
- subnetting, 73
- sun!hotline
 - reporting bugs, 206
- sun!stb-editor, 26, 39, 67, 70, 181, 219, 291
- sun!sunbugs
 - reporting bugs, 206
- SunAlis
 - bugs, 167
 - memory requirements, 251
 - release 2.0, 249
 - windows, 252
- sunbugs@sun.COM
 - reporting bugs, 206
- SunCGI, 280
- SunCore, 282
- SunINGRES
 - bugs, 168
 - installing release 5.0, 258
 - memory requirements, 259
 - release 5.0, 254
- SunLink Internet Router, 51
 - requirements for, 53
- SunOS
 - determining release of, 205
 - release 3.3 and subnets, 264
- Suntools
 - exiting, 55
- suntools
 - frame buffers, 277
 - reprogramming namestripes, 27
- SunUNIFY

SunUNIFY, *continued*
 bugs, 172
 SunView
 bugs, 142
 color frame buffers, 278
 swap space
 ND, 229
 switcher (1)
 colormaps, 287
 system administration
 bugs, 148

T

tape verification, 210
 TCP
 sockets, 12
 telnet, 44
 Bridge terminal server, 44
 throughput
 Ethernet, 246
 time zones
 TZ, 23
 uucico, 23
 transfer
 naming convention, 243
 troff
 previewing output, 198
 tty
 ownership, 54
 TZ, 23
 DST rules table, 25

U

utilities
 bugs, 156
 formatter bugs, 156
 mail bugs, 158
 make bugs, 158
 printer bugs, 159
 utility program bugs, 160
 uucp bugs, 165
 yellow pages, 33
 uucico
 time zones, 23
 uucp
 Hayes-Compatible, 220
 line speeds, 214

V

variables
 answermail environment, 322
 verification
 tapes, 210
 vi
 maps, 69

W

windows, 276
 color frame buffers, 277
 Prism, 287
 with SunAlis, 252
 write

write, continued
 naming convention, 243
 write permission
 read, 59

X

X.25, 61

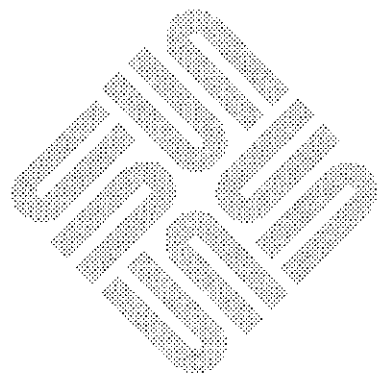
Y

yellow pages, 31
 installation, 32
 mail aliases, 291
 utilities list, 33
 YP, 31
 clients, 31
 domains, 32
 installation, 32
 maps, 34
 master server, 32
 rpc, 33
 server maps, 31
 slave servers, 31
 utilities list, 33
 ypbind, 32
 ypserv, 32



Revision History

<i>Revision</i>	<i>Date</i>	<i>Comments</i>
FINAL	July 1987	Sixth issue of Software Technical Bulletin (Software Information Services).







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

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

European Headquarters
Sun Microsystems Europe, Inc.
Sun House
31-41 Pembroke Broadway
Camberley
Surrey GU15 3XD
England
0276 62111
TLX 859017

Australia: 61-2-436-4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
The Netherlands: 02155 24888
UK: 0276.62111

Europe, Middle East, and Africa,
call European Headquarters:
0276.62111

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

