sun®
microsystems

Software Technical Bulletin

October 1987

Software Information Services

Part Number 812-8701-09
Issue 1987 – 9
October 1987

Software Technical Bulletin

October 1987

Software Information Services

Part Number 812-8701-09
Issue 1987 – 9
October 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 2-312, 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**.

Contents

1

NOTES & COMMENTS

1

NOTES & COMMENTS

Editor's Notes

Editor's Notes

The October editor's notes for the Software Technical Bulletin (STB) include an announcement of new SunOS 'Dot Dot' releases, the current Sun software products and release levels table, becoming a beta test site volunteer, announcement of a Sun Education email bulletin board, a note on STB editor email responses, and a console messages log program in "The Hackers' Corner".

SunOS 'Dot Dot' Releases

Sun Microsystems is now releasing bundled patches every two or three months. These releases are called 'Dot Dot' releases and appear between other SunOS releases. See the 'Dot Dot' release article in this *Notes and Comments* section for lists of fixes that appear and are available now in SunOS releases 3.4.1 and 3.4.2.

Current Sun Software Products and Release Levels Table

The October Software Technical Bulletin (STB) includes the current version table. The current release level is shown for each product.

Use this table along with STB articles that appear in one or two issues after a new current release is available for a particular product. You can then better determine what your software needs are, what functions are available in a new release, and whether the release you are using is down-level from the most current product release.

Beta Site Volunteers

Look to an article in this *Notes and Comments* section about becoming a beta site volunteer to test upcoming new releases of Sun software products. If you are interested, simply fill out the enclosed questionnaire. A product manager for the appropriate product will then contact you for further details.

Sun Education email Bulletin Board

Sun Educational Services has set up an email bulletin board facility for use by Sun customers. Look to the article in this section for details on course outlines, catalogs, and the like.

STB Editor Email Responses

Every effort is made to respond to each email message sent to the STB editor. However, you may not receive a direct reply in the case that your reply is returned to Sun by the network daemon. Please note, however, that each request for back copies, changes in mailing addresses, and subscription-related issues is forwarded to the appropriate Sun group.

The Hackers' Corner

This month's code comes from Sun Europe and is useful to those needing a log of console messages, both stored in a file and available to a printer.

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

Thanks.

The STB Editor

New SunOS 'Dot Dot' Releases

SunOS 'Dot Dot' Releases

Sun Microsystems is now releasing tapes containing bundled patches every two or three months, between other SunOS releases. These new releases are called 'Dot Dot' releases. Look to this article and future articles in the STB 'Notes and Comments' section that contain announcements of 'Dot Dot' releases, lists of specific fixes, fix reference numbers, and a synopsis of each corrected problem.

SunOS 'Dot Dot' Release Availability

SunOS 'Dot Dot' releases are available at no charge to Sun customers holding software support contracts, and to all Sun customers under warranty. Other Sun customers wishing to purchase a particular release may do so for $200 USD.

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

The first two releases available at this time are SunOS releases 3.4.1 and 3.4.2. Please note that these two dot dot releases can be installed separately or together on systems currently running SunOS release 3.4.

'Dot Dot' Ordering Information

Use the information below to order SunOS releases 3.4.1 or 3.4.2 or both.

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

SunOS Release 3.4.1

Description	CPU-type	Media Size	OMAR #	Unit Price
Docs, & Tape	68010	1/4"	DOT2-01-3.4.1	$200
Docs, & Tape	68010	1/2"	DOT2-02-3.4.1	$200
Docs, & Tape	68020	1/4"	DOT3-01-3.4.1	$200
Docs, & Tape	68020	1/2"	DOT3-02-3.4.1	$200

SunOS Release 3.4.2

Description	CPU-type	Media Size	OMAR #	Unit Price
Docs, & Tape	68010	1/4"	DOT2-01-3.4.2	$200
Docs, & Tape	68010	1/2"	DOT2-02-3.4.2	$200
Docs, & Tape	68020	1/4"	DOT3-01-3.4.2	$200
Docs, & Tape	68020	1/2"	DOT3-02-3.4.2	$200

SunOS Release 3.4.1

A list of SunOS 3.4.1 fixes, fix reference numbers, and a synopsis for each solved problem appears below.

- □ `blank`, Reference Number: 1004642

  Synopsis: `screenblank` allows the `-k` and `-m` options while in `suntools`.

- □ `cgi`, Reference Number: 1003572

  Synopsis: Bad `inquire_cell_array` and `inquire_pixel_array name` argument.

- □ `cgi`, Reference Number: 1003687

  Synopsis: The CGI Mouse cursor is always visible.

- □ `cgi`, Reference Number: 1004825

  Synopsis: `-lcgi` requires `-lsuntool` to compile a `cgi` program.

- □ `cgi`, Reference Number: 1005251

  Synopsis: `close_cgi_pw()` fails if no viewsurface is active.

- □ `cursor`, Reference Number: 1003864

  Synopsis: The crosshair cursor does not work when `CANVAS_FAST_MONO` is used.

- □ `fpa`, Reference Number: 1004500

  Synopsis: A program compiled using `-ffpa` causes an `FPA KERNEL BUS ERROR` to occur.

- □ `fsck`, Reference Number: 1003023

  Synopsis: The `fsck: HOLD BAD BLOCK` message is undocumented.

- □ `gp1`, Reference Number: 1004863

  Synopsis: This is a `GP1_PR_PGON_TEX` problem.

- □ `gp1`, Reference Number: 1004984

  Synopsis: `GP1_PR_ROP_TEX` semantics are wrong for a 1-bit deep `src`.

- ▫ `loopback`, Reference Number: 1005131

  Synopsis: The resolver has the wrong `loopback` address.

- ▫ `make`, Reference Number: 1003151

  Synopsis: `make` does not always build the objects that it should.

- ▫ `ping`, Reference Number: 1004791

  Synopsis: `ping` says machines are up even when they are not.

- ▫ `printer`, Reference Number: 1004074

  Synopsis: `lprm` causes line printer daemon to disappear.

- ▫ `rexd`, Reference Number: 1005140

  Synopsis: A `rexd` race condition occurs when mounting in `/tmp`.

- ▫ `scsi2`, Reference Number: 1004639

  Synopsis: This is a bug in the Sun-2 SCSI driver.

- ▫ `sendmail`, Reference Number: 1005042

  Synopsis: Yellow Page alias must use primary host names.

- ▫ `socket`, Reference Number: 1003135

  Synopsis: `panic: mfree` occurs with `AF_UNIX SOCK_STREAM` out-of-band (OOB) data.

- ▫ `suncore`, Reference Number: 1000895

  Synopsis: Transformation of text that does not clip.

- ▫ `sunpro`, Reference Number: 1004898

  Synopsis: The `install_sunpro` script fails for all configurations.

- ▫ `termcap`, Reference Number: 1004731

  Synopsis: `termcap` entry for `TERM=wy` breaks `initscr()`.

SunOS Release 3.4.2

A list of SunOS 3.4.2 fixes, fix reference numbers, and a synopsis for each solved problem appears below.

- □ dbx, Reference Number: 1003647

  Synopsis: Lexically recursive #includes confuse dbx

- □ dbx, Reference Number: 1004996

  Synopsis: dbx shows segmentation violation while stepiing

- □ diag, Reference Number: 1005466

  Synopsis: sysdiag's sptest fails with /dev/tty[a,b]; does not respond

- □ disk, Reference Number: 1005360

  Synopsis: SCSI disk driver hangs when ACB4000 reports write fault

- □ disk, Reference Number: 1005363

  Synopsis: Some SCSI MD21 (141 MB) errors cause system hang

- □ ether, Reference Number: 1006127

  Synopsis: Ethernet problems induced by bad ICMP address mask reply.

- □ io, Reference Number: 1005930

  Synopsis: physio bug causes writev(2V) failure

- □ io, Reference Number: 1001069

  Synopsis: bug in physio breaks readv

- □ kernel, Reference Number: 1006165

  Synopsis: sysdiag's softfp and mc68881 core dump (illegal instruction)

- □ line, Reference Number: 1004863

  Synopsis: This is a GP1_PR_PGON_TEX problem.

- □ line, Reference Number: 1004984

  Synopsis: GP1_PR_ROP_TEX semantics are wrong for a 1-bit deep src.

- □ line, Reference Number: 1005359

  Synopsis: Problem using pw_line and pw_polyline

- □ lockf, Reference Number: 1004336

  Synopsis: lockf() very slow

- □ look, Reference Number: 1003885

  Synopsis: look may dump core on long lines

- □ net, Reference Number: 1004765

  Synopsis: subnet broadcast address computed incorrectly

- □ nfs, Reference Number: 1005489

  Synopsis: NFS attribute cache functions incorrectly

- □ rpc, Reference Number: 1004739

  Synopsis: rpc.lockd fails to free, thus using excess memory

- □ sccs, Reference Number: 1003207

  Synopsis: SCCS uses delta times for diffs

- □ sccs, Reference Number: 1005438

  Synopsis: SCCS deledit duplicates random lines in a file

- □ scsi3, Reference Number: 1005366

  Synopsis: System panics when using ttya with SCSI3

- □ serial, Reference Number: 1006154

  Synopsis: system is flooded with zs interrupts on synca/b transitions

- □ sunpro, Reference Number: 1004598

  Synopsis: make does not handle square bracket characters in target filenames

- □ `sunpro`, seven unnumbered fixes

  Descriptions:

  1) No longer dumps core if the source needed to build a library member does not exist; instead reports "Don't know how to build x".

  2) Fixed the `-k` option so that it works for lists of targets given on the make command line.

  3) Remove the `.make.state` lock file if make is interrupted.

  4) Use the `varargs` mechanism for the error routines.

  5) Fixed bug that caused very long command lines to be read incorrectly.

  6) Fixed bug that caused `$${X}.il` to be read incorrectly when used as a dependency.

  7) Made it possible to undefine default suffix rules from the user's makefile.

- □ `tape`, Reference Number: 1004559

  Synopsis: UNIX hangs while booting if `xt` controller has on-line drive

- □ `transfer`, Reference Number: 1006132

  Synopsis: TCP/IP file transfer using `ftp` hangs/stops when using 3.4

Sun Software Produce Releases

Current Software Sun Products and Release Levels

Product Name	Current Release
SunOS	3.4
Cross Compiler	1.0
SunLink BSC3270	4.0
SunLink Local 3270	4.0
SunLink SNA3270	5.0
SunLink IR	5.0
SunLink DDN	5.0
SunLink DNI	4.0
SunLink OSI	5.0
SunLink MCP	5.0
SunLink TE100	4.0
SunLink X.25	5.0
FORTRAN	1.0
NeWS	1.0
Sun Common Lisp	2.0
Modula-2	1.0
SunAlis	2.1
SunGKS	2.0
SunINGRES	5.0
SunSimplify	1.0
SunUNIFY	2.0
Transcript	2.0
SunIPC	1.1
PC-NFS	2.0

Current Sun Software Products and Release Levels

The table appearing above contains a list of current Sun software products and their respective current release levels.

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

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

This table appears monthly in the STB for your convenience.

Becoming a Sun Beta Site

Beta Sites for Sun Software Products

This article contains an introduction, overview of beta site benefits and risks, software testing activities associated with beta sites, a description of beta site contact persons, a list of Sun software product categories, and a survey questionnaire for those Sun customers interested in further beta site information or in volunteering to become a beta site for future releases of Sun software products.

Introduction

Software products go through several phases within manufacturing companies before and after becoming available for purchase. These phases, including beta site testing, are listed below.

- □ Initial conceptualization
- □ Market analysis
- □ Development and debugging
- □ Alpha site testing (internal)
- □ Beta site testing (internal and customer)
- □ Released, First Customer Ship (FCS)
- □ Subsequent release levels
- □ End-of-life, support ends

Of course, the goal is to ship a perfect product every time. The purpose of Sun beta testing is to test debugged code in order to release a thoroughly tested First Customer Ship (FCS) product. However, product complexity requires testing in many environments to discover applications where actual use reveals product areas needing further development. This is where beta site testing can be of advantage to both the manufacturer and the customer.

Beta Sites: An Overview

Sun Microsystems is interested in developing beta sites at customers' places of business as one of the several testing phases of Sun software product development.

Being a beta site really means a new relationship between the customer and Sun, plus additional understanding of software product applications. Part of this relationship includes testing of new products and a written non-disclosure agreement for beta site testing.

There are both benefits and risks in such testing, and both need to be considered.

Benefits!

As a beta testing site for Sun software products, you will be among the first to gain product knowledge and to know how applicable a particular Sun software product is to your ongoing production or manufacturing needs. (Beta software is not intended for use in developing 'ongoing, current, and immediate' products.)

Your own product planning and development processes can be done in light of *hands on* experience with Sun software products.

Your employees will be trained through on-the-job use of a software product that may subsequently become part of their work processes.

You will get first-hand knowledge of how the product behaves, solving actual end-user problems. Ease of use, operator guides, and development of user notes and procedures all may result. *Seeing is believing* and *show me* often allow better understanding of a product's capabilities.

Beta site testing allows input to Sun developers since product design includes flexibility to include changes. This is the case both at the beta site phase for first-time products, as well as including changes in subsequent releases following FCS. Flexibility exists to include minor changes from beta testing before FCS, with more extensive design and development scheduled for subsequent releases.

Risks!

Customers interested in becoming beta test sites for particular Sun software products may minimize risks associated with testing unreleased products. Approached with common sense, the risks can be minimized and are outweighed by the mutual benefits.

A beta site is a test site for an unreleased product. Customers interested in testing need to have enough resources to use the unreleased product in *test* mode. Beta sites are not for those needing early, pre-release versions of a working product.

End-users need to know that the product is being *tested*, outside the usual and ongoing production or manufacturing processes at the customer site. The ideal test is exercising all features to the extent possible, not a limited use in a routine sequence, and yet not impacting ongoing production processes.

Finally, the benefit from beta site testing will be known for sure only *after* the testing. However, weighed against the greater benefits, many customers choose to take the risks to gain the benefits.

Beta Site Testing Activities

Beta site testing is best done by getting as many customer points of view as possible. Managers, project leaders, software developers, and end-users all have important feedback to give in beta sites.

Tests need to be done based on a mutually agreed upon schedule, both within the customer's ongoing production or manufacturing schedule as well as within Sun's product development and release schedule. The resources for beta testing need to be anticipated and allocated in advance of each test period. This would be done after follow-up discussions to the survey questionnaire at the end of this article.

The test feedback is best when specific, clear comments are made for a particular feature or application. The feedback is best summarized in writing, allowing Sun program managers and developers an opportunity to develop follow-up responses, questions, and product improvements.

Specific beta site activities begin with choosing a particular product with potential application in the customer's ongoing production or manufacturing processes. See the table in this article for a list of Sun software products and current release levels.

Beta Site Contact Person

Sun software product managers and customer managers will each designate a primary, beta site contact person. These two people will act as liaisons to whom technical information and test results will be directed. The contact person from Sun will be someone assigned to a technical position who can effect technical change.

This will not be a sales-oriented contact, but one of a necessary two-way exchange of technical information. One of the points that will be covered is a test agreement to be signed prior to testing.

Current Software Sun Product Categories

General categories of Sun software products are listed below. List one or more of these categories on your beta site interest questionnaire which appears in the next paragraph.

- □ SunOS
- □ Graphics

□ Window Environments

□ Communication

□ Network Products

□ Languages (Specify C, FORTRAN, Pascal)

□ OA Products

□ Development Environments

□ PC-Based Products

Beta Site Interest Questionnaire

If you are interested in further information about becoming a beta test site for a product listed in the table above, please fill out the questionnaire at the end of this article. The appropriate Sun product manager will then contact you regarding further, detailed information.

Fill out the beta site interest questionnaire and send it to the address shown below.

```
Software Technical Bulletins
Sun Microsystems, Inc.
2550 Garcia Avenue
M/S 2-312
Mountain View, CA  94043
```

You can also send your response by electronic mail to *sun!stb-editor*. Your response will result in a follow up contact from the appropriate product manager for more detailed discussions on beta site testing, planning, scheduling, and the like.

As a beta site, Sun will loan you the product for test installation and use by technically qualified managers, project leaders, or end-users.

Name ______________________________

Position:

_____ End-User
_____ Manager
_____ Product Manager
_____ Project Leader
_____ Other

Company Name ______________________________

Company Address ______________________________

E-Mail Address ______________________________

Phone Number ______________________________

Product(s) ______________________________

Type of Industry:

_____ Aerospace
_____ Academic
_____ Automotive
_____ Civil A/E
_____ Defense
_____ Financial
_____ Automotive
_____ Manufacturing
_____ Oil & Gas
_____ Pharmaceutical
_____ Power Utilities
_____ Semiconductor
_____ Software

Sun Workstations:

Type ______________________________

Configuration ______________________________

SunOS Release ______________________________

Sun Product Categories of Interest:

_____ Automotive
_____ Communication
_____ Development Environments
_____ Graphics
_____ Languages (Specify C, FORTRAN, Pascal)
_____ Network Products
_____ QA Products
_____ PC-Based Products
_____ SunOS
_____ Window Environments

Sun Education e-mail Board

Sun Educational Services email Bulletin Board

Sun Educational Services, located in Milpitas, California, has set up an 'email Bulletin Board' facility for Sun Microsystems customers.

This facility provides a means for interested customers to learn about new courses and related developments happening within Educational Services. This facility is also a good means for customers to direct questions and general requests for information to Educational Services. These questions and requests might be regarding course outlines, catalogs, how to arrange for a dedicated class at the customer's site, and so on.

How to Be Added to the email Bulletin Board

Customers who want to be added to the Education Services email Bulletin Board should send their Usenet or DARPA addresses to:

```
customer-training@sun.com
suncustomer-training
```

To ask questions, simply send the question to one of the above email addresses.

The New Educational Services Course Brochure

Sun Microsystems customers should soon be receiving the new Sun Educational Services course brochure. The brochure contains complete course descriptions as well as the course offerings scheduled from January, 1988 through June, 1988.

If you do not receive this brochure by November 10,1987, please contact your local sales office or Sun Educational Services directly at either of the toll-free numbers listed below.

In California:	800-423-8020
Elsewhere in the continental U.S.:	800-422-8020

Sun customers in the United Kingdom and in Europe should contact their local service center or sales representative.

Errata

Errata: Unsupported `asm` Usage

A short subject on page 215 of the June STB recommends using `asm` to embed assembly code in a C source file. Please delete this short subject since this is an unsupported mechanism. This causes numerous problems, especially when used with the optimizer.

A Supported Mechanism

A supported mechanism uses assembly-level, inline expansion. This is fully documented in Appendix G of the *Floating-Point Programmer's Guide for the Sun Workstation*, part number 800-1552.

You can use a simple, inline expansion facility to integrate assembly routines into higher-level C, Pascal, or FORTRAN routines. The peephole optimizer `c2` has been modified to accept code containing inline-expanded routines.

The C, FORTRAN, and Pascal compilers assume that file names ending with the `.il` suffix contain inline-expandable assembly routines. Several examples are provided to illustrate proper uses of the `/usr/lib/inline` program.

2

ARTICLES

2

ARTICLES

Adding 2400 Baud Modems

Using 2400 Baud Modems with UUCP

Higher-speed modems are becoming more cost effective in data transfer due to increasing phone line costs and decreasing modem prices. This article explains how to modify the appropriate files on a Sun workstation to enable higher-speed modems to be used with UUCP.

Overview

Once the modems have been properly attached to the workstation, the SunOS and UUCP software need to be modified in order to use the faster transmission rates. An overview of the steps required to add higher-speed capabilities to your workstation is shown below.

□ Edit `/etc/gettytab` to add the new speeds into the 'Fast dialup terminals' definition

□ Edit `/usr/lib/uucp/L-devices` to define the new line speeds to UUCP

□ Edit `/usr/lib/uucp/L.sys` to modify the remote site modem speed

□ Edit `/etc/ttys` to modify the initial line speed setting

□ Reinitialize `/etc/init` to use the new line speed settings

These steps are detailed below. See the references at the end of this article for additional, detailed information.

Adding 2400 Baud Capability: A Procedure

Use the following procedure to add 2400 baud or other higher-speed capabilities to your Sun workstation. Each step includes examples of typical files before and after the file modifications. The information applies to SunOS release 3.2 and subsequent releases.

Step 1: Login

Login as superuser to modify the system and UUCP files.

Step 2: Modify `/etc/gettytab`

The `/etc/gettytab` file defines the terminal line characteristics. You need to add a 2400 baud entry to the 'Fast dialup terminals' definition.

In the standard SunOS release of UNIX, this definition is set to start at 1200 baud and then to cycle down to 300 baud if needed. The standard definition will then cycle back to 1200 baud if needed. In order to add the new 2400 baud speed into this loop, it is necessary to add a new entry into this cycle. Modifying the `/etc/gettytab` file, as shown below, will add the 2400 baud capability.

Original File:

```
#
# Fast dialup terminals, 1200/300 rotary (can start either way)
#
3|D1200|Fast-Dial-1200:\
    :nx=D300:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
    :nx=D1200:tc=300-baud:
```

Modified File:

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start any way)
#
3|D1200|Fast-Dial-1200:\
    :nx=D2400:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
    :nx=D1200:tc=300-baud:
J|D2400|Fast-Dial-2400:\
    :nx=D300:tc=2400-baud:
```

Note that the last two lines, the one starting with 'J' and its continuation, have been added to the definition. In the second line, the entry 'nx=D300' has been changed to 'nx=D2400' to force the line speed to jump from 1200 to 2400 baud if a 'break' condition is generated by a dialed-up terminal or equivalent device. Should the 2400 baud test unsuccessfully, the line will skip to 300 baud. Similar changes may be made to add 4800 or 9600 baud capabilities.

Also, note that you may wish to remove the 300 baud entry from the cycle since 300-baud modems are used in fewer and fewer applications. Do this by changing 'nx=D300' to 'nx=D1200' in the 'D2400' definition.

The terminal line speed may new be cycled from 2400 to 1200 baud and back, normally by pressing the 'break' key of most terminals. Your modified `/etc/gettytab` file should now look like the example shown below.

Modified File (reduced 300 baud cycling capability):

```
#
# Fast dialup terminals, 2400/1200 rotary (can start any way)
#
3|D1200|Fast-Dial-1200:\
    :nx=D2400:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
    :nx=D1200:tc=300-baud:
J|D2400|Fast-Dial-2400:\
    :nx=D1200:tc=2400-baud:
```

Step 3: Modify `/usr/lib/uucp/L-devices`

Now that the new baud rate has been added, you need to get UUCP to use it. To add 2400 baud capabilities to UUCP, modify the `/usr/lib/uucp/L-devices` file by adding definitions for the new speed. A standard file set to use 1200 baud with two modems might look like the example shown below.

Original File:

```
ACUHAYES cua1 cua1 1200
ACUHAYES cua0 cua0 1200
```

The example file, after being modified for 2400 baud, now looks like the following example. This example allows the both modem ports to start at either 1200 or 2400 baud.

Modified File:

```
ACUHAYES cua1 cua1 1200
ACUHAYES cua1 cua1 2400
ACUHAYES cua0 cua0 1200
ACUHAYES cua0 cua0 2400
```

Step 4: Modify `/usr/lib/uucp/L.sys`

Modify the `/usr/lib/uucp/L.sys` file by changing the modem speed definition. UUCP will now call sites at the new 2400 baud rate whenever either dialer is available at the time it tries to dial out. A typical `/usr/lib/uucp/L.sys` entry is given below.

Original File:

```
sun Any ACUHAYES 1200 5551212 ogin: Usun ssword: stbuucp
```

You need to change the modem speed, the fourth field from the left, from '1200' to '2400'. This change causes UUCP to call the specified site at 2400 baud automatically. Your modified entry should look like the example shown below.

Modified File:

```
sun Any ACUHAYES 2400 5551212 ogin: Usun ssword: stbuucp
```

Repeat this step as needed to call other systems at other sites at the 2400 baud rate.

Step 5: Modify /etc/ttys

Since a 'J' entry was added to the /etc/gettytab file in step 2, line speed initialization field(s) in the /etc/ttys file may now include a 'J' in the second column to allow dialins at 2400 baud. Make this change if most of the modems calling your system do so at 2400 baud. A standard /etc/ttys file contains entries similar to those shown below.

Original File:

```
13ttyd0
13ttyd1
```

These entries are the definitions for the dialup lines. To enable the modem to answer at 2400 baud, change the '3' to 'J' in each line. Your modified file should now look like the sample shown below.

Modified File:

```
1Jttyd0
1Jttyd1
```

Step 6: Reinitialize /etc/init

The /etc/init process must re-read the /etc/ttys file to use the new initialization. You can reinitialize /etc/init by issuing it a SIGHUP.

The /etc/init process is the first process started when the system is booted, and therefore has a process id of '1'. Issuing a SIGHUP to /etc/init causes it to reread /etc/ttys and to reset all of the changed parameters. Use the following command to accomplish this.

```
# kill -HUP 1
```

This completes the procedure to add 2400 baud rates to reduce the cost of phone line service by using faster modems. Again, note that similar changes may be made to include the 4800 or 9600 baud rates as well. Please note that UUCP has a maximum speed of 4800 baud, while the Sun workstation has 9600 and 19.2 kbaud capability. The highest useful rate is thus limited to 4800 baud.

Summary

You have now added the 2400 baud definition to all of the appropriate system and UUCP files. The system and UUCP can now use the new transmission rate. Not only will UUCP dial and receive calls at 2400 baud, but users can now use 2400 baud modems for dialin as well. This will cut the data transfer transmission time and improve interactive response.

References

See the following two sections of *System Administration for the Sun Workstation*, part number 800-1323, for more information on the UUCP file changes.

- Chapter 5.4, 'Adding a Modem to Your System'
- Appendix C, 'UUCP Implementation Description'

See the following two manual pages for more information on the system file changes and procedural steps.

- `gettytab(5)`
- `ttys(5)`
- `init(8)`

SCLISP Quick Check

Sun Common Lisp Quick Check

The following interactive sequence provides the system administrator with a means to quickly and easily check a newly-installed Sun Common Lisp (SCLISP) development environment.

Functions Performed by the Quick Check

The functions performed by this quick check, namely garbage collection (which reorganizes the dynamic memory) and computation with large integers, both give SCLISP considerable exercise, and can be performed very easily.

The object of this check is to ensure that the computations are performed without aborting or returning error messages, rather than to verify the correctness of the numerical values produced. The values returned by Garbage Collection (GC) will differ from system to system. Note that the factorial function definition in the interactive sequence below omits type checking.

Running the Quick Check

To run the quick check, enter Lisp. After the banner text is displayed, the SCLISP > prompt appears. If the `Editor` subsystem is present, the interactive sequence can be performed in the `*Lisp Buffer*` window of the `Editor`. To enter the `Editor`, enter **(ed)** at the prompt, then perform the interactive sequence as shown below from within the `*Lisp Buffer*` window.

To run the sequence, enter the information shown in **`bold`** following the SCLISP > prompt.

In the system responses shown below, note that carriage returns have been inserted to break the large integers into two lines. This prevents truncation during printing.

```
;;;;  (SCLISP banner text)

> t
T
> (defun fact (n)
    (cond ((zerop n) 1)
          (t (* n (fact (1- n))))))
FACT
> (fact 100)
9332621544394415268169923885626670049071596826438162146859296389521759999322991
5608941463976156518286253697920827223758251185210916864000000000000000000000000
> (compile 'fact)
;;; Compiling function FACT...tail merging...assembling...emitting...done.
FACT
> (fact 100)
9332621544394415268169923885626670049071596826438162146859296389521759999322991
5608941463976156518286253697920827223758251185210916864000000000000000000000000
> (gc)
;;; GC: 8546 words [34184 bytes] of dynamic storage in use.
;;; 122274 words [489096 bytes] of free storage available before a GC.
;;; 253094 words [1012376 bytes] of free storage available if GC is disabled.
34184
489096
1012376
> (quit)
```

Additional Checking

Once the interactive sequence has successfully run, the user can continue with more extensive testing. For example, if the `windows` and `flavors` systems are both present, playing a game of `hexapawn` is a good test, since it exercises both systems.

To play `hexapawn`, change to the `goodies` subdirectory, then enter Lisp. At the prompt, enter **`(load "load-flavors-hexapawn")`**, then follow the instructions displayed in the window. To exit `hexapawn`, select `quit` from the window's SunView frame menu.

Incidentally, the `hexapawn` source code is provided in the `goodies` directory as a sample of object-oriented SCLISP programming involving windows.

FPA and Recomputation

Floating-Point Accelerator Performance Description Enhancement

The following is an enhancement to the description of Floating-Point Acceleration (FPA) beginning on page 26 of the *Floating-Point Programmer's Guide for the Sun Workstation*, part number 800-1552-10.

Performance

Because the treatment of floating-point exceptions is so variable, most floating-point programs that run reliably on machines with different types of arithmetic do not cause any floating-point exceptions to occur, other than in an inexact case. Consequently, Sun's `FPA` recomputation scheme was designed without consideration of efficiency, other than in the inexact case. Programs that generate many exceptions may run extremely slowly -- slower than software floating-point -- due to the intervention of the operating system to handle bus errors and signaling. The most likely instances where this might arise are in computations which frequently underflow, or which handle many 'Not a Number' (NaN) circumstances.

The most common symptom indicating that frequent recomputation may be occurring is that the proportion of system time to user time consumed by a process is far greater than normal. If frequent recomputation is suspected, it may be investigated by running the program under the control of `adb` or `dbx`, then examining `WSTATUS` register bits 8 to 11 after each recomputation request. Bits 8 to 11 of the `WSTATUS` register correspond to the Weitek 1164/1165 S+ bus output status codes as shown in the table below.

**WEITEK 1164/1165 S+ BUS
EXCEPTION STATUS CODES**

Code	Exception Status
0	Exact zero result
1	Exact infinite result
3	Exact finite result
4	Not used
5	Overflow and inexact
6	Underflow
7	Underflow and inexact
8	Operand A subnormal
9	Operand B subnormal
10	Both operands subnormal
11	Division by zero
12	Operand A is `NaN`
13	Operand B is `NaN`
14	Both operands `NaN`
15	IEEE invalid operation

Interaction Examples

The interactions shown in the examples below demonstrate some typical results of suspected frequent recomputation investigations using `adb.a`.

Example One: In the example below, the '3' in `wstat = 2300` indicates inexact result.

```
host% adb.a out

:r
SIGFPE 8: numerical exception, CHK, or TRAP
stopped at       _MAIN_+0x48:            cmpl       #0x989680,d7
$x
mode3 = 2 wstat = 2300 state = 11 imask = 1 ldptr = 0 ierr = 20
```

The `FPA` always requests recomputation on the first inexact result.

Example Two: In the example below, the '7' in `wstat = 2700` indicates an inexact underflow.

```
host% adb.a out

:c
SIGFPE 8: numerical exception, CHK, or TRAP
stopped at       _MAIN_+0x48:            cmpl       #0x989680,d7
$x
mode3 = 2 wstat = 2700 state = 11 imask = 0 ldptr = 0 ierr = 20
```

The result of recomputation will be a subnormal number.

Example Three: In the example below, the '8' in `wstat = 2800` indicates a subnormal operand.

```
host% adb.a out

:c
SIGFPE 8: numerical exception, CHK, or TRAP
stopped at       _MAIN_+0x3c:            fpmul3s fpa5,fpa4,fpa4
$x
mode3 = 2 wstat = 2800 state = 11 imask = 0 ldptr = 0 ierr = 20
```

Recomputation will treat the operand according to its current nonzero value. If the 'fast' Weitek mode had been selected, `mode3 = 3` and its operand would have been treated as a zero, without generating this exception.

If recomputation performance is an issue, it can be partially ameliorated by using the Weitek chips in 'fast' mode. 'Fast' mode, although inconsistent with the IEEE standard, causes no ill effects on programs that work correctly on a variety of non-IEEE machines. This is because non-IEEE machines most likely underflow abruptly to zero, without creating subnormal (denormalized) numbers as IEEE machines do. In the normal 'IEEE' mode of Weitek chips, recomputation is invoked if either the operand or the result is a subnormal number. In the 'fast' mode of Weitek chips, recomputation is only invoked if the result would be a subnormal number. In this case, subnormal operands are treated as zero without generating any exception.

SunUnify Configurations

SunUnify Configuration Considerations

This article describes how to properly configure a SunUnify installation so that the following features are set up for correct operation.

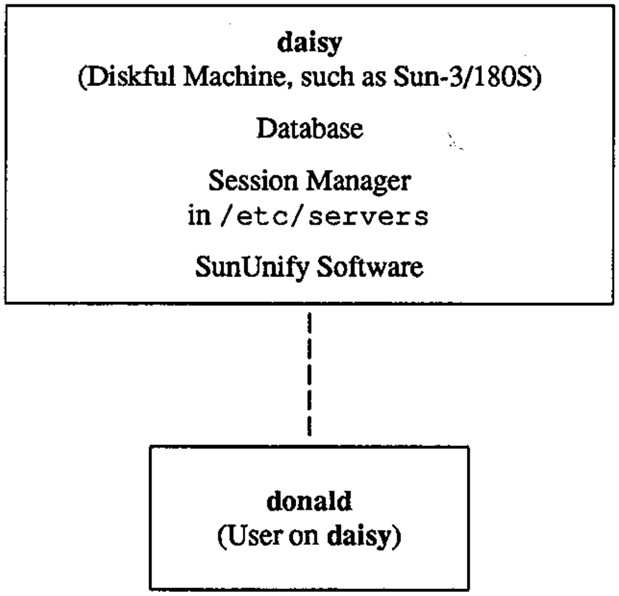
- □ SunUnify database
- □ Data manager
- □ Session manager
- □ The SunUnify Software including
  - executables
  - libraries
  - `include` files
  - `compat` directory

The most important aspect to keep in mind when configuring a SunUnify installation is that the SunUnify session manager *must* run on the same machine as the database. Three common Sun hardware configurations are illustrated in the following pages.

A Diskful Machine Configuration

When SunUnify is configured for a diskful machine, such as a Sun-3/180S, the SunUnify database, the session manager, and the SunUnify software all reside on one machine.

In the example below, the diskful machine named **daisy** is configured for the user named **donald** to properly use SunUnify.



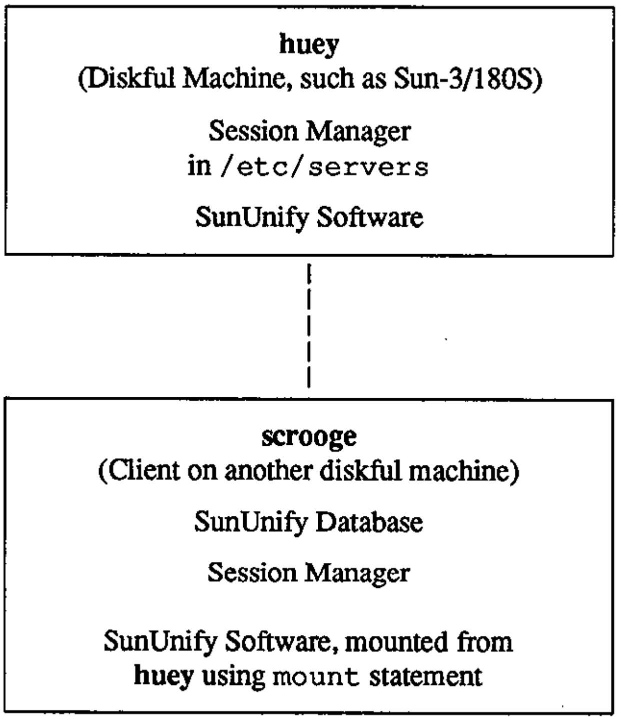
The session manager entry in `/etc/servers` takes the form shown below.

```
rcp tcp /<path_name>/unify/bin/dbsessionmgr 100016 3
```

Configuring a User on Another Diskful Machine

If SunUnify is being configured for a diskful machine, such as a Sun3-160, where the SunUnify software is NFS mounted and the database resides locally, the correct `mount` entry must appear in `/etc/fstab`. The entry for the session manager must appear in `/etc/servers`.

In the example below, the machine named **scrooge** has the SunUnify database on its local disk. The session manager is running on **scrooge**, and the necessary SunUnify libraries and executables are mounted from **huey**, as shown below.


huey
(Diskful Machine, such as Sun-3/180S)
Session Manager
in /etc/servers
SunUnify Software
scrooge
(Client on another diskful machine)
SunUnify Database
Session Manager
SunUnify Software, mounted from
huey using mount statement

The `mount` statement on **scrooge** takes the form shown below.

```
huey:/<path_name>/unify /<new_path_name>/unify nfs rw,hard,bg 0 0
```

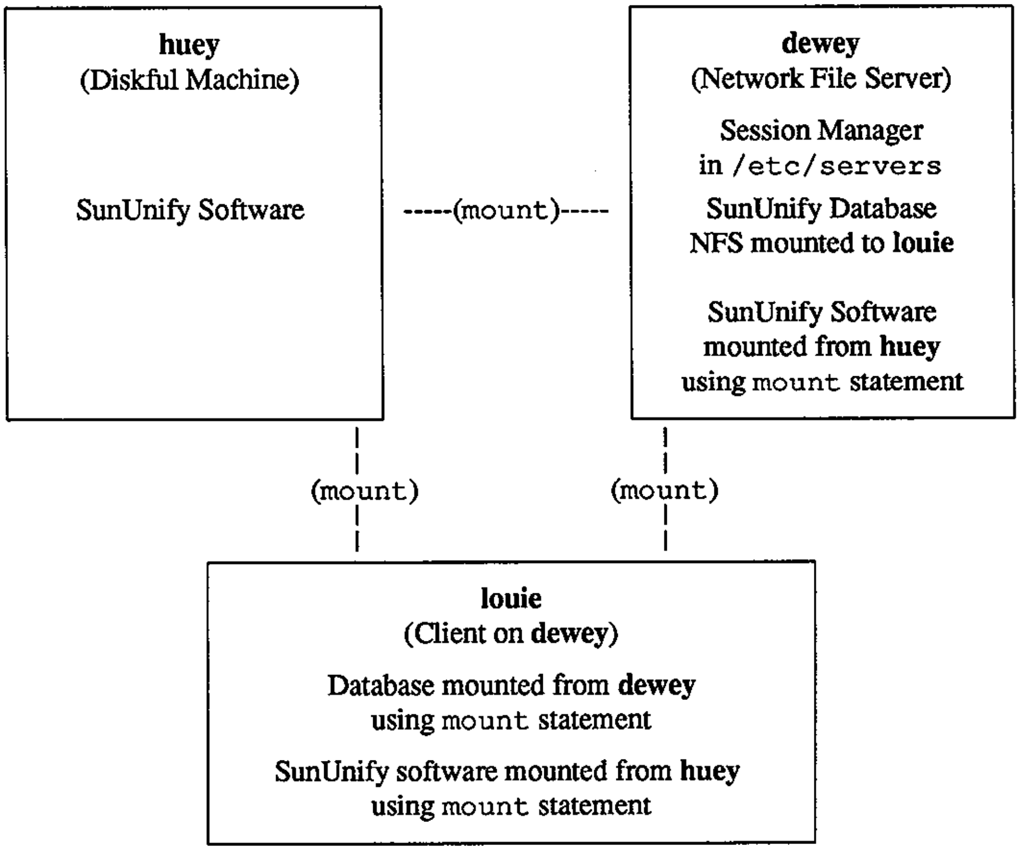
The session manager entry in `/etc/servers` takes the form shown below.

```
rcp tcp /<path_name>/unify/bin/dbsessionmgr 100016 3
```

A Diskless Client Configuration

When configuring a diskless client such as Sun-3/50, the database resides in the user's NFS-mounted home directory. The SunUnify software is mounted from the diskful machine to the diskless client and the client's nfs server. The session manager runs on the nfs server.

In the example below, **louie** is a diskless client on **dewey**. The database resides in the user's home directory, which i s NFS mounted from **dewey**. The SunUnify software is mounted on both **louie** and **dewey** from **huey**, and the database session manager is running on **dewey**, as shown below.



The mount statements on **louie** take the form shown below.

```
huey:/<path_name>/unify /<new_path_name>/unify nfs rw,hard,bg 0 0

dewey:/<path_name>/<home_directory> /<new_path_name>/<home_directory> nfs rw hard bg 0 0
```

The session manager entry in **dewey**'s `/etc/servers` takes the form shown below.

```
rcp tcp /<path_name>/unify/bin/dbsessionmgr 100016 3
```

FORTRAN 1.0 Announcement

Sun FORTRAN 1.0

This article is a brief overview of VMS-compatible Sun FORTRAN 1.0, including new features, functions, and usage considerations.

Introduction

Sun FORTRAN 1.0 is an enhanced ANSI FORTRAN 77 development system. It is Government Services Administration (GSA)-certified with VAX/VMS FORTRAN 4.0 extensions, thus providing a development system for a significantly expanded body of FORTRAN source code. Existing VMS FORTRAN applications can be ported to the Sun workstation environment. Through SunLink DNI, users can develop the same FORTRAN programs on both Sun and DEC/VAX systems with only minor modifications.

The package consists of three components: extensions to the `f77` compiler to support most of the VMS FORTRAN features; extensions to the debugger to support some of these same VMS FORTRAN features; and the `f77cvt` Source-Code Converter program, used to convert most of the remaining extensions into statements t hat the Sun FORTRAN compiler will accept. When used together, the compiler and converter provide almost total compatibility with VMS FORTRAN.

Sun FORTRAN 1.0 is designed to be compatible with Sun Operating System (SunOS) release 3.2 and release 3.4 for Sun2 and Sun3 systems.

Sun FORTRAN Compiler Extensions

The new compiler consists of the previous Sun FORTRAN compiler, `f77`, and several language extensions. These extensions have been separated into the three groups listed below.

- Group I: Direct compiler, run-time, library, and debugger extensions to Sun FORTRAN
- Group II: Extensions that are converted from VMS FORTRAN to Sun FORTRAN
- Group III: Unimplemented extensions from VMS FORTRAN, which are diagnosed by the source code translator

Each of these groups is discussed in detail below.

Group I: Direct Sun FORTRAN Extensions

Group I consists of direct extensions to Sun FORTRAN to provide general enhancements as well as VMS-specific features. These direct extensions include compiler, run-time, library, and `dbx` debugger extensions. These extensions provide additional facilities for FORTRAN program developers, and significantly decrease the time required to port applications to the Sun environment. The most significant extensions are listed below.

Compiler, Run-time, and Library Extensions

The extensions for the compiler, run-time, and library are listed below.

- Namelist-directed I/O
- `DO/END DO` and `DO WHILE` statements
- `BYTE` datatype, logical operations on integers, and arithmetic operations on logicals
- Long identifier names (up to 32 characters)
- Long source lines (up to 132 characters)
- Identifiers containing '_' and '$'
- `STRUCTURE, RECORD, UNION,` and `MAP` declarations
- Additional field and edit descriptors for `FORMAT` statements, and default field indicators for certain field descriptors
- `POINTER` datatype, which provides increased compatibility with vectorized FORTRAN versions
- Intrinsic functions in `DATA` and `PARAMETER` statements, such as `AND, OR, NOT, XOR, LSHIFT, RSHIFT, LGE, LGT, LLE, LLT`
- Improved error messages and error recovery
- Run-time generation of upper-case strings
- Flagging of non-ANSI constructs under the control of a compiler option

`dbx` *Debugger Extensions*

The extensions for the `dbx` debugger are listed below.

- `STRUCTURE, RECORD, UNION,` and `MAP` declarations
- `POINTER` datatype
- Improved error messages and error recovery

Group II: Converted VMS FORTRAN Extensions

The `f77cvt` Source Code Converter program accepts valid VMS FORTRAN code and outputs code that is compatible with Sun FORTRAN. In addition to the direct translations and extensions in Group I, the converter program handles many indirect conversions. In these instances, the VMS FORTRAN extensions have an equivalent, but not identical, Sun FORTRAN extension. These indirect extensions are listed below.

- Embedded comments
- Debugging statements
- VMS tab format
- Alternate syntax for direct-access I/O
- Radix-50 constants
- Typeless hex and octal constants
- `%VAL()` and `%LOC()` functions
- Non-standard length specifiers in `FUNCTION` statements
- `REAL*16` constants and declarations, which are converted to `REAL*8`
- Initialization of variables in `COMMON` blocks
- Non-Standard `PARAMETER` statement
- `DATA` statements mixed with specification statements
- `ENCODE` and `DECODE` statements
- `VIRTUAL` statement
- `IMPLICIT NONE` statement
- Variable initialization in declaration statements
- Implicit type conversion from `REAL` to `INTEGER`
- `TYPE` and `ACCEPT` statements
- Options to `OPEN: TYPE, NAME`, and `RECORDSIZE`
- Options to `CLOSE: DISP` and `DISPOSE`
- Non-`CHARACTER` run-time format and file specifiers
- Non-Standard alternate return actual arguments
- Omitted actual arguments

- %REF() function with non-CHARACTER argument
- Flagging of unsupported VMS extensions

Group III: Unsupported VMS Extensions

Some VMS FORTRAN extensions cannot be transferred or converted by the f77cvt Source-Code Translator into Sun FORTRAN. In these instances, the unsupported extensions will be identified and diagnosed by the translator for quick identification and resolution once the porting process is complete. These unsupported VMS FORTRAN extensions are listed below.

- DEFINE FILE, FIND, DELETE, REWRITE, and UNLOCK statements
- OPTIONS statement
- Nonstandard INQUIRE specifiers: CARRIAGECONTROL, DEFAULTSFILE, KEYED, ORGANIZATION, and RECORDTYPE
- Nonstandard OPEN specifiers, such as ASSOCIATEVARIABLE, BLOCKSIZE, CARRIAGECCONTROL, EXTENDSIZE, NOSPANDBLOCKS, RECORDTYPE, and so on
- READs into Hollerith edit descriptors
- Variable expressions in FORMAT statements
- Structure field initialization
- %REF() function when used with CHARACTER argument
- %DESCR
- Non-CHARACTER dummy argument used as ENCODE/DECODE buffer
- Intrinsics that convert to a specific INTEGER size, such as IINT, JINT, IIDINT, JIDINT, and so on
- VMS system calls

The f77cvt Source-Code Converter

The migration of VMS FORTRAN code into Sun FORTRAN 1.0 format is accomplished through the f77cvt Source-Code Converter. This program accepts a file containing valid VMS FORTRAN source code, and produces a file containing FORTRAN source code that is acceptable to the f77 compiler. If the converter finds a construct that it cannot convert, it produces an appropriate message on standard error.

The resulting output file contains few, if any, nonstandard features or syntax of the `f77` compiler that are not also supported by the VMS compiler. One of the converter's options generates a list of warning messages for each nonstandard `f77` extension that is produces. A new compiler option identifies all non-ANSI extensions, including those that are not the result of conversion. Thus, in most cases the user should be able to compile the converted code on both the VMS FORTRAN and the `f77` compilers.

The source code converter accepts options that correspond to similar options on the VMS compiler. These options either generate the appropriate FORTRAN statements, or produce a diagnostic message indicating that the option is unrecognizable by the program. The converter has an option to write to standard output a comment on each conversion it has performed.

This conversion is performed as a separate step, and is a one-way translation. Therefore, the `f77cvt` Source-Code Converter cannot be used to transfer Sun FORTRAN 1.0 back to VMS FORTRAN; however, in most cases, the converted code will run under the VMS compiler.

There are some important considerations to keep in mind when using Sun FORTRAN and the converter program. These are discussed below.

Changes to Comment Lines and Indentation

Due to parser limitations, comment lines move and indentation is changed in the converted code when the compiler of the `f77cvt` converter program is parsing VMS FORTRAN code.

The before-and-after conversion examples below demonstrate how comment lines and indentation are affected in the source code file named `test.for`. Note the messages inserted by the converter program during the conversion process.

VMS FORTRAN source code comments before conversion

```
     i = i
c       all comments between the beginning of statement N and the
     1    +
c       beginning of statement N+1 are moved to precede statement N
     2    1
c       in f77cvt output.
      j = i
      end
```

VMS FORTRAN source code comments after conversion

```
c       all comments between the beginning of statement N and the
c       beginning of statement N+1 are moved to precede statement N
c       in f77cvt output.
# 1 "test.for"
      i = i + 1
# 7 "test.for"
      j = i
      end
```

VMS FORTRAN source code indents before conversion

```
c       original indentation is lost:
      dimension a(5, 10)
      do 20 i = 1, 5
          do 10 j = 1, 10
              a(i, j) = 0.0
10        continue
20    continue
      end
```

VMS FORTRAN source code indents after conversion

```
c       original indentation is lost:
      dimension a(5, 10)
# 3 "test.for"
      do 20 i = 1, 5
      do 10 j = 1, 10
      a(i,j) = 0.0
   10 continue
   20 continue
      end
```

INCLUDE Statements

The files referred to in the INCLUDE statements must be also be converted prior to use if they make use of VMS FORTRAN extensions.

The design of the converter program expects legal FORTRAN input; that is, complete program units. Therefore, files referred to within INCLUDE statements must be manually edited prior to conversion by appending an END statement to make the file appear as a program unit. After conversion, the END statement should be removed to return to the INCLUDE file back to its normal format.

The INCLUDE files must be separately converted prior to usage if they make use of VMS FORTRAN extensions.

DATA Statements

DATA statements are affected in the following ways:

- Initializations of local variables are moved to follow specification statements
- Initializations of variables in COMMON are moved into a generated BLOCK DATA subprogram, along with all necessary specification statements
- Implied-DO loops are unrolled

The following before-and-after conversion example demonstrates how DATA statements are affected in the source code file named test.for.

VMS FORTRAN source code DATA *statements before conversion*

```
      data (a(i), b(i), i = 1, 3) / 1, 2, 3, 4, 5, 6 /
      dimension a(3), b(3)
      common /c/ b
      print *, a, b
      end
```

VMS FORTRAN source code DATA *statements after conversion*

```
c       initializations of local variables are moved to follow
c       specification statements;  initializations of variables
c       in COMMON are moved into a generated BLOCK DATA subprogram,
c       along with all necessary specification statements.
c       implied-DO loops are unrolled.
      dimension a(3), b(3)
      common /c/ b
      data a(1) / 1 /
      data a(2) / 3 /
      data a(3) / 5 /
# 10 "test.for"
      write(unit=*, fmt=*) a, b
      end
      block data v00000000000000000000000000001
      common /c/ b
      real b(3)
      data b(1) / 2 /
      data b(2) / 4 /
      data b(3) / 6 /
      end
```

Additional Code Conversion Characteristics

The following before-and-after example demonstrates some additional characteristics of how the conversion program translates code from VMS FORTRAN into Sun FORTRAN 1.0.

VMS FORTRAN source code before conversion

```
    integer function binary_search(a, n, x)
    integer a(n)                    ! the array to be searched
    integer x                       ! the search object

    integer i                       ! lower end of search range
    integer j                       ! upper limit
    integer k                       ! midpoint

    i = 1
    j = n
    do while (i .le. j)
            k = (i+j) / 2
d           type *, k
            if (a(k) .eq. x) then
                    binary_search = k
                    return
            else if (a(k) .lt. x) then
                   i = k + 1
            else
                    j = k - 1
            endif
    end do
    binary_search = 0
    end
```

VMS FORTRAN source code after conversion

```
      integer function binary_search(a, n, x)
c the array to be searched
      integer a(n)
c the search object
      integer x
c lower end of search range
      integer i
c upper limit
      integer j
c midpoint
      integer k
# 9 "test.for"
      i = 1
      j = n
      do while (i .le. j)
c               type *, k
# 12 "test.for"
      k = (i + j) / 2
# 14 "test.for"
      if (a(k) .eq. x) then
      binary_search = k
      return
      else if (a(k) .lt. x) then
      i = k + 1
      else
      j = k - 1
      end if
      end do
      binary_search = 0
      end
```

Use of Double Quotes

In VMS FORTRAN and `f77cvt`, double quotes are used as an octal constant delimiter. For example, the statement `x = "77` is equivalent to `x = 63`. In Sun FORTRAN 1.0, double quotes are used to indicate a character string delimiter.

New and Changed FORMAT Statement Options

The following table lists new and changed FORMAT statement options and descriptors that are included with Sun FORTRAN 1.0.

Note that the field descriptors A, D, E, F, G, I, L, O, and Z can be written without the w, d, and e field indicators. If these are left unspecified, the appropriate defaults will be used, based on the data type of the I/O list element.

Option	Description
Q	Returns the length of an input record, or of the remaining portion that is unread
$	Suppresses the carriage return, and does not depend on the first character of the format.
O	Octal notation
Z	Hexadecimal notation

3

STB SHORT SUBJECTS

3

STB SHORT SUBJECTS

Bridge Box Processing Overloads

Bridge Box Processing Power

Some customers have observed problems between two machines connected to each other through a bridge box. With one machine mounted to the second machine's file system and all `fstab` entries correct, the error message shown below may result. This occurs with large files or during an attempt to run an executable. It does not occur when working with small files.

```
NFS Read Error for Server B
RPC timeout
```

The Workaround

Many Ethernet bridge boxes do not have the processing power necessary to keep up with a busy Ethernet, especially Sun workstations sending back-to-back packets on the net. It is possible to tune the NFS using parameters to the `mount` command to avoid placing too much processing demand on the bridge box.

Use the parameters shown below for use over Sun Internetwork Routers.

```
rsize=512, wsize=512, timeo=100
```

These should cause the amount of back-to-back traffic to remain within the power of the bridge box, at the price of reduced NFS throughput.

Refer to the article *Back-to-Back Ethernet Packets* on page 245 of the July STB for additional details on bridge box processing problems associated with sending large files across busy networks.

Using the id Utility

The id Utility

The id utility is new starting with SunOS release 3.2 and is loaded only if you select the System V optional software category. The id utility tells you who you are. This may be helpful in the case that you run into permission problems when you think you are logged in as root. You may be yourself instead.

An example of using the id utility is shown below.

```
plaid# id
uid=0(root) gid=1(daemon)
plaid# exit
plaid# plaid% id
uid=2295(chuq) gid=40(support)
plaid%
```

The id utility is also useful in determining group permissions and investigating other permissions problems since the command whoami may be used to determine user id (UID).

An example of this use is the situation when password files are not consistent between the NFS server and client, a user has one UID on the server and another UID on the client, and the user's login name is the same. In this case, everything *appears* to be correct, but the user is unable to access or modify files as expected.

Using id with usr/5bin/ls -n quickly identifies these discrepancies.

4

IN DEPTH

4

IN DEPTH

Network Information Transfers

Networking: Transfer of Information

This article contains an overview of several network-related topics. Most of the topics are software aspects of networking, with some hardware topics describing the physical network layout.

The major topics in this article are listed below.

- □ Ethernet theory of operation
- □ Network analysis and troubleshooting hints
- □ Network performance
- □ Subnet addressing
- □ Avoiding physical network problems
- □ Thin Ethernet (Cheapernet) specifications
- □ Level 1 and Level 2 equipment differences
- □ Frequently asked questions and answers

Ethernet Theory of Operation

Ethernet theory of operation includes a definition of CSMA/CD and a description of how the single network channel (Ether) is used.

CSMA/CD Definition

Ethernet activity is Carrier Sense Multiple Access with Collision Detection (CSMA/CD). CSMA/CD technology allows many devices access to the same network, in the absence of any central controller that manages channel access. Further, there are neither pre-allocated time slots as in token ring technology, nor fixed sharing of frequency bands.

Any device wanting to transmit onto the shared channel *contends* for channel (Ether) use until the channel is *acquired.* The device then transmits a packet onto the acquired channel. In this process, each device senses the channel carrier level and looks for collisions on the network.

Ether Use

Each device wanting to transmit a packet of information onto the Ether first senses whether or not a carrier signal level is present on the Ether. One of two voltages is sensed, depending on whether another device on the network is transmitting a message at that moment in time. This is Carrier Sense (CS) and your device considers the net busy if a carrier is sensed.

Your device *defers transmitting* your message until it senses that the Ether is quiet. This is when no carrier voltage level is sensed and the net is then considered *not busy.* Once a quiet net is detected, the deferring device immediately begins transmitting the packet containing your message. This is Multiple Access (MA) since each device connected to the Ether shares the channel in real time, in some ways similar to a 'party-line' telephone service.

Your device listens for a collision during the time it takes to transmit the packet containing your message. This is the Collision Detection (CD) that lets each device know that its messages have been transmitted without any other device trying to 'break into' your 'conversation'. A collision is defined as any two devices transmitting to the same network at the same time. Unlike the old-fashioned, 'party line' telephone line, all messages are *retransmitted* if a collision is detected.

All devices look for collisions when transmitting since there is a short time interval just after a transmission begins when another device on the Ether might also begin to transmit its message. This occurs since the second device may not have sensed the carrier voltage that the first device places on the channel to transmit the packet containing the message.

This time interval is called the *collision window* or *collision interval,* after which all devices on the network will detect your carrier and defer their transmissions until yours is completed. This interval is a result of the end-to-end propagation delay on the net.

Your device continues to transmit the packet if no collision is detected during the collision window. However, each device continues to look for any collisions during the entire transmission process, in the case that a malfunctioning device is connected to the network. For example, a machine can lose the ability to detect a carrier on the network and therefore cannot sense collisions. This is known as *babble mode.*

If your station detects a collision, the current transmission is *aborted.* The network is briefly jammed by any connected station what detects a collision by invoking a collision consensus enforcement procedure. Basically, a given packet is fully transmitted only when no device on the net senses a collision, either during or after the collision window.

Each machine involved in the collisions then schedules its packet for *retransmission* at a later time. The two stations sharing the collision use a random delay period so that they each retransmit at a different time. This delay reduces the possibility that each station will retransmit at the same time and cause a second collision.

Network Analysis and Troubleshooting Hints

This discussion of network analysis and troubleshooting includes running the proper SunOS release level with a proper netmask and diagnosing network routing problems.

SunOS Releases 3.3 and 3.4; and Subnetting

Most customer calls on networking result from trying to use subnets on SunOS release that do not support subnetting. You need to run SunOS release 3.3 or subsequent releases for subnetting. Further, you need to use the proper netmask. Subnet addressing and netmasks are discussed later in this article.

Routing Problem Troubleshooting

Several commands may be issued from your command line to diagnose network routing problems. Commands discussed in this topic are shown below. Please note that the following examples are provided for illustrative purposes only and do not represent actual networks.

- □ Using `netstat -r`
- □ Using `ifconfig`
- □ Using `/etc/networks`

Using `netstat -r`

When `ping` or `rlogin` report *Network Unreachable*, you will need to look at the network routing tables. This allows you to see what gateway the machines think they should use to access other networks. An example of output from the `network` command using the `-r` option is shown below.

```
machine# netstat -r

Routing tables
Destination      Gateway          Flags     Refcnt Use       Interface
sunpie-ptp       sunsnow          UGH       0      0         le0
dish-ptp         sunsnow          UGH       0      0         le0
backbone         sunsnow          UG        1      249       le0
gsd-localnet     sunsnow          UG        0      0         le0
supnet           machine          U         9      16631     le0
loopback         localhost        U         2      1363      lo0

machine#
```

Your device and your internal, loopback network are designated as being UP in the 'Flags' column by `U`. `G` denotes a machine that is a gateway to the network named in the 'Destination' column. H signifies the network host, the gateway to be used to gain access to the named network.

Using the -r option gives similar information with networks identified by the Internet subnet addresses. An example is shown below.

```
machine# netstat -n -r

Routing tables
Destination      Gateway          Flags    Refcnt Use        Interface
192.9.22.1       192.8.5.44       UGH      0      0          le0
192.9.21.1       192.8.5.44       UGH      0      0          le0
192.9.3          192.8.5.44       UG       1      249        le0
192.9.133        192.8.5.44       UG       0      0          le0
192.9.9          192.9.2.14       U        8      16949      le0
127.0.1          127.0.0.1        U        3      1396       lo0

machine#
```

Using ifconfig

When ping or rlogin report *Connection Timed Out*, you will need to determine whether the Internet address has changed, or whether the gateway between the machines is down. The problem can also be lack of physical connection between machines.

Use the ifconfig command to determine the status of a network interface, its hosthame, and network address.

```
machine# ifconfig le0

le0: 192.9.1.14 flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>

machine# ifconfig le0 netmask 0xffffff00
machine# ifconfig le0

le0: 192.9.1.14 netmask 255.255.255.0 flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
```

Note that ifconfig reports the netmask only if it has been set. If it has not been set, the appropriate default netmask shown below is in use.

Network Address Class	Default Netmask (Hexadecimal)	Default Netmask (Decimal)
Class A	0xff 00 00 00	255 00 00 00
Class B	0xff ff 00 00	255 255 00 00
Class C	0xff ff ff 00	255 255 255 00

Using /etc/networks

The networks file is located in either the /etc or yp domain_name directory. This file is needed for correct propagation of routing tables. This file allows you to more conveniently refer to networks by name rather than by Internet number. A sample networks file is shown below.

```
machine# more /etc/networks

#
# Sun customer networks
#
loopback     127
sun-ether    192.8.193    sunether ethernet localnet
#
# Internet networks
#
arpanet      10       arpa
#
# Local networks
#
ptp-net      192.2.300    my-net her-net
gymnet       128.8.3      # Home Ec subnet
solnet       192.8.21     # Solidarity Lab Network
nflsubnet    192.8.14.145    # NFL Subnet
lasernet     192.8.54     # Laser lab net
#
# Internetwork Routers
#
spdinr7      192.8.13     # Rond Road
spdinr8      192.8.14     # Rond Road
spdinr9      192.8.15     # Calistoga
```

Network Performance

Use the netstat and nfsstat commands to determine whether your network error, collisions, and nfs and nd statistics are within acceptable limits.

Using netstat

Calculate your collision rate by first dividing the amount shown under 'Collis' by the amount shown under 'Opkts' and then multiplying the quotient by 100; (Collis/Opkts)*100=collision rate. Collision rate acceptability is great for a 1-2% range, fair for 3-5%, and poor for over 5%. Note that errors are never acceptable.

An example of netstat output using the -i option is shown below. This example shows a bad collision rate of 9.3%. This was calculated as 16477 / 177024 x 100 = 9.3%, a very bad collision rate.

```
machine# netstat -i

Name  Mtu   Net/Dest     Address      Ipkts   Ierrs Opkts   Oerrs Collis Queue
le0   1500  supnet       snowflake    832513  0     177024  1     16477  0
lo0   1536  loopback     localhost    1961    0     1961    0     0      0
```

A second use of netstat using the -i and 5 options is shown below. This shows the packets and errors for the input, output, and total input and output, for the time since the machine was booted, and every five seconds.

```
machine# netstat -i 5

    input   (le0)     output            input   (Total)    output
packets errs  packets errs  colls packets errs  packets errs  colls
832857  0     177306  1     16486 834838  0     179287  1     16486
16      0     65      0     0     16      0     65      0     0
13      0     0       0     0     13      0     0       0     0
7       0     0       0     0     7       0     0       0     0
```

The final example shows netstat using the -m option. This option reports the buffers allocated to network processes.

```
machine# netstat -m

302/448 mbufs in use:
        65 mbufs allocated to socket structures
        83 mbufs allocated to protocol control blocks
       154 mbufs allocated to routing table entries
0/16 mapped pages in use
184 Kbytes allocated to network (20% in use)
0 requests for memory denied
```

Using nfsstat

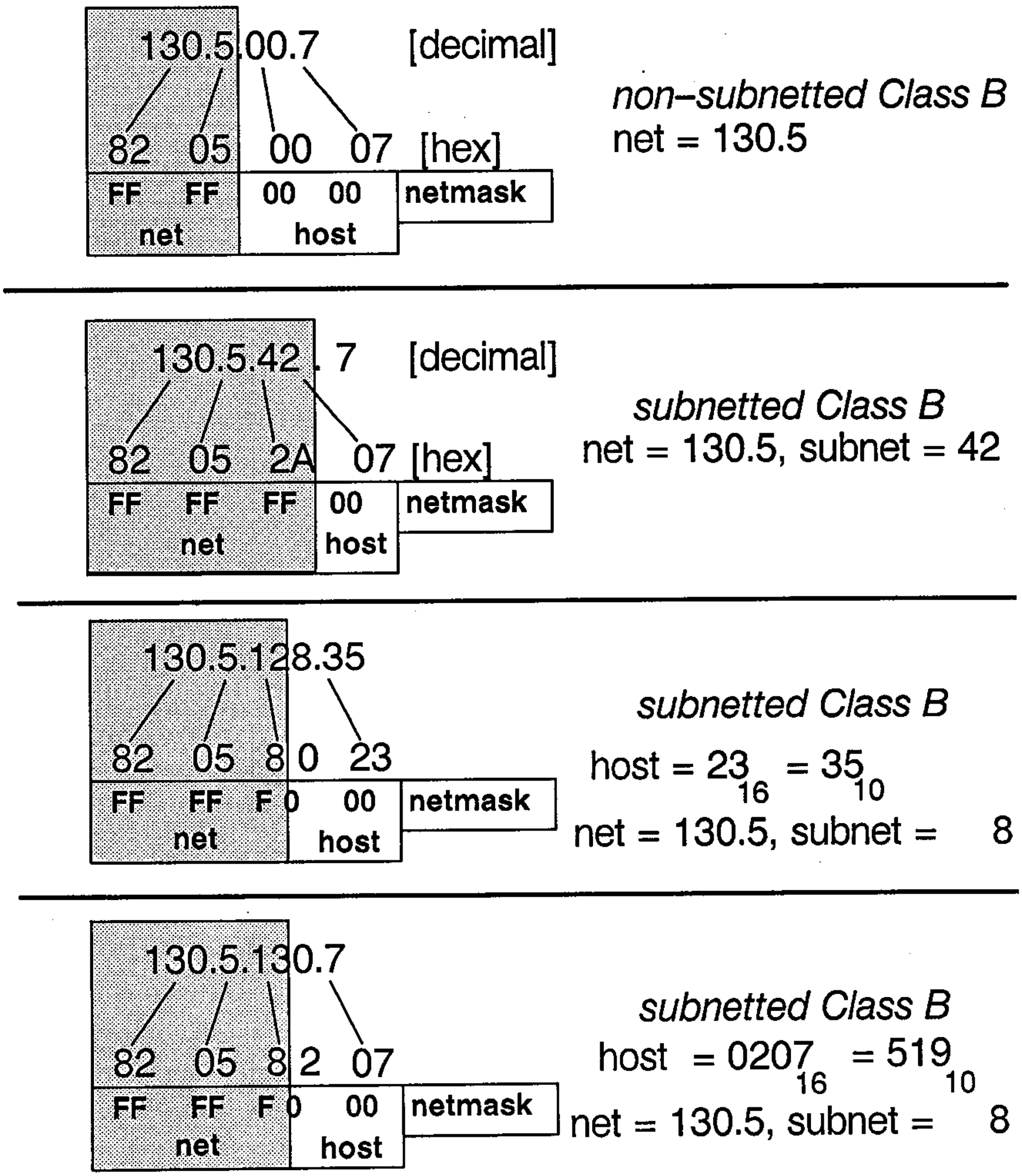
Use the nfsstat command for checking nfs and nd statistics, particularly retransmission figures. These figures give you an idea of how good the network connection is. Look for timeouts and any entry labeled *bad....* A sample usage of nfsstat is shown below.

```
machine# /usr/etc/nfsstat

Network Disk:
rcv 188971  snd 161467  retrans 66  (0.04%)
notuser 0  noumatch 107  nobuf 0  lbusy 0  operrs 0
rseq 41  wseq 0  badreq 0  stimo 0  utimo 0  iseq 0

Server rpc:
calls      badcalls   nullrecv   badlen     xdrcall
0          0          0          0          0

Server nfs:
calls      badcalls
0          0
null       getattr    setattr    root       lookup     readlink   read
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
wrcache    write      create     remove     rename     link       symlink
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
mkdir      rmdir      readdir    fsstat
0 0%       0 0%       0 0%       0 0%

Client rpc:
calls      badcalls   retrans    badxid     timeout    wait       newcred
13048      0          3          1          3          0          0

Client nfs:
calls      badcalls   nclget     nclsleep
13048      0          13048      0
null       getattr    setattr    root       lookup     readlink   read
0   0%     4346 33%   10   0%    0   0%     2021 15%   4656 35%   1325 10%
wrcache    write      create     remove     rename     link       symlink
0   0%     93  0%     52  0%     1   0%     6   0%     0   0%     0   0%
mkdir      rmdir      readdir    fsstat
0   0%     0   0%     532  4%    6   0%
```

Subnet Addressing

You cannot determine the meaning of a class B subnet address by inspection. The host address and net address share the address digits in one of several ways. See the following figure for one non-subnetted class B example, and three subnetted class B address examples.

Subnet Addressing



Avoiding Physical Network Problems

You can avoid problems with the network hardware by observing the following points.

- Avoid flourescent lights and power conduits when running network coaxial cables through the ceiling.
- Ground an Ethernet segment in a *single* place only. Multiple grounding points cause ground loops which may be misinterpreted by machines connected to the network as transmission carriers.
- Have 100 or fewer nodes on the net, not up to 1024 as indicated in the network specification.
- Place nodes only every 2.5m on the thick Ethernet coax. Any other placement causes standing waves on the cable which may be misinterpreted by machines. High error and collision rates may result.
- Place only one or two Ethernet repeaters between *any* two nodes.
- Place only one or two thick-to-thin Ethernet converters on a network.
- Have only a single network number on any one physical network. Multicasting is not supported in SunOS releases 3.x.
- Ensure that the bore is cleaned out using a swab when installing a vampire-type transceiver.
- Define only a single Ethernet address for a gateway machine. A gateway has only one Ethernet address, but two Internet addresses, and two hostnames. The Ethernet address originates from the CPU HostID chip. Note that an Ethernet address needs to be unique for any one network only. The same Ethernet address may be used on *different* networks.

Thin Ethernet (Cheapernet) Specifications

The trunk cable (thin enet cable) is of constant impedance, coaxial construction. It is terminated at each of the two ends by a terminator of 50 ohms, plus or minus 1%, measured from 0 - 20 MHz. The terminator power rating shall be 0.5 watts or greater.[6]

The cable parameters are normally met by cable types RG58 A/U or RG 58 C/U. The center conductor shall be stranded, tinned copper with an overall diameter of 0.89mm, plus or minus 0.05mm. Devices attached to the trunk require a BNC 'T' connector of 50-ohm impedance. Note that this connector must be connected *directly* to the device, *without* any additional connecting cable.

[6] The information contained in this discussion is taken from *IEEE Draft Standard 802.3*, Section 10, 'Medium Attachment Unit and Baseband Medium Specifications', Type 10BASE2, dated March 1985.

The attenuation of a cable of the maximum 600 foot (185m) length shall not exceed 8.5dB measured at 10MHz, or 6.0dB measured at 5MHz. Also, the maximum length of a segment is 600 feet, with a maximum of 30 devices or Medium Attachment Units (MAUs) connected to the segment.

The maximum end-to-end propagation delay for a coaxial segment is 950ns.

The maximum transmission path permitted between any two MAUs is limited by the maximum of four repeater sets that can be connected in series. This shall consist of no more than three tapped coaxial segments; the remainder shall be link segments.

The minimum distance between any two nodes is 1.6 feet (0.5m). No more than 30 nodes shall be on a single network segment. *No cables are allowed between the BNC T-connector and the device!* Finally, the network may neither loop nor branch.

Level 1 and Level 2 Equipment Differences

Ethernet hardware equipment is available in two versions or levels. These two versions are also known as DIX 1 and DIX 2 (DEC, Intel, Xerox). The two levels are compatible on the coaxial cable. However, a level 1 Ethernet board should be connected to a level 1 transceiver, and similarly for level 2 boards and transceivers. There are usually no problems communicating between level 1 and 2 systems across the coaxial cable, unless the network has grown extremely large.

Two features distinguish level 2 transceivers from level 1 equipment. The first feature is *jabber* which allows the transceiver to watch the packet data stream during transmission. If the data stream is longer than the maximum legal packet size, the level 2 transceiver shuts down. Note that this feature is not universally available on all level 2 transceivers. The jabber feature prevents a malfunctioning CPU from rendering the network unusable.

The second unique, level 2 feature is *heartbeat* which is a signal that occurs each time the transceiver receives a transmission from the attached device. The heartbeat signal is transmitted back to the device using a specific line in the transceiver cable.

The IEEE 802.3 standard is an Ethernet-like protocol, and is hardware-compatible with level 2 Ethernet equipment. However, the packet structure differs such that the two protocols do not work together on the software level. From the hardware perspective, IEEE 802.2 packets, IEEE 802.3 packets, and Ethernet packets *can* coexist on the same physical network. The Ethernet controllers read all three types of packets.

The driver software, however, must treat some of the packet fields differently. Therefore, IEEE 802.3 and Ethernet machines can talk to like-machines only on the same physical network, with unlike-machines being transparent.

Ethernet level 1 and level 2 differences are found in Ethernet controller and transceiver interaction only. The differences effect neither the electrical interface nor packets sent between transceivers on the coaxial cable. Note that the Sun Ethernet board can be configured as either level 1 or level 2.

Frequently Asked Questions and Answers

1. *What is the longest length of Ethernet cable in a network?*

   The longest length possible is 1640.5 feet (500m). This length can be composed of one full, continuous piece of cable. It can also be composed of segments of 23.4m, 70.2m, or 117m lengths.

   If odd-length cable sections must be used, choose the length so that the resulting reflected signals on the cable are not out of phase with the actual signal. This is usually accomplished by using lengths which are odd multiples of the half-wave length at 5Mhz. This length corresponds to the recommended segment lengths of 23.4m 70.2m, and 117m.

   Transceiver drop cables have a maximum length of 164 feet (50m).

2. *What is the maximum time between recognition of the collision and repeating of the collision?*

   The maximum time interval between collision recognition and a repeat collision excluding the carrier sense random retiming delays is 200ns.

3. *Should I ground the device or system?*

   Yes! The sheath conductor of the transceiver shall be connected to an earth ground or chassis, but at *one and only one* point per segment. Any second or additional ground points will only introduce degenerative ground loops, which are seen as a higher voltage level on the coaxial cable. This can be misinterpreted and sensed as a transmission carrier signal voltage.

4. *What are the proper lengths for adding transceivers to the network cable?*

   Transceivers may be added at a minimum of 2.5m from each other, or at multiples of 2.5m. Proper intervals are printed on the outside plastic sheath of the Ethernet coaxial cable.

5. *How many transceivers may be placed on the network?*

   A maximum of 100 nodes may be placed on the network.

6. *What are the timing constraints between a detected collision and carrier sense?*

   The channel logic must assert the collision-detect signal within 200ns following the collision. The channel must then deassert the collision-detect signal within 160ns after the loss of the collision-occurring signal. The carrier-sense is asserted when two stations are transmitting at the same time must be asserted within 200ns. The channel then as 160ns to deassert carrier-sense after a carrier is no longer present on the network.

7. *What do I need to know about repeaters?*

A repeater is a device used to extend the cable length beyond the single coaxial segment length of 500m. Repeaters require a transceiver at each of the segments for which it is repeating signals. A maximum of two repeaters may be in the signal path between any two transceivers.

The repeater implements the carrier-sense and the collision-detect/repeat function for each cable segment it connects. All signals are retimed and amplified to allow for propagation to the other connected cable segment. With carrier-detect, the propagation delay through the repeater cannot exceed 800ns. With collision-detect and retransmission, the delay cannot exceed 200ns.

The repeater ensures that the signal retransmitted from one cable segment to the other has the same amplification as it would when leaving a station's transceiver and entering the Ethernet coaxial cable.

8. *How do I know that a packet preamble is correct?*

The preamble should appear as shown below. This 64-bit preamble may be viewed with proper software or a waveform analyzer. The preamble will appear on an oscilloscope as a periodic waveform of 5Mhz frequency.

```
10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011
```

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

Q&A, and Tip of the Month

Hints & Tips #7

This is the seventh in a continuing series of this column which I have created for two purposes.[7] First, some questions are asked regularly on the AnswerLine. I feel everyone can benefit from distributing discussions of these problems as widely as possible. Second, a large and constantly growing body of information, hints, and tips are not documented anywhere.

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

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

The SunOS and Network Services

Many parts of the Sun operating system (SunOS) use network services -- programs and features such as NeWS and perfmeters (performance meters) that are based on the code that interacts with the ethernet and other machines.

Some machines, however, are not attached to networks. Without some changes to a couple of files, you will not be able to use most network services. These changes are not clearly documented and are not done automatically by `setup` during SunOS installation.

[7] This continuing column is submitted by Chuq Von Rospach, Customer Software Services.

The way to get the network services to work properly when no network exists is to change your system to use the internal 'loopback' network. To do this, you have to change two files.

SunOS File Changes

First, edit `/etc/hosts`. On a non-networked machine, there should be two entries in the file as shown below.

```
192.9.200.1 hostname        # the number may vary
127.0.0.1   localhost
```

Change the file so that both names are defined to be host number 127.0.0.1. This is done by deleting the *hostname line*, and by adding *hostname* to the original localhost line. See the example below, and note that *hostname* must appear first. Also ensure that you do not define your hostname twice by forgetting to delete the original hostname line.

```
127.0.0.1   hostname localhost  # order is important!
```

Second, edit `/etc/rc.boot`. Three lines configure the network interfaces and initialize them using the `ifconfig` command. Comment-out the three lines as shown in the following example.

```
#/etc/ifconfig ec0 $hostname -trailers up
#/etc/ifconfig ie0 $hostname -trailers up
#/etc/ifconfig le0 $hostname -trailers up
```

After this, you will need to reboot as shown below.

```
machine% su
machine# /etc/halt
   ...
Syncing disks...  done
Unix Halted

>b
   ...
machine login:
```

Your system is now configured to use the loopback device as the network, and all network programs should function normally. If you later decide to install the machine on a real network, it is simple to remove the changes to these two files and return your system to the normal networking functions.

Tip of the Month (TOM)

This month's tip discusses a way of letting the `Mail` program send messages to the printer. `Mail` does not have the capability of spooling a copy of a mail message to the printer. You must first save the message in a file, then print the file, and finally delete it. A better solution is to allow the message to be printed without first saving it in a file. The way to do this is to set the `EDITOR` variable in `/usr/lib/Mail.rc`, the system-wide parameter list, to `/usr/ucb/lpr`, as shown in the following example.

```
set EDITOR=/usr/ucb/lpr
```

If you do not want to change the `EDITOR` variable for the entire machine, a user can put the command shown above in his or her personal `.mailrc` file. This sets `EDITOR` for that individual only. The user then can enter `e <message-number>`, and the message is printed.

Users who are accustomed to using the `ex` editor by entering `e <message-number>` can set the `VISUAL` variable to `/usr/ucb/ex` and edit by entering `v <message-number>`. You could otherwise set the `EDITOR` variable back to `/usr/ucb/ex` and set the `VISUAL` variable to `/usr/ucb/lpr`. You would then use `v <message-number>` for the printing function. Again, these changes would be made in the *user's* personal `.mailrc` file.

6

THE HACKERS' CORNER

6

THE HACKERS' CORNER

Logging Console Messages

A Console Messages Log Program

The program shown below was developed by Sun in the United Kingdom. It logs console messages into a file or onto a printer. The program uses the same `ioctl` as `cmdtool`. Before using this program, you may need to search all `.suntools` files as well as the system-wide `.suntools` file to remove the `-C` option for the console `cmdtool`.

Note that a better method than that shown above is to remove the `-C` option from the `cmdtool` program itself.

Program Use

Comments on console messages log program features, improvements, or the user interface are welcomed. Please send them to *sun!stb-editor*. I will then forward them to Sun Europe.

Please consult your local shell script or programming expert regarding any script or code problems. The console messages log program is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves. Note that the program may not work in all cases (e.g. guaranteed append mode is not supported over NFS), and may not be compatible with future SunOS releases.

The Program

The program code appears on the following pages.

```
/* Program to log console messages in a file. */
/* Command syntax: conslog [file]. */
/* If no file is given, logging is done on `stdout'. */
/* `cmdtool' also takes control of console logging (-C option). */
/* Make sure all `cmdtool -C' options are removed from all `.suntools' */
/* and `/usr/lib/suntools' files,
/* or change the `toolmerge' or the `cmdtool' itself. */
/* The program uses 1 `pty, to connect to the console. */

#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/file.h>
#include <stdio.h>

int width;
int cons,mstr,ctty;

char pty[]={
    "/dev/ptyp0"};
char tty[]={
    "/dev/ttyp0"};

struct timeval timeout;

char buffer[512];

#define MAX1 '9'+1
#define MAX2 'f'+1

char *timestr()
{
    int t;
    char *ctime(),*s;

    t=time(0);
    s=ctime(&t);
    return(s);
}

main(argc,argv)
int argc;
char **argv;
{
    int r,w,s,log;
    register int rmask,i;

    if (argc > 1)
    {
        if((log=open(argv[1],O_RDWR | O_APPEND)) < 0)
            if((log=creat(argv[1],O_RDWR | 0666)) < 0)
            {
                perror("conslog: cannot create ",argv[1]);
                exit(2);
            }
    }
    else
        log=1;

    mstr=getpty();
    ctty=open(tty,O_RDWR);

    if(ioctl(ctty,TIOCCONS) == -1)
    {
        perror("conslog: Cannot attach console to pty.\n");
        exit(1);
    }

    width=32;
    rmask=(1<<mstr);

    s=0;
    w=0;
    do {
        r=rmask;
        select(width,&r,&w,&s,0);
        if(r)
        {
            i=read(mstr,buffer,512);

            write(log,timestr(),25);
            write(log,buffer,i);
        }
    } while (i >= 0);
    close(mstr);
    perror("conslog:");
    exit(3);
}

getpty()
{
    int i=0,f= -1;
    while(i <  48)
    {
        if((f=open(pty,O_RDWR)) > 0)
            return(f);
        i++;

        ++pty[9];
        ++tty[9];
        switch(pty[9])
        {
        case 'MAX1':
            pty[9]='a';
            tty[9]='a';
```

```
                break;
            case 'MAX2':
                pty[9]='0';
                pty[8]++;
                tty[9]='0';
                tty[8]++;
                break;
            }
        }
        perror("conslog: Cannot open a pty");
        exit(4);
}
```

7

CUMULATIVE INDEX: 1987

Index

Revision History

Revision	*Date*	*Comments*
FINAL	October 1987	Ninth 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