

# *Sun Systems Fault Analysis Workshop*

*ST-350*

## Student Guide



Sun Microsystems, Inc.  
MS BRM01-209  
500 Eldorado Blvd  
Broomfield, Colorado 80021  
U.S.A.

Revision D.1, July 1999

Copyright © 1999 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, OpenBoot, Ultra, SunSolve, SunVTS, AnswerBook, NFS, SyMON, UltraSPARC, Ultra Enterprise, SunDiag, and SPARCstorage are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government approval required when exporting the product.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Govt is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



Please  
Recycle



Adobe PostScript

# Contents

---

<b>About This Course</b> .....	xi
Course Map .....	xii
Module-by-Module Overview .....	xiii
Course Objectives.....	xvi
Skills Gained by Module.....	xvii
Guidelines for Module Pacing .....	xviii
Topics Not Covered.....	xix
How Prepared Are You?.....	xxi
Introductions .....	xxii
How to Use Course Materials .....	xxiii
Course Icons.....	xxv
Typographical Conventions .....	xxvi
Attention VARs .....	xxvii
<b>Fault Analysis and Diagnosis</b> .....	1-1
The Fault Analysis Method .....	1-3
Eight Steps of Fault Analysis and Diagnosis .....	1-4
Fault Analysis .....	1-4
Diagnosis .....	1-5
Stating the Problem.....	1-6
Writing a Problem Statement .....	1-6
Guidelines for a Problem Statement .....	1-7
Describing the Problem.....	1-8
Listing All Observed Facts.....	1-9
Establishing Comparative Facts.....	1-12
Identifying Differences.....	1-13
Listing Relevant Changes .....	1-15
Generating Likely Causes .....	1-16
Stating Your Hypothesis .....	1-16
Testing Likely Causes.....	1-18
Verification.....	1-19
Verifying the Most Likely Cause .....	1-21
Taking Corrective Action.....	1-22
Fault Analysis Worksheet Example .....	1-23

Exercise: Conducting Fault Analysis and Diagnosis .....	1-26
Preparation.....	1-26
Tasks .....	1-26
Fault Analysis Worksheet Template .....	1-27
Exercise Summary.....	1-29
Check Your Progress .....	1-30
Think Beyond .....	1-31
<b>Diagnostic Tools.....</b>	<b>2-1</b>
Introduction to Hardware and Software Errors.....	2-3
Error Categories .....	2-4
Error Reporting Mechanisms .....	2-4
Type of Errors.....	2-5
Software Errors.....	2-5
Hardware-Corrected Errors .....	2-5
Recoverable Errors.....	2-6
Critical Errors .....	2-6
Fatal Errors.....	2-6
Error Reporting Mechanisms .....	2-7
Bus Errors.....	2-7
Interrupts.....	2-8
Watchdog Resets.....	2-9
CPU Watchdog Reset .....	2-9
System Watchdog Reset.....	2-9
Differentiating Hardware and Software Faults.....	2-10
Diagnostic Commands and Tools.....	2-12
Diagnostic Files .....	2-17
Applying the Tools .....	2-20
Device Management Utilities .....	2-21
The <code>drvconfig</code> Utility .....	2-21
The <code>devlinks</code> Utility.....	2-22
The <code>tapes</code> Utility .....	2-22
The <code>disks</code> Utility .....	2-22
The <code>ports</code> Utility .....	2-23
The <code>modinfo</code> Utility.....	2-24
The <code>modload</code> Utility.....	2-25
The <code>modunload</code> Utility.....	2-27
Networking Utilities.....	2-28
The <code>ifconfig</code> Utility.....	2-28
The <code>netstat</code> Command .....	2-31
Debugging Utilities.....	2-33
The <code>truss</code> Utility .....	2-33
Error-Free <code>truss</code> Command Output .....	2-34
Interpreting <code>truss</code> Command Output .....	2-37
General Guidelines .....	2-37
The <code>truss ls</code> Example.....	2-38

---

Example of truss Command Output With Errors.....	2-39
Interpreting Errors in truss Command Output.....	2-41
General Guidelines .....	2-41
The truss ls Example.....	2-42
The errno.h Header File.....	2-43
Exercise: Using Solaris Troubleshooting Tools.....	2-44
Preparation.....	2-44
Tasks .....	2-44
Exercise Summary.....	2-48
Check Your Progress .....	2-49
Think Beyond .....	2-50
<b>POST Diagnostics .....</b>	<b>3-1</b>
Diagnostics Overview .....	3-3
Boot PROM POST .....	3-4
POSTs.....	3-4
SunVTS Diagnostics.....	3-5
POST Viewing Methods.....	3-6
Viewing POST Using the tip hardware Command .....	3-7
Primary Buses.....	3-8
Sun 4m Architecture.....	3-9
SPARCstation 5 POST Example With tip .....	3-10
Ultra 5 and Ultra 10 Architecture .....	3-13
UltraSPARC pci-based Device Tree Hierarchy .....	3-14
UltraSPARC pci-based Example With tip .....	3-15
UltraSPARC Sbus-based Architecture.....	3-20
CPU/Memory Board Layout .....	3-20
I/O SBus Board Component Locations .....	3-21
Centerplane Numbering Scheme.....	3-22
Reading Device Path.....	3-23
POST Sample Output .....	3-25
POST Diagnostic Workshop Using tip.....	3-34
Using Terminal Interface Protocol (TIP) for Remote Diagnostics.....	3-34
POST tip Commands .....	3-35
Exercise: Using tip to Observe POST Diagnostics.....	3-37
Preparation.....	3-37
Tasks .....	3-38
Exercise Summary.....	3-41
Check Your Progress .....	3-42
Think Beyond .....	3-43
<b>OBP Diagnostics and Commands.....</b>	<b>4-1</b>
Functions and Capabilities of Open Boot PROM.....	4-3
OBP Features .....	4-4
Open Boot PROM.....	4-6
NVRAM Contents – Sun4m Architecture .....	4-7

---

NVRAM Contents – Ultra Workstations .....	4-9
The eeprom Command.....	4-11
Changing PROM Variables With eeprom .....	4-12
Events During Power On.....	4-13
Default Boot Sequence.....	4-14
The rc Files and Directories .....	4-18
OBP Device Tree Navigation – UltraSPARC Workstation .....	4-20
OBP Device Tree Navigation – SPARCstation 1000 System.....	4-23
OBP Features – Ultra Enterprise Servers.....	4-24
NVRAM Parameters – Ultra Enterprise Servers .....	4-25
Running OBP Commands With Key Sequences .....	4-26
Flash PROM Update for Sun4u Architectures.....	4-27
Systems and Firmware .....	4-27
Summary of Procedure .....	4-28
Ultra Workstation OBP Command Examples .....	4-29
SPARCstation 20 OBP Command Examples .....	4-31
Exercise: Using Ultra Workstation OBP Utilities .....	4-33
Preparation.....	4-33
Tasks .....	4-34
Exercise: Working With SPARC 5 Workstation OBP Utilities .	4-41
Preparation.....	4-41
Tasks .....	4-42
Exercise Summary.....	4-51
Check Your Progress .....	4-52
Think Beyond .....	4-53
<b>SunSolve Database Information .....</b>	<b>5-1</b>
Distribution.....	5-3
SunSolve Online Account .....	5-4
Installing SunSolve Software.....	5-5
Configuring a SunSolve Server .....	5-8
Running SunSolve Software.....	5-10
The SunSolve Home Page Selections .....	5-11
Power Search Tool Overview .....	5-14
Power Search Collections.....	5-15
Defining Search Criteria.....	5-16
Power Search Results.....	5-17
Boolean Search Syntax.....	5-18
The Current Patch Report .....	5-19
The Solaris 2.6 Patch Report .....	5-20
Patch Diag Tool .....	5-21
Patch Diag Report .....	5-22
Installing Patches .....	5-24
Removing a Patch .....	5-26
Useful SunSolve Documents .....	5-27
Exercise: Using SunSolve Fault Analysis .....	5-29

Preparation.....	5-29
Tasks .....	5-30
Exercise Summary.....	5-32
Check Your Progress .....	5-33
Think Beyond .....	5-34
<b>SunVTS System Diagnostics .....</b>	<b>6-1</b>
Introduction .....	6-4
SunVTS Software Overview .....	6-4
Hardware and Software Requirements .....	6-5
The SunVTS Architecture .....	6-6
Interfaces .....	6-7
User .....	6-7
Kernel.....	6-7
Hardware Tests .....	6-8
Installing SunVTS Software.....	6-9
Starting the SunVTS Software.....	6-11
The SunVTS Graphical Interface.....	6-12
The SunVTS Window Panels.....	6-13
The SunVTS Window Icons.....	6-15
The SunVTS Menu Selections.....	6-16
The Schedule Options Menu .....	6-18
The Test Execution Menu .....	6-19
The Advance Options Menu .....	6-22
Intervention Mode .....	6-23
Performance Monitor Panel.....	6-24
Using SunVTS in TTY Mode .....	6-26
Negotiating the SunVTS TTY Interface .....	6-27
Running SunVTS Remotely.....	6-28
Kernel Interface .....	6-28
User Interface.....	6-28
Graphical User Interface .....	6-29
The User Interface.....	6-30
Connecting Directly to the Remote Computer .....	6-30
TTY interface.....	6-30
Exercise: Diagnosing Problems With SunVTS.....	6-31
Preparation.....	6-31
Tasks .....	6-32
Exercise Summary.....	6-34
Check Your Progress .....	6-35
Think Beyond .....	6-36
<b>Kernel Core Dump Analysis.....</b>	<b>7-1</b>
Introduction .....	7-3
System Panics .....	7-3
System Hangs .....	7-4
Configuring a System to Collect Crash Dumps.....	7-5

The dumpadm Utility .....	7-7
The savecore Utility .....	7-9
Debuggers .....	7-11
adb .....	7-11
crash .....	7-11
kadb .....	7-12
Invoking the Debuggers .....	7-13
adb .....	7-13
crash .....	7-14
General Purpose adb Commands .....	7-15
Format Specifications With adb .....	7-17
adb Commands .....	7-18
Syntax Example .....	7-19
Register References in adb .....	7-20
Using Macros in adb .....	7-22
adb Macros .....	7-23
Header Files and adb Macros .....	7-24
Frequently Used Header Files and adb Macros .....	7-26
Core Dump Analysis .....	7-27
Sample Problem .....	7-27
Additional Information .....	7-28
Core Dump Analysis Using adb .....	7-29
Information Gathering Using strings .....	7-35
Commonly Used crash Commands .....	7-38
Core Dump Analysis – Summary for Using crash .....	7-40
Core Dump Analysis Using crash .....	7-41
The ISCDA Document .....	7-43
Initial System Crash Dump Analysis .....	7-43
Exercise: Performing Kernel Core Dump Analysis .....	7-45
Preparation .....	7-45
Tasks .....	7-46
Analyzing a Hung System .....	7-51
Modifying Kernel Parameters .....	7-52
(Optional) Core Dump Analysis With kadb .....	7-53
Exercise Summary .....	7-54
Check Your Progress .....	7-55
Think Beyond .....	7-56
<b>Fault Tracker Progress Chart .....</b>	<b>A-1</b>
Fault Tracker Progress Chart .....	A-2
Fault Analysis Worksheet Template .....	A-4
<b>Watchdog Resets .....</b>	<b>B-1</b>
Troubleshooting Watchdog Resets .....	B-2
OBP Environment .....	B-2
OBP Register Commands .....	B-3
OBP Register Commands – sun4u .....	B-3



OBP Register Commands – sun4m.....	B-3
Procedure for Troubleshooting Watchdog Resets .....	B-4
Virtual Memory Overview .....	B-5
Page Faults .....	B-5
The Memory Management Unit .....	B-6
Exercise: Analyzing Watchdog Resets and Virtual Memory ....	B-7
Preparation.....	B-7
Tasks .....	B-8
Locating Virtual Memory Information .....	B-14
Exercise Summary.....	B-15
<b>kadb and nm Commands .....</b>	<b>C-1</b>
Invoking the kadb Debugger.....	C-2
Introduction to Kernel Structures.....	C-3
Using kadb .....	C-3
Exercise: Conducting Kernel Core Dump Analysis With kadb	C-5
Tasks .....	C-5
Debugging Utilities.....	C-9
The nm Utility .....	C-9
The nm Command Output Display .....	C-10
The nm Column Headers .....	C-11
<b>Fault Worksheets.....</b>	<b>D-1</b>
Preliminary Task – Create a Student Account.....	D-1
Fault Worksheets – Student Guide.....	D-2
Fault Worksheet #1 – Blank Monitor.....	D-5
Fault Worksheet #2 – Device Error During Boot.....	D-7
Fault Worksheet #3 – File System Errors During Boot .....	D-9
Fault Worksheet #4 – Incomplete Boot to Solaris Operating	
System.....	D-11
Fault Worksheet #5 – Login Problem.....	D-13
Fault Worksheet #6 – adb Macro Error .....	D-15
Fault Worksheet #7 – Feckless.....	D-17
Fault Worksheet #8 – System Hangs During Boot.....	D-19
Fault Worksheet #9 – Turn the Page .....	D-21
Fault Worksheet #10 – Login Problem.....	D-23
Fault Worksheet #11 – Admintool Problem.....	D-25
Fault Worksheet #12 – Common Desktop Environment	
Problem .....	D-27
Fault Worksheet #13 – Shutdown During CDE Startup.....	D-29
Fault Worksheet #14 – Network Printer Problem .....	D-31
Fault Worksheet #15 – Boot Failure.....	D-33
Fault Worksheet #16 – Constant Reboot Problem.....	D-35
Fault Worksheet #17 – The ps Command Returns Nothing....	D-37
Fault Worksheet #18 – NIS or NIS+ Network Problem.....	D-39
Fault Worksheet #19 – Network Problem .....	D-41
Fault Worksheet #20 – No CDE Login Screen .....	D-43

---

Fault Worksheet #21 – Banner Logo Has Been Changed .....	D-45
Fault Worksheet #22 – Do Not Tread on Me.....	D-47
Fault Worksheet #23 – vi Editor Problem.....	D-49
Fault Worksheet #24 – Cracker Intrudes the System .....	D-51
Fault Worksheet #25 – No Common Desktop Environment ...	D-53
Fault Worksheet #26 – Login Problem.....	D-55
Fault Worksheet #27 – ASCII Terminal Goes Blank .....	D-57
Fault Worksheet #28 – No Network.....	D-59
Fault Worksheet #29 – Where It Is At .....	D-61
Fault Worksheet #30 – Faulty CD-ROM .....	D-63
Fault Worksheet #31 – See It Now, SPARC 5 Example .....	D-65
Fault Worksheet #31 – See It Now, Ultra Example .....	D-69
Fault Worksheet #32 – Cannot Identify Root.....	D-73
Fault Worksheet #33 – No Network or Interface.....	D-75
Fault Worksheet #34 – Script Hangs the System .....	D-77
Fault Worksheet #35 – No shcat.....	D-81
Fault Worksheet #36 – Login Problem.....	D-83
Fault Worksheet #37 - Client-Server ftp Problem.....	D-85
Fault Worksheet #38 – Network Problem .....	D-87
Fault Worksheet #39 - Slow and Fast Perceptions .....	D-89
Fault Worksheet #40 – User Application Problems .....	D-97
Fault Worksheet #41 – SunSolve Workshop .....	D-99
Fault Worksheet #42 – Let Me In .....	D-102
Fault Worksheet #43 – File Transfer Protocol Unavailable....	D-104
Fault Worksheet #44 – Slow NFS Server .....	D-106
Fault Worksheet #45 – System Unavailable to Users.....	D-108
Fault Worksheet #46 – Cannot Talk to Machine A.....	D-110
Fault Worksheet #47 – Not on This Network .....	D-112
Fault Worksheet #48 – Do Not Point at Me.....	D-114
Fault Worksheet #49 – Resource Temporarily Unavailable...	D-116
Fault Worksheet #50 – Student Designed Workshop .....	D-121

## *About This Course*

---

### *Course Goal*

The primary objective of this course is to teach students a systematic fault analysis technique to troubleshoot intermediate and advanced Solaris™ system faults.

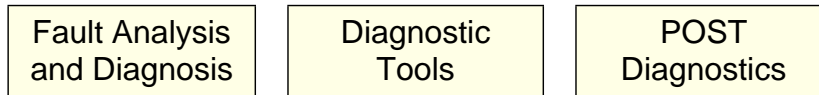
This course is intended for system administrators and support engineers who maintain systems and isolate system faults generating from a broad spectrum of software and hardware related causes. It may also be helpful to programmers who need to provide technical support for programming endeavors on Solaris systems.

---

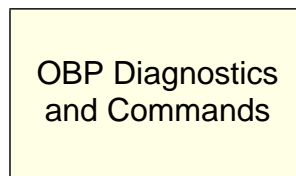
## Course Map

The following course map for this course enables you to see what you have accomplished and where you are going in reference to the course goal.

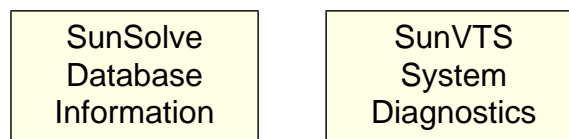
### General Methods, Tools, and Files



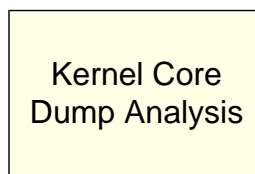
### OpenBoot PROM



### Software Packages and Services



### Crash Dump Analysis





## Module-by-Module Overview

- Module 1 – Fault Analysis and Diagnosis
- Module 2 – Diagnostic Tools
- Module 3 – POST Diagnostics
- Module 4 – OBP Diagnostics and Commands
- Module 5 – SunSolve Database Information
- Module 6 – SunVTS System Diagnostics
- Module 7 – Kernel Core Dump Analysis

### *Module-by-Module Overview*

- Module 1 – Fault Analysis and Diagnosis

This module defines the fault analysis method which is used throughout the class. It lays the foundation for all of the workshops as well as for the laboratory exercises.

Lab exercise – Apply the fault analysis method in a small group session.

- Module 2 – Diagnostic Tools

Standard Solaris utilities and files that are helpful in fault analysis are presented and discussed in this module. The use of certain tools not covered through class prerequisite work is expanded with working examples.

Lab exercise – Complete tasks that require the application of tools presented in the module.

---

## *Module-by-Module Overview*

- Module 3 – POST Diagnostics

This module explains the functionality and application of power-on self-tests in relation to fault analysis. The configuration and method of performing remote diagnostics is also presented.

Lab exercise – A remote diagnostic session, which tutorially presents configuration and diagnostic steps, is simulated.

- Module 4 – OBP Diagnostics and Commands

The environment available through OpenBoot™ programmable read-only memory (PROM) is explored with a focus on diagnostic tools that are helpful in fault analysis. Architecture dependencies across Ultra™ and SPARC™ 5 stations are included, and a thorough examination of the boot process is provided.

- Module 5 – SunSolve Database Information

This module develops the skills needed to install and fully use the SunSolve™ software accessed by CD-ROM or by the web site. Some useful SunSolve documents are listed, and the management of patch utilities is presented.

Lab exercise – Use the SunSolve software from a classroom server to locate information and to solve one of the class workshops.

- Module 6 – SunVTS System Diagnostics

The installation and use of the comprehensive SunVTS™ software is discussed in this module. Each major menu with the graphic SunVTS interface is explained.

Lab exercise – Install SunVTS software and perform selected test sequences suitable for the classroom lab equipment.

---

## *Module-by-Module Overview*

- **Module 7 – Kernel Core Dump Analysis**

This module differentiates the system conditions of panic and hang. Usage of the `crash` and `adb` debuggers is explained, and procedures for analyzing panics and hang conditions are presented.

Lab exercise – Analyze a core dump in the lab, and complete two workshops, one involving analysis of a hung system, and one involving the use of a debugger to solve a system problem.

---

## *Course Objectives*

Upon completion of this course, you should be able to

- Define a comprehensive fault analysis method
- Solve problems using the fault analysis method
- Differentiate hardware and software problems
- Use a comprehensive set of Solaris tools to analyze and solve problems
- Interpret power-on self tests (POST) diagnostic output
- Perform remote diagnostics using the Terminal Interface Protocol (TIP) interface
- Maintain and troubleshoot system problems using OpenBoot™ PROM (OBP) commands
- Look up technical information in the SunSolve database
- Analyze hardware problems using SunVTS system diagnostics
- Perform an initial crash dump analysis
- Analyze hung systems



## Skills Gained by Module

The main skills for *Sun Systems Fault Analysis Workshop* are shown in column 1 of the matrix below. The black boxes indicate the main coverage for a topic; the gray boxes indicate the topic is briefly discussed.

Skills Gained	Module						
	1	2	3	4	5	6	7
Define a comprehensive fault analysis method	Black	Gray	Gray	Gray	Gray	Gray	Gray
Solve problems using the fault analysis method	Black	Gray	Gray	Gray	Gray	Gray	Gray
Differentiate hardware and software problems	White	Black	Gray	Gray	Gray	Gray	Gray
Use a comprehensive set of Solaris tools to analyze and solve problems	White	Black	Gray	Gray	Gray	Gray	Gray
Interpret POST diagnostic output	White	White	Black	White	White	White	Gray
Perform remote diagnostics using the TIP interface	White	White	Black	White	White	White	Gray
Maintain and troubleshoot system problems using OBP commands	White	White	White	Black	White	White	Gray
Look up technical information in the SunSolve database	White	White	White	White	Black	White	White
Analyze hardware problems using SunVTS system diagnostics	White	White	White	White	White	Black	White
Perform an initial crash dump analysis	White	White	White	White	White	White	Black
Analyze hung system	White	White	White	White	White	White	Black

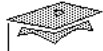
---

## Guidelines for Module Pacing

The following table provides a rough estimate of pacing for this course:

Module	Day 1	Day 2	Day 3	Day 4	Day 5
About This Course	A.M.				
Module 1 – Fault Analysis and Diagnosis	A.M.				
Module 2 – Diagnostic Tools	A.M./ P.M.				
Selected Workshops	P.M.				
Module 3 – POST Diagnostics		A.M.			
Module 4 – OBP Diagnostics and Commands		A.M.			
Selected Workshops		P.M.			
Module 5 – SunSolve Database Information			A.M.		
Module 6 – SunVTS System Diagnostics			A.M.		
Selected Workshops			P.M.		
Module 7 – Kernel Core Dump Analysis				A.M./ P.M.	
Selected Workshops				P.M.	A.M./ P.M.

---



## Topics Not Covered

- Basic system administration
  - Software installation
  - User account configuration
  - Printer management
  - Basic security policies

### *Topics Not Covered*

This course does not cover the following topics. Many of the topics are covered in other courses offered by Sun Educational Services.

- Basic system administration topics – Covered in SA-235 and SA-237, *Solaris System Administration I*
  - ▼ Software installation
  - ▼ User account configuration
  - ▼ Printer management
  - ▼ Basic security policies



## Topics Not Covered

- Advanced system administration topics
  - Device naming conventions
  - NFS™, *cachefs* and automounter administration
  - NIS/NIS+ administration
  - Disk and file system configuration
  - Network configuration
  - The SAF utility

### *Topics Not Covered*

- Advanced system administration topics – SA-286 and SA-287, *Solaris System Administration II*
  - ▼ Device naming conventions
  - ▼ NFS™, *cachefs* and automounter administration
  - ▼ NIS/NIS+ administration
  - ▼ Disk and file system configuration
  - ▼ Network configuration
  - ▼ The SAF utility

Refer to the Sun Educational Services catalog for specific information and registration.



## How Prepared Are You?

- Have you completed
  - SA-235: *Solaris 2.x System Administration I* or SA-237: *Solaris 7 System Administration I*
  - SA-286: *Solaris 2.x System Administration II* or SA-287: *Solaris 7 System Administration II*
  - SA-271: *Solaris 1.x to 2.x System Administration*

or

- Do you have six months of field system administration or system maintenance experience in Sun environments?

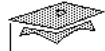
## *How Prepared Are You?*

To be sure you are prepared to take this course, can you answer one of the following questions in the affirmative?

- Have you completed SA-235: *Solaris 2.x System Administration I*, or SA-237: *Solaris 7 System Administration I*, and SA-286: *Solaris 2.x System Administration II* or SA-287: *Solaris 7 System Administration II*, or SA-271: *Solaris 1.x to Solaris 2.x System Administration*?

or

- Do you have at least six months of field system administration or system maintenance experience in Sun™ environments?

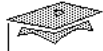


## Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Troubleshooting experience
- Fault analysis and diagnosis experience
- Application tools experience
- Reasons for enrolling in this course
- Expectations for this course

### *Introductions*

Now that you have been introduced to the course, introduce yourself to each other and the instructor, addressing the items shown on the above overhead.



## How to Use Course Materials

- Objectives
- Relevance
- Overhead image
- Lecture
- Exercise
- Check your progress
- Think beyond

### *How to Use Course Materials*

To enable you to succeed in this course, these course materials employ a learning model that is composed of the following components:

- **Objectives** - Each module begins with a list of objectives that represent the skills you acquire by completing the module and the associated laboratory exercises.
- **Relevance** – The relevance section for each module provides scenarios or questions that introduce you to the information contained in the module and provoke you to think about how the module content relates to your interest analyzing and diagnosing system faults.
- **Lecture** – The instructor will present information specific to the topic of the module. This information will help you learn the knowledge and skills necessary to succeed with the exercises.

---

## *How to Use Course Materials*

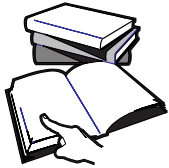
- **Exercise** – Lab exercises will give you the opportunity to practice your skills and apply the concepts presented in the lecture.
- **Check your progress** – Module objectives are restated, sometimes in question format, so that before moving on to the next module you are sure that you can accomplish the objectives of the current module.
- **Think beyond** – Thought-provoking questions are posed to help you apply the content of the module or predict the content in the next module.



---

## Course Icons

The following icons and typographical conventions are used in this course to represent various training elements and alternative learning resources:



**Additional resources** – Indicates additional reference materials are available.



**Discussion** – Indicates a small-group or class discussion on the current topic is recommended at this time.



**Exercise objective** – Indicates the objective for the lab exercises that follow. The exercises are appropriate for the material being discussed.

---

**Note** – Additional important, reinforcing, interesting or special information.

---



---

**Caution** – A potential hazard to data or machinery.

---



---

**Warning** – Anything that poses personal danger or irreversible damage to data or the operating system.

---

---

## Typographical Conventions

Courier is used for the names of command, files, and directories, as well as on-screen computer output. For example:

```
Use ls -al to list all files.  
system% You have mail.
```

**Courier bold** is used for characters and numbers that you type. For example:

```
system% su  
Password:
```

*Courier italic* is used for variables and command-line placeholders that are replaced with a real name or value. For example:

To delete a file, type `rm filename`.

*Palatino italics* is used for book titles, new words or terms, or words that are emphasized. For example:

Read Chapter 6 in *User's Guide*.  
These are called *class* options  
You *must* be root to do this.

---

## *Attention VARs*

If you are an Sun Microsystems Computer Corporation authorized reseller taking this course for Competency 2000 certification credit, the Drake tests and certification specifications for this course revision are being upgraded.

- Retain your signed course certificate.
- Schedule yourself to take the appropriate Drake test by contacting:

John Shedaker  
Sun Microsystems  
2550 Garcia Ave., MS UMIL06-01  
Mountain View, CA 94043

Fax Number (408)-945-9476  
Phone Number (408)-276-1315



## *Objectives*

Upon completion of this module, you should be able to

- Use an organized total system approach for fault analysis and diagnosis
- Write accurate problem statements
- Describe a system problem in terms of error messages, symptoms, relative comparisons and technical conditions
- Identify and use commonly available resources to solve technical problems
- Generate and test a list of likely causes on a per fault basis
- Communicate and document information gathered during fault analysis
- Use the Fault Analysis Worksheet to gather and document facts

## Relevance



**Discussion** – This module establishes the two-part method that is used for fault analysis throughout the class. Analytical and diagnostic procedures are thoroughly explained. A laboratory session provides the context for applying this method in a group workshop format.

Before beginning Module 1, consider the following questions in relation to the work you currently do on your Solaris operating systems:

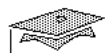
- Do you have a technical problem-solving procedure in place at your company?
- Do you document information related to technical problems that occur on systems?
- Do you maintain system logs for any or all of your Solaris systems?
- What tools have you found useful in troubleshooting system problems?
- Have you interviewed a user or customer to clarify a technical problem?

## References



**Additional resources** – The following reference can provide additional details on the topics discussed in this module:

- *Alamo Learning Systems AdvantEdge Analysis Program*



Sun Educational Services

## The Fault Analysis Method

- Fault analysis – Identify the problem and organize fact gathering and comparisons
- Diagnosis – Organize the actual discovery, testing, repair, and reporting of the problem

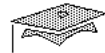
### *The Fault Analysis Method*

Fault analysis and diagnosis is an efficient and reliable method to isolate and repair Sun™ system faults using a two-stage process:

- *Fault analysis* – Identify the problem and organize fact gathering and comparisons
- *Diagnosis* – Organize the actual discovery, testing, repair, and reporting of the problem

You may or may not be an expert. With the fault analysis method developed here, you gather data and use your experience and the experience of others to determine causes. Fault analysis and diagnosis provides you with a powerful tool to analyze data and focus on the likely causes of a complex problem or a problem outside of your immediate experience.

Keeping notes in the fault analysis format enhances communication about the status of a problem and provides a source of information for future system work.



## Eight Steps of Fault Analysis and Diagnosis

1. State the problem.
2. Describe the problem.
3. Identify differences.
4. List relevant changes.
5. Generate likely causes.
6. Test likely causes.
7. Verify the most likely cause.
8. Take action to correct the fault.

### *Eight Steps of Fault Analysis and Diagnosis*

Fault analysis and diagnosis consists of eight main steps. The first steps deal with fault analysis, the latter four steps are diagnostic.

#### *Fault Analysis*

The steps taken are:

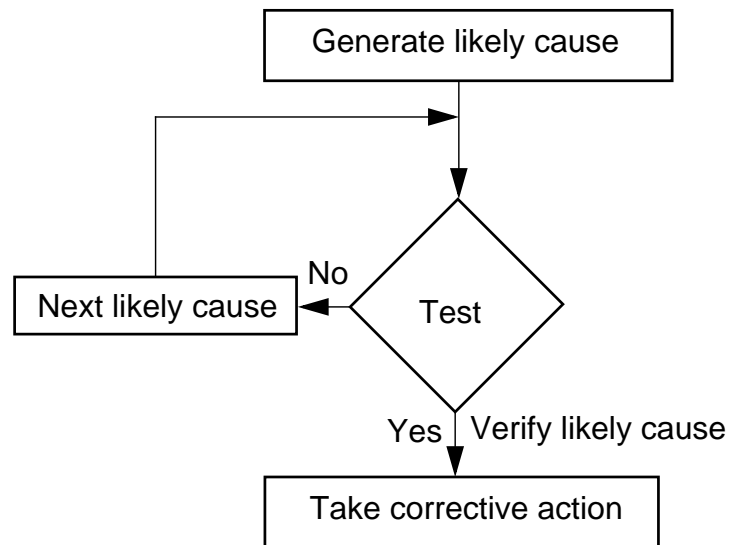
1. State the problem.
2. Describe the problem.
3. Identify differences.
4. List relevant changes.



## *Eight Steps of Fault Analysis and Diagnosis*

### *Diagnosis*

5. Generate likely causes.
6. Test likely causes.
7. Verify the most likely cause.
8. Take action to correct the fault.



**Figure 1-1** Steps for Fault Analysis and Diagnosis



## Stating the Problem

- Writing a problem statement
  - Determine which object, device, or subsystem exhibits the problem
  - Hypothesize what is wrong
- Guidelines for a problem statement
  - Identify the exact object with the exact defect
  - Be certain that the cause is not already known
  - Limit the problem statement to a single object and a single defect

### *Stating the Problem*

#### *Writing a Problem Statement*

Given a system problem, identify the object and its defect, and write a problem statement. A problem statement answers these questions:

- What object, device, or subsystem exhibits the problem?
- What is wrong? What is the defect or deviation from the standard?

The following is an example of a problem statement:

The printer Grumpy will not print.

The object, printer Grumpy, has a defect (deviation from the standard)—it will not print.

---

## *Stating the Problem*

### *Guidelines for a Problem Statement*

In your problem statement,

- Identify the exact object with the exact defect
- Be certain that the cause is not already known
- Limit the problem statement to a single object and a single defect

---

**Note** – Most bugs that become a disaster happen because the original problem is not identified correctly.

---



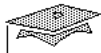
## Describing the Problem

- List all observed facts
- Establish comparative facts
- Identify the unique elements of the problem
- List any explicit error messages displayed
- Describe conditions and symptoms
- Identify any timely and related changes on the system

### *Describing the Problem*

The next step in system fault analysis and diagnosis is to describe the problem in detail.

- List all observed facts.
- Establish comparative facts. To help isolate the likely cause, identify what works properly as well as what does not work. This helps to establish what the problem is not.
- Identify the unique elements of the problem.
- List any explicit error messages displayed.
- Describe conditions and symptoms that are manifest with the problem.
- Identify any timely and related changes on the system.



## Listing All Observed Facts

- Who observed the problem?
- What is the problem?
- Where is the problem observed?
- What is the magnitude or size of problem?
- Are there any other questions?

## *Describing the Problem*

### *Listing All Observed Facts*

Some recommended questions to ask at this point are:

- Who observed the problem?
- What is the problem?
- Where is the problem observed?
- What is the magnitude or size of problem?

You can expand and customize a question list for your own environment.



## Listing All Observed Facts

- Customer complaints
- Customer interviews
- Interviews of others involved
- Diagnostics
- Dumps
- Other

### *Describing the Problem*

#### *Listing All Observed Facts*

Identify the sources of the observed facts you listed.

- Customer complaints – Use the original message from the customer.
- Customer interviews – Use the list of questions shown previously to interview customers about the problem. Expand and customize the question list for your own style and environment
- Interviews of others involved – Include other colleagues such as administrators, programmers, and technical support staff.
- Diagnostics – Consider changed environments and operating system levels
- Dumps – Evaluate the results of crash analysis if a core file is generated and available.

---

## *Describing the Problem*

### *Listing All Observed Facts (Continued)*

What other sources do you know of?

- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_



## Establishing Comparative Facts

- Do other systems with similar OS revision, architecture, configuration have this problem?
- What parts of system are functional?
- When was the problem first observed?
- Was the problem observed at the same time on more than one system?
- What events occurred in the environment that may have contributed to the problem?
- Are the systems used similar enough to compare?

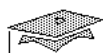
## *Describing the Problem*

### *Establishing Comparative Facts*

In describing the problem, it is helpful to compare the problem system to other systems in your environment.

- Do other systems with similar operating system (OS) revision, architecture, and configuration have this problem?
- What parts of the system are functional?
- When was the problem first observed?
- Was the problem observed at the same time on one or more systems?
- What events occurred in the environment that may have contributed to the problem?
- Are the systems being compared similar enough to establish valid observations and conclusions?





## Identifying Differences

- Focus on one set of comparisons at a time
- List unique differences
- State the facts, not opinions
- Analyze comparisons for contrasts
- Analyze comparisons for what is the same
- Keep records

### *Identifying Differences*

Once you have listed the observable facts and established comparative facts, identify any differences that exist.

- Focus on one set of comparisons at a time.
- List unique differences within your observations and comparisons. Look for distinctions between systems, such as hardware mix, system load, surrounding temperature, patches applied, and so on.

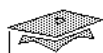
For example, what is the difference between System A (problem object) and System B (operational object)?

System A is running the Solaris 7 operating environment and the Network Information System Plus (NIS+) software, and it is installed on the network using a 10BASE-T Ethernet connection.

## *Identifying Differences*

System B is running the Solaris 7 operating environment and the Network Information System (NIS) software, and it is installed on the network using a 10BASE-5 Ethernet connection.

- State facts and differences, not opinions. For example, stating that the NIS software on System B is flawed is an opinion, but to say that the NIS software on System B is a different domain service than NIS+ is stating a fact.
- Analyze comparisons for contrasts. For example, state that System A is an NIS+ software client and that System B is an NIS software server.
- Analyze comparisons for what is the same. Many observed and comparative facts show no contrast, so just note them as no difference. This often helps to establish what the problem is not.
- Keep records. In this course, fault analysis worksheets are used. Refer to the templates at the end of this module.



## Listing Relevant Changes

- Examine differences.
- Describe each change and the date of its occurrence.
- Determine if each change represents something new or unusual about a difference.
- Determine if the problem occurred during a certain period of time.

### *Listing Relevant Changes*

A change can cause or identify a problem. The differences between observed and comparative facts can identify changes. Determine if the changes are relevant to the problem.

- Examine the differences and ask what, if anything, has changed.
- Describe each relevant change and the date or time of its occurrence. A relevant change that happened before the problem occurred is more likely to be a cause than one that happened after the problem occurred.
- Determine if each change represents something new or unusual.
- Determine if the problem occurred during a certain period of time; for example, one month before, two days after, or the same time. For example:
  - ▼ Power supply was upgraded Thursday night.
  - ▼ Administrator added four users Friday afternoon.



## Generating Likely Causes

- Stating your hypothesis  
How could the *fault analysis element* have caused this problem?
- Example
  - Problem – Slow system response
  - Relevant change – Administrator added four users on Friday
  - Hypothesis – How could adding four users on Friday have caused slow system response?

### *Generating Likely Causes*

The next step in fault analysis and diagnosis is generating likely causes.

Use differences and relevant changes to discover likely causes of the problem. Then form an hypothesis about the cause, and analyze the problem with facts, differences, and relevant changes. This helps diagnose the problem.

### *Stating Your Hypothesis*

First, state your hypothesis in the form of a question; for example:

How could the *fault analysis element* have caused this problem?

The answer, or rephrased hypothesis, is that changing *A* may cause *B*.

---

## Generating Likely Causes

### *Stating Your Hypothesis (Continued)*

For the *fault analysis element*, insert one of the following:

- A relevant change
- Two or more relevant changes
- A relevant change and a difference
- A single difference

#### *Example*

Suppose the problem is slow system response and that a relevant change is that the administrator added four users on Friday.

Your hypothesis, in question format, is

How could adding four users on Friday have caused slow system response?

The answer, or rephrased hypothesis, is

Additional users increase resource usage and processing which can result in memory, central processing unit (CPU), and disk bottlenecks, causing slow system response.

You can develop as many hypotheses as you have facts. Use your experience and judgement to prioritize the list to the most logical and likely cause(s).



## Testing Likely Causes

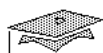
- Closely examine doubtful answers.
- Investigate assumed facts.
- Eliminate likely causes that cannot be the cause.

### *Testing Likely Causes*

Testing likely causes (identified by each of your hypothesis) is the next diagnostic step. Testing increases the probability that you can discover the actual cause of the problem before embarking on or recommending a potentially costly, time-consuming solution. In this step, you

- Closely examine any doubtful answers
- Investigate assumed facts
- Eliminate a likely cause when you are certain it cannot be the true cause of the problem

Test your hypothesis aggressively. Be careful not to change facts to support a hypothesis, especially in hurried or stressful situations. Changing facts usually makes the problem worse.



## Verification

- Factual and logical
- Realistic
- Result-oriented

### *Testing Likely Causes*

#### *Verification*

Three approaches used to verify the most likely cause of a problem are:

- *Factual and logical* – In this approach, your conclusions of likely causes are based on information gathered on the fault analysis worksheet and on past experience. This results in likely causes that make the most sense.
- *Realistic* – In this approach, the most likely cause must pass an experiment to show conclusively that it is or is not the cause. For example, try a new driver without overwriting the old one. This provides a quick, non-disruptive verification with good, but not complete, conclusiveness.

## Testing Likely Causes

### *Verification (Continued)*

- *Result-oriented* – In this approach, you assume, without proof, that the most likely cause you choose is the actual cause, and take the indicated corrective action. This is the least conclusive verification, and can be disruptive, expensive, and time-consuming, especially if your assumptions are not correct.

---

**Note** – No one approach is presented as better than another for all problems. Each approach has its strengths and weaknesses, and you must choose the approach most appropriate for your situation.

---





## Verifying the Most Likely Cause

Use a method that is

- Least disruptive
- Least expensive
- Least time-consuming
- Most conclusive

### *Verifying the Most Likely Cause*

Now you are ready to verify if the most likely cause is the actual cause of the problem. Verifying the most likely cause should remove all uncertainty about the cause of a problem.

Use the method that is

- Least disruptive
- Least expensive
- Least time-consuming
- Most conclusive



## Taking Corrective Action

1. Complete the repair.
2. Test and verify the repair.
3. Document results.
4. Obtain confirmation and acceptance.

### *Taking Corrective Action*

Taking corrective action is the final step in fault analysis and diagnosis. It consists of the following substeps:

1. Complete the repair.
2. Test and verify the repair.
3. Document results.
4. Obtain confirmation and acceptance.

---

## *Fault Analysis Worksheet Example*

The following example illustrates how the fault analysis worksheet template provided at the end of this module can be completed to help identify and correct a problem.

### *Analysis Phase*

#### *Initial Customer Description*

Machine does not boot. It had been working fine until a new member of the administrative staff worked on the system to load a driver module.

#### *Problem Statement*

Machine fails to boot; it returns to the OK prompt early in the boot procedure.

#### *Resources*

Available resources are

- The man pages on boot and kernel
- Other functioning systems
- Technical colleagues (workshop group members)
- AnswerBook™ and SunSolve databases

## *Fault Analysis Worksheet Example*

### Analysis Phase

#### *Problem Description*

<b>Error Messages</b>	<b>Symptoms and Conditions</b>	<b>Relevant Changes and/or History</b>	<b>Comparative Facts</b>
	Every attempt to boot causes the system to fail.	No hardware upgrades or configuration changes	The troublesome system boots with target 3; working systems boot with target 0
	The banner displays successfully, and it appears to successfully locate the boot device.	No software upgrades	Only the system subject to driver configuration has the boot problem.
Data access Exception	Error message occurs immediately after the Display boot device message. Then the system fails, issuing the OK prompt.	Administrative work done in kernel directory modules just prior to the incidence of the boot problem.	Other machines in this department do not have this boot problem.

## Fault Analysis Worksheet Example

### Diagnostic Phase

#### Test and Verification

Likely Cause(s)	Test(s)	Result	Verification
Wrong boot-device setting in programmable read-only memory (PROM)	Use <code>probe-scsi</code> or <code>probe-ide</code> and <code>printenv</code> to validate default boot setting. Check that the drive is on-line	PROM settings are correct.	NA
Missing or wrong <code>ufsboot</code> program, <code>bootblk</code>	Boot from the CD-ROM to compare <code>ufsboot</code> file with a functioning system. Use <code>sum</code> and <code>ls -l</code> for checksum, and file modes.	The <code>ufsboot</code> file is the correct one for this architecture.	NA
Missing or wrong kernel core driver files	Compare <code>/kernel</code> and <code>/usr/kernel</code> directories and files on malfunctioning and on a functioning system.	System has <code>unix</code> in <code>/kernel/</code> ; should have <code>genunix</code> . Rename the file to <code>/kernel/genunix</code> .	Boot the system with this repair and check for error messages. See if the system completely boots.

#### Corrective Action

Final Repair	Communication	Documentation
Name the UNIX <sup>®</sup> file and situate properly as <code>/kernel/genunix</code> .	Inform the administrator of the naming convention for the UNIX file on Solaris. Explain to users and managers as well.	Update the log files that are maintained with the system, including initials and timestamp.

## *Exercise: Conducting Fault Analysis and Diagnosis*



**Exercise objective** – The objective of this exercise is to apply the methods and steps for fault analysis and diagnosis.

### *Preparation*

Your instructor will coordinate the creation of workgroup units for the exercise. The instructor's role is that of the customer or the user; you can ask the instructor questions about the problem.

### *Tasks*

As a group, solve Workshop #46 in Appendix D. Use the methods discussed in this module to analyze and diagnose the problem.

---

# *Fault Analysis Worksheet Template*

## *Analysis Phase*

### *Initial Customer Description*

### *Problem Statement*

### *Resources*

### *Problem Description*

<b>Error Messages</b>	<b>Symptoms and Conditions</b>	<b>Relevant Changes</b>	<b>Comparative Facts</b>

## *Fault Analysis Worksheet Template*

### Diagnostic Phase

#### *Test and Verification*

<b>Likely Cause(s)</b>	<b>Test(s)</b>	<b>Result(s)</b>	<b>Verification(s)</b>

#### *Corrective Action*

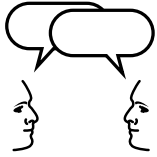
<b>Final Repair</b>	<b>Communication</b>	<b>Documentation</b>



---

## *Exercise: Conducting Fault Analysis and Diagnosis*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

## *Check Your Progress*

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Use an organized total system approach for fault analysis and diagnosis
- Write accurate problem statements
- Describe a system problem in terms of error messages, symptoms, relative comparisons, and technical conditions
- Identify and use commonly available resources to solve technical problems
- Generate a list of likely causes on a per fault basis
- Design and implement tests to validate likely causes
- Communicate and document information gathered in fault analysis
- Use the Fault Analysis Worksheet to gather and document facts

---

## *Think Beyond*

Can you describe a way to improve the current fault analysis procedure used in your company environment? What on-line resources are available at your company site?



### *Objectives*

Upon completion of this module, you should be able to

- Differentiate watchdog resets, panics, and system hangs
- Differentiate hardware and software problems
- Provide examples of fatal and non-fatal error conditions
- Identify a comprehensive set of Solaris commands and utilities which are useful in fault analysis
- Identify a comprehensive list of Solaris system files which contain information that is useful in fault analysis
- Describe the syntax, function, and relevance of each command or system file
- Use Solaris commands and files to determine system configuration and status information
- Solve workshop problems using Solaris utilities and system files

## Relevance



**Discussion** – This module presents a comprehensive set of fault analysis tools which are readily available on the Solaris operating system. The usefulness of each tool is explained, and examples that demonstrate usage are provided. Expand the list with additional tools that you know of, or combination uses of the tools presented here.

Before beginning Module 2, consider the following questions related to your own experience with the Solaris operating system:

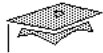
- What tools and utilities do you currently find most useful?
  - ▼ \_\_\_\_\_
  - ▼ \_\_\_\_\_
  - ▼ \_\_\_\_\_
  - ▼ \_\_\_\_\_
- Are there operations you would like to be able to perform on a Solaris platform but have not been able to?
- What tools are you most interested in learning?
- Can you describe a recent Solaris system problem and how it was solved?

## References



**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- *Solaris User and System Administration AnswerBooks*  
(<http://docs.sun.com>)
- Solaris man pages
- The SunSolve database



*Sun Educational Services*

## Introduction to Hardware and Software Errors

- Error categories – Software, hardware-corrected, recoverable, fatal, and critical
- Error reporting mechanisms – Bus errors, interrupts, and resets

### *Introduction to Hardware and Software Errors*

The fundamental error detection mechanisms for all architectures are basically the same. As architecture designs became more complex, the error detection mechanisms became more sophisticated, specifically in the multiprocessor environment.

This module begins by classifying the fundamental error detection mechanisms and providing examples of errors that represent each major fault category.

## *Introduction to Hardware and Software Errors*

### *Error Categories*

Error categories include

- Software errors
- Hardware-corrected errors
- Recoverable errors
- Fatal errors
- Critical errors

### *Error Reporting Mechanisms*

Error reporting mechanisms include

- Bus errors
- Interrupts
- Resets





## Types of Errors

- Software errors
- Hardware-corrected errors
- Recoverable errors
- Critical errors
- Fatal errors

### *Type of Errors*

#### *Software Errors*

Errors that do not originate in the hardware are classified as software errors. All such errors are detected by the processor and are reported. Examples of software errors are programming errors or bugs in the kernel code.

#### *Hardware-Corrected Errors*

For error-logging purposes, hardware-corrected errors are always signaled by an interrupt. No recovery action is normally required. One bit error from memory is corrected by the error checking and correcting (ECC) logic. This is reported in the error log.

## *Type of Errors*

### *Recoverable Errors*

Recoverable errors caused by hardware are usually signaled by a bus error posted to the requesting device and a specified interrupt, which could broadcast the error. Error recovery in such cases is normally handled by the trap routines, while error logging is done by the interrupt handler.

The contents of trap registers can be examined, using `adb` and the OBP register commands, to determine the trap type. Software trap types are defined in `/usr/include/sys/trap.h`; hardware trap types are defined in `machtrap.h` which is located in `/usr/include/<version_number>/sys`.

A non-essential device losing power or becoming inaccessible is an example of a recoverable error.

### *Critical Errors*

Critical errors require immediate attention, system shutdown, and power-off. They are notified through a high-level broadcast interrupt if at all possible. Types of critical errors include

- An alternating current/direct current (AC/DC) failure
- Temperature warning
- Fan failure

### *Fatal Errors*

A fatal error is a hardware error in which proper system operation cannot be guaranteed. All fatal errors initiate a system-watchdog reset. Parity errors on backplanes are an example of a fatal error.



## Error Reporting Mechanisms

- Bus errors
- Interrupts
- Watchdog resets
  - CPU watchdog reset
  - System watchdog reset

### *Error Reporting Mechanisms*

#### *Bus Errors*

Bus errors are one of the mechanisms for error reporting on the system. Bus errors are issued to the processor when the processor references a virtual or physical location that cannot be satisfied for hardware reasons. Some typical bus errors that occur are

- Illegal address or internal hardware failure
- Instruction fetch or data load
- On an SBus, direct virtual memory access (DVMA) operations
- Synchronous/asynchronous data store
- Memory management unit (MMU) operations

## *Error Reporting Mechanisms*

### *Interrupts*

Interrupts are another mechanism for error reporting. These are device dependent and are issued to notify the CPU of external device conditions that are asynchronous with the normal operation.

Interrupts indicate

- Device done or ready
- Error detected (the action taken depends on the device driver)
- Change in power status

---

## *Watchdog Resets*

A watchdog reset is one of the most difficult computer failures to diagnose. It is an error mode that attempts to bring the system to a well-known (*deterministic*) state. It represents a response by the system to a fault condition which is deemed potentially dangerous.

As a safety precaution, the system goes immediately down without taking a core dump. The absence of the core dump file makes the watchdog reset condition more difficult to analyze than the case of a panic. Often, only OBP register commands are available to discern the cause of the fault. Systems with sun4u and sun4d architectures can also use `prtdiag -v`.

### *CPU Watchdog Reset*

A CPU watchdog reset is initiated on a single processor machine when a trap condition occurs while traps are disabled and a register bit to enable traps is not set. Because this condition is related to a register setting, it is referred to as a CPU watchdog reset.

The CPU branches to a reserved physical address, and system goes immediately down. The causes for CPU watchdog resets can be either hardware or software, and usually affect only one CPU.

### *System Watchdog Reset*

When a fatal error is detected on a multiprocessor machine, a system watchdog reset is initiated. A system watchdog reset affects all CPUs and I/O devices. Writes in progress may be lost, but the state of main memory is not altered and continues to be refreshed after a system watchdog reset. In most cases, the system watchdog reset condition is hardware related.



## Differentiating Hardware and Software Faults

- Fault frequency
- History
- Stack trace routines
- Watchdog resets
- Log files
- Panics

### *Differentiating Hardware and Software Faults*

Observation of the following details assist in differentiating hardware from software problems when panic or watchdog resets occur:

- *Fault frequency* – Hardware problems tend to come on suddenly, be more random, and become aggravated as time goes on, while software problems tend to more predictable and often repeatable.
- *History* – Traumatic events such as frequent power failures, dropping hardware or mishandling can lead to hardware problems.

---

## *Differentiating Hardware and Software Faults*

- *Watchdog resets* – Usually, (not always) watchdog resets are hardware related.
- *Log files* – Related messages in `/var/adm/messages` are checked. Sometimes hardware and software failures are prefaced by messages written to the log file. If there are disk error messages in the log file, and UNIX<sup>®</sup> file system (UFS) routines listed on a stack trace, it is likely that there is a disk hardware problem.
- *Panics* – Often, system panics can originate from software, however, it is possible to incur a panic from a hardware fault. Some panic messages that indicate hardware problems include
  - ▼ *Asynchronous memory error* – Indicates a memory problem
  - ▼ *Asynchronous memory fault* – Usually indicates a bus problem between memory and CPU

The remainder of this module discusses diagnostic commands, tools, and files.

## Diagnostic Commands and Tools

**Table 2-1** Diagnostic Commands and Tools

Command or Tool	Use
adb	Analyze dumps and a running system.
AnswerBook	Display online reference manuals in areas of hardware, user, system administration, and other
arch	Display architecture and kernel architecture information.
arp	Display the Address Resolution Protocol tables
aset	Use the Automated Security Enhancement Tool
catman -w	Create the <code>/usr/share/man/windex</code> database for use with index function available through the <code>apropos</code> command
crash	Analyze crash dumps
devlinks	Create symbolic links in <code>/dev</code> using information in <code>/devices</code>
df -k	Display disk space usage in Kbytes, including free space
dfmounts	Display remote file system mount information
dfshares	Display shared file system information
diff	Compare file contents
dmesg	Analyze recent log messages
disks	Create symbolic links in <code>/dev/dsk</code> and <code>/dev/rdsk</code>
drvconfig	Configure the devices directory and the device information tree
eeprom	Analyze and change programmable read-only memory (PROM) settings
file	Determine a file's type
find	Search for specific files in the file system structure
format	Analyze or modify disk partition information
fsck	Check UFS file systems for inconsistencies
fsdb	Use file system debugger (see <code>fsdb_ufs</code> in man pages)
fstyp	Display extensive file system parameters for a specified file system



**Table 2-1** Diagnostic Commands and Tools (Continued)

Command or Tool	Use
grep	Analyze file contents, and search for specific patterns
groups	Display group definitions for a given user
grpck	Check the <code>/etc/group</code> file for syntax errors or inconsistencies
ifconfig	Analyze the status of network interfaces
infocmp	Compare <code>tic</code> formatted files
iostat	Analyze I/O performance issues
kadb	Trap kernel and low-level faults
last	Display history of system login information
ls	Analyze file properties
ndd	Get and set named device driver parameters
netstat (-i, -r)	Analyze network tuning information, including active routes.
newfs	Create and examine file system parameters
nfsstat	Analyze NFS™ performance information
nm	Display symbol table information for (unstripped) executables (in <code>/usr/ccs/bin</code> )
pagesize	Print the size of a memory page in bytes
perfmeter	Provide a graphic display of performance metrics
ping	Contact network hosts by sending Internet Control Message Protocol (ICMP) request and reply datagrams
pkgchk	Check file integrity and accuracy of installation
prtdiag (on sun4u and sun4d)	Display system configuration and diagnostic information (kept in <code>/usr/platform/`uname -m`/sbin</code> )
prtconf -v	Get system device information from POST probe
prtconf -vp	Display PROM version (OBP)
ps	List properties of running processes ( <code>/bin/ps</code> , <code>/usr/ucb/ps</code> )
pwck	Check the <code>/etc/passwd</code> file for errors and inconsistencies
route	Add, remove, and display kernel route table information

**Table 2-1** Diagnostic Commands and Tools (Continued)

Command or Tool	Use
<code>rpcinfo</code>	Display information about Remote Procedure Call (RPC) services
A running system	Compare symptoms and results between functioning and faulting systems
<code>sar</code>	Analyze system performance information (must be initialized in <code>/etc/init.d/perf</code> )
<code>shells</code>	Use <code>-x</code> and <code>-v</code> options to provide debugging information
<code>showrev -p</code>	List currently installed patches; <code>patchadd -p</code> in Solaris 2.6 and above
<code>snoop</code>	Display and analyze network traffic
<code>strings</code>	Search object and binary files for ASCII strings
SunSolve database	List bug, patch, release, and general technical information for hardware and software
<code>sysdef</code>	Analyze device and software configuration information
<code>swap</code>	Add, delete, and monitor system swap areas
<code>sum</code>	Calculate and print a checksum value for a named file
SyMON™	Use system monitor utility package with interfaces to the diagnostics package, <code>SUNvts</code>
<code>sys_unconfig</code>	Enable you to change information entered during <code>sysidtool</code> phase of installation
<code>tail -f</code>	Leave file open for reading and display what is there
<code>tapes</code>	Create logical links to device special files in <code>/devices</code> for tapes
<code>tic</code>	Terminfo compiler; translate a terminfo file from the source format into the compiled format
<code>timex</code>	List runtime and system activity information during command execution
<code>traceroute</code>	Show the routes followed by packets transferred in a subnetted environment
<code>truss</code>	Trace system calls issued and used by a program or command
<code>tunefs</code>	Modify file system parameters that affect layout policies

**Table 2-1** Diagnostic Commands and Tools (Continued)

<b>Command or Tool</b>	<b>Use</b>
<code>/usr/proc/bin/*</code>	Use utilities that exercise the functions of the <code>/proc</code> file system
<code>uname</code>	Print platform, architecture, operating system, and system node information
<code>vmstat</code>	Analyze memory performance statistics
<code>who am i</code>	Display the effective current user name, terminal line and login time

## *Open Discussion*

What other tools have you used that could be added to the list?

- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

## Diagnostic Files

**Table 2-2** Diagnostic Files

File	Contents and Use
/etc/defaultdomain	Name the current domain; read and set it at each boot by the script /etc/init.d/inetinit
/etc/default/cron	Determine logging activity for the cron daemon through specification of the <i>cronlog</i> variable
/etc/default/login	Control root logins at the console through specification of the <i>console</i> variable, and other defaults
/etc/default/su	Determine /etc/hostname.1e0 logging activity for the su command through specification of the <i>su</i> log variable
/etc/dfs/dfstab	List which distributed file systems will be shared at boot time
/etc/dfs/sharetab	List currently shared NFS files and directories
/etc/hosts	Associate Internet Protocol (IP) addresses to particular system names (symbolically linked to /etc/inet/hosts)
/etc/hostname.1e0 /etc/hostname.hme0	Assign a system name, and through cross-referencing the /etc/hosts file, add an IP address to a particular network interface
/etc/inetd.conf	List information for network services that can be invoked by the inetd daemon
/etc/inittab	Read by init daemon at startup to determine which rc scripts to execute; also contains default run level
/etc/minor_perm	Specify permissions to be assigned to device files
/etc/mnttab	Display a list of currently mounted file systems
/etc/name_to_major	Display a list of configured major device numbers
/etc/netconfig	Display the network configuration database read during network initialization and use
/etc/netmasks	Display the netmask value; read at boot time during network initialization
/etc/nsswitch.conf	List the database configuration file for the name service switch engine

**Table 2-2** Diagnostic Files (Continued)

<b>File</b>	<b>Contents and Use</b>
/etc/path_to_inst	List the contents of the system device tree using the format of physical device names and instance numbers
/etc/protocols	List known protocols used in conjunction with Internet
/etc/rmtab	List current remotely mounted file systems
/etc/rpc	List available RPC programs
/etc/services	List well-known networking services and associated port numbers; maintained by the Network Information Center (NIC)
/etc/system	Use for setting tunable kernel parameters
/etc/vfstab	List local and remote file systems mounted at boot time
/var/adm/messages	List recent console window and/or boot messages
/var/adm/sulog	Display a record for each invocation of the su command
/var/adm/utmpx	List user and accounting information for the who and login commands
/var/adm/wtmpx	Maintain history of user information for the accounting package and report facility
/var/lp/log	List print services activity
/var/sadm/install/ contents	List installed software packages
/var/saf/_log	List activity of the Service Access Facility (SAF)

---

## *Open Discussion*

What other files have you used that could be added to the list?

- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_



## Applying the Tools

- Crash debuggers
- SunSolve databases
- OpenBoot™ PROM (OBP) commands

### *Applying the Tools*

On the next several pages are examples of the syntax and usage of many of the less well-known tools listed in Table 2-1. The following items assist in locating information about how to use these tools:

- The `adb` and `crash` debuggers are explained in Module 7, “Kernel Core Dump Analysis.”
- The SunSolve database is discussed in Module 5, “SunSolve Database Information.”
- OpenBoot™ PROM (OBP) commands are included in Module 4, “OBP Diagnostics and Commands.”

Commands and files which are otherwise not expanded in this section of Module 2 are considered to be of a level normally mastered by way of meeting the course prerequisites. These can be found in the man pages, AnswerBook documentation, and SunSolve databases.





*Sun Educational Services*

## Device Management Utilities

- The `drvconfig` utility – Creates the device directory tree in `/devices`
- The `devlinks` utility – Creates links in `/dev` according to entries in the `/devices` directory tree.
- The `tapes` utility – Creates symbolic links in the `/dev/rmt` directory.
- The `disks` utility – Creates symbolic links for disks that are attached to the system.

### *Device Management Utilities*

This section includes commands that can be used to configure new devices or create a device tree on a system. With root privilege, they can be invoked on the command line.

#### *The `drvconfig` Utility*

The `drvconfig` utility creates the device directory tree in the `/devices` directory based on the hardware configuration of the machine. The special files created include an entry for every attached, turned on hardware device, as well as pseudo drivers, such as `/dev/log`.

The `/etc/minor_perm` file is used to determine which permissions to assign to newly created special files.

---

## *Device Management Utilities*

### *The devlinks Utility*

The `devlinks` utility creates symbolic links in the `/dev` directory according to entries in the `/devices` directory tree. It is run each time a reconfiguration boot is performed. If run manually, the `drvconfig` utility must be executed first to ensure that the devices tree is intact before symbolic links which point to these devices are created.

It can be run with the debug option, `-d`, to provide information without creating the links.

### *The tapes Utility*

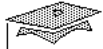
The `tapes` utility creates symbolic links in the `/dev/rmt` directory that point to the physical tape device file names in the `/devices` directory tree. Each time a reconfiguration boot is performed, the `tapes` utility is run automatically. If running the utility manually, `drvconfig` must be run first to ensure the needed entries are present in the `/devices` directory.

### *The disks Utility*

The `disks` utility creates the symbolic links in `/dev/dsk` and `/dev/rdisk` for disks that are attached to the system. It performs the following tasks:

- The kernel device tree is searched to see what devices are attached and to generate the appropriate names
- If needed, symbolic links are created for any entries found in the `/devices` directory

Each time a reconfiguration boot is performed, the `disks` utility is run automatically. If running the utility manually, the `drvconfig` utility needs to be run first to ensure the needed entries are present in `/devices` directory.



*Sun Educational Services*

## Device Management Utilities

- The `ports` utility – Creates links in `/dev/term` and `/dev/cua` for the serial port entries in `/devices`.
- The `modinfo` utility – Displays information about loaded kernel modules.

### *Device Management Utilities*

#### *The ports Utility*

The `ports` command creates symbolic links in the `/dev/term` and `/dev/cua` directories corresponding to serial port entries in the `/devices` directory and creates any needed entries in the `/etc/inittab` file for non-system ports found. It is run automatically whenever a reconfiguration boot is performed. Links to non-existent ports are removed. For ports beyond those represented by `/dev/term/a` and `/dev/term/b`, port monitor entries are created using the `sacadm` command.

## Device Management Utilities

### The modinfo Utility

The modinfo utility displays information about loaded kernel modules. With no options, it displays all loaded modules with their associated module identification number and module name. The `-i` option can be used with a module identification number to display only information about a specific module.

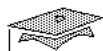
Following is an example of partial modinfo output display:

#### # modinfo

```

Id Loadaddr  Size  Info  Rev  Module Name
 6 f59ddb30 2bc8   1    1  TS (time sharing sched class)
 7 f59e06f8  4a4   -    1  TS_DPTBL (Time sharing dispatch table)
 8 f5ab9000 235ac  2    1  ufs (filesystem for ufs)
 9 f59e0ba0  dc4b 226   1  rpcmod (RPC syscall)
10 f59ee7f0 277ef  0    1  ip (IP Streams module)
13 f5a16c98  530  62   1  dma (Direct Memory Access driver)
14 f5a171c8  aeb  59   1  sbus (SBus nexus driver)
15 f5a17cb8 19a6  76   1  iommu (iommu nexus driver)
16 f5a19660 14e8  12   1  sad (Streams Administrative driver's)
17 f5a1ab48  538  2    1  pseudo (nexus driver for 'pseudo')
18 f5b1d000 f0e1  32   1  sd (SCSI Disk Driver 1.257)
19 f5a1b080 5f26  -    1  scsi (SCSI Bus Utility Routines)
20 f5a20fa8  ca68 61   1  esp (ESP SCSI HBA Driver 1.250)
25 f5a2da10 11fd6  5    1  procfs (filesystem for proc)
27 f594c000 bb40  8    1  sockfs (filesystem for sockfs)
30 f5a40058 15108  2    1  tcp (TCP Streams module)
31 f5a55160 3fb4  3    1  udp (UDP Streams module)
32 f5a59118 364d  4    1  icmp (ICMP Streams module)
33 f5a5c768 4ea3  5    1  arp (ARP Streams module)
34 f5a61610 390c  6    1  timod (transport interface str mod)
36 f5a64f20 819a  29   1  zs (Z8530 serial driver V4.116)
37 f5a6d0c0  76d  58   1  obio (obio nexus driver)
39 f5a6d830 1684  7    1  ms (streams module for mouse)
42 f5a73268  9b0  16   1  conskbd (Console kbd Multiplexer driver)

```



Sun Educational Services

## The modload Utility

- Loads a kernel module into a running system.

- Example command

```
# modload -p misc/obpsym
```

- In the `/etc/system` file:

```
forceload: misc/obpsym
```

## Device Management Utilities

### The modload Utility

The `modload` utility loads a kernel module into a running system. The *filename* argument specifies the name of the module to load. The *filename* argument can be specified as an absolute path. If the *filename* argument does not begin with a “/”, then the module loaded is found relative to the current directory, unless the `-p` option is specified.

The `-p` option tells the utility to search according to the kernel's *modpath* variable which is set in `/etc/system`. The default value for *modpath* is `/kernel /usr/kernel`. The following command example,

```
# modload -p misc/obpsym
```

loads the OBP Symbol Table module into the kernel. This module allows the maximum amount of information to be obtained when debugging Watchdog Reset conditions in OBP.

## *Device Management Utilities*

### *The modload Utility (Continued)*

An alternative way to load a module is to add the `forceload` command in the `/etc/system` file. For example,

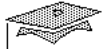
**`forceload: misc/obpsym`**

can be placed in `/etc/system` to force the load of the OBP Symbol Callback module with each boot.

---

**Note** – The directory specified in the `/etc/system` file for the `forceload` command *must* be a relative path name. A relative path name must also be used with the `modload` command.

---



*Sun Educational Services*

## The modunload Utility

- Unloads a kernel module from a running system.
- Example commands:

```
# modinfo | grep obpsym
89 f5a60d48 e1e - 1 obpsym (OBP symbol callbacks)
# modunload -i 89
```

## *Device Management Utilities*

### *The modunload Utility*

The modunload utility unloads an inactive kernel module from a running system. The `-i` option is used to specify which module to unload; for example,

```
# modinfo | grep obpsym
89 f5a60d48 e1e - 1 obpsym (OBP symbol callbacks)
# modunload -i 89
```

lists the obpsym module and then unloads it with the modunload command using the module identification number.



## The `ifconfig` Utility

```
# ifconfig le0 inet 128.50.101.121 netmask 255.255.255.0
-trailers up broadcast +

# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232 inet
127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST>mtu
1500 inet 129.151.28.5 netmask fffffff0 broadcast 129.151.28.25
ether 8:0:20:7e:b9:cd
```

## Networking Utilities

### The `ifconfig` Utility

The `ifconfig` (interface configuration) utility is used to initialize, modify, manage, and query network interfaces. It is responsible for the primary initialization of each network interface that occurs at boot time through the `/etc/rcS.d/S30rootusr` script.

The following command sets values for interface characteristics and sets the interface to the UP state. This command is similar to the initialization that occurs at boot through the `/etc/init.d/rootusr` script.

```
# ifconfig le0 inet 128.50.101.121 netmask 255.255.255.0
-trailers up broadcast +
```

One of the most useful options to the `ifconfig` command for fault analysis purposes is the `-a` option, which lists the status of all configured network interfaces. The output, depending on the system hardware, is similar to the following example:




---

## *Networking Utilities*

### *The ifconfig Utility (Continued)*

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
inet 127.0.0.1 netmask ff000000
le0:
flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST>mtu
1500 inet 129.151.28.5 netmask fffffff0 broadcast
129.151.28.25 ether 8:0:20:7e:b9:cd
```

 *Sun Educational Services*

---

## The `ifconfig` Utility

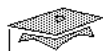
- FLAGS
- MTU
- INET
- NETMASK
- BROADCAST
- ETHER

## *Networking Utilities*

### *The `ifconfig` Utility (Continued)*

The fields displayed in the previous examples can be modified by using the `ifconfig` command. Such operations require root privilege. The fields are described here.

- **FLAGS** – The interface status. The flags `UP` and `RUNNING` are listed when an interface is functional.
- **MTU** – Maximum transmission unit, 1500 for Ethernet.
- **INET** – Internet address. This reflects the Internet address as assigned in `/etc/hosts` on the local system.
- **NETMASK** – The subnet mask value as assigned in `/etc/netmasks`.
- **BROADCAST** – The broadcast address to use when communicating to all hosts on a network. The `rusers` and `rwall` commands use a broadcast address operation.
- **ETHER** – The hardware Ethernet address for this interface.



## The netstat Command

- Returns the contents of network data structures and tables, including status of active sockets, interfaces, routing tables, and DHCP
- Examples
  - `netstat -i` – Lists statistics per interface
  - `netstat -r` – Lists routing table statistics

## Networking Utilities

### *The netstat Command*

The `netstat` command returns the contents of network data structures and tables, including status of active sockets, interfaces, routing tables, and DHCP (Dynamic Host Configuration Protocol). Some useful options are listed here.

# **netstat -i**

For each interface, this command displays the number of input and output packets, errors, collisions, and number of requests in the queue. From this, the collision rate can be calculated by dividing the number of collisions by the number of output packets and multiplying by 100. Ideally, this value should not exceed 5 to 10 percent.

# **netstat -r**

This command displays the contents of the routing table and is helpful in checking the status of routes to other networks, particularly in a subnetted environment.

## Networking Utilities

### *The netstat Command (Continued)*

Some examples of the screen output displayed with each netstat command option referenced previously are as follows:

```
# netstat -i
```

Name	Mtu	Net/Dest	Address	Ipkts	Ierrs	Opkts	Oerrs	Collis	Queue
lo0	8232	localhost	localhost	28113	0	28113	0	0	0
le0	1500	killington	killington	598631	45814	98937	27	7462	0

```
# netstat -r
```

```
Routing Table:
```

Destination	Gateway	Flags	Ref	Use	Interface
129.151.28.0	killington	U	3	445	le0
224.0.0.0	killington	U	3	0	le0
default	net28-5om	UG	0	192	
localhost	localhost	UH	0	27634	lo0



## Debugging Utilities

- The `truss` utility – Traces system calls, library calls, and signal activity for the program passed to it as an argument on the command line.

### *Debugging Utilities*

As mentioned previously, the `crash` and `adb` debugging commands are thoroughly presented in Module 7. The `truss` and `nm` utilities, very useful for certain types of tasks in fault analysis, are covered in this section.

#### *The truss Utility*

The `truss` utility, also known as `trace` on the Sun Berkeley System Distribution, traces system calls, library calls, and signal activity for the program passed to it as an argument on the command line. It is extremely helpful in determining how programs execute, and identifying points of failure in programs which return error conditions.

While knowledge of system and library calls is helpful in interpreting `truss` output, users inexperienced in this area can also benefit from the information returned by the `truss` command.

## Error-Free truss Command Output

The following example shows characteristic output from the truss utility during an error-free execution of the ls command:

```
# truss ls

execve("/usr/bin/ls", 0xEFFFFFF510, 0xEFFFFFF518)
open("/dev/zero", O_RDONLY) = 3
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF7C0000
open("/usr/openwin/lib/libc.so.1", O_RDONLY)
open("/usr/lib/libc.so.1", O_RDONLY) = 4
fstat(4, 0xEFFFFFF024) = 0
mmap(0x00000000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) =
0xEF7B0000
mmap(0x00000000, 700416, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) =
0xEF700000
munmap(0xEF793000, 61440) = 0
mmap(0xEF7A2000, 27680, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 4, 598016)
= 0xEF7A2000
mmap(0xEF7A9000, 5936, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0xE
F7A9000
close(4) = 0
open("/usr/openwin/lib/libdl.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/lib/libdl.so.1", O_RDONLY) = 4
fstat(4, 0xEFFFFFF024) = 0
mmap(0xEF7B0000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0)
= 0xEF7B0000
close(4) = 0
```

## Error-Free truss Command Output

```
open("/usr/platform/SUNW,SPARCstation-4/lib/libc_psr.so.1", O_RDONLY)
close(3) = 0
brk(0x00024720) = 0
brk(0x00026720) = 0
open("/usr/lib/locale/en_US/en_US.so.1", O_RDONLY) = 3
fstat(3, 0xEFFFDFFC) = 0
mmap(0x00000000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF6F0000
mmap(0x00000000, 81920, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF6D0000
munmap(0xEF6D4000, 61440) = 0
mmap(0xEF6E3000, 2740, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 12288) =
0xEF6E3000
close(3) = 0
open("/dev/zero", O_RDONLY) = 3
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF6C0000
munmap(0xEF6C0000, 4096) = 0
close(3) = 0
munmap(0xEF6F0000, 4096) = 0
time() = 909165315
ioctl(1, TCGETA, 0xEFFFF3A4) = 0
ioctl(1, TIOCGWINSZ, 0x000241B2) = 0
brk(0x00026720) = 0
brk(0x0002E720) = 0
lstat64(".", 0xEFFFF320) = 0
open64(".", O_RDONLY|O_NDELAY) = 3
fcntl(3, F_SETFD, 0x00000001) = 0
fstat64(3, 0xEFFFF260) = 0
```

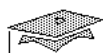
## Error-Free truss Command Output

```

getdents64(3, 0x0002DDF8, 1048)           = 840
getdents64(3, 0x0002DDF8, 1048)           = 848
getdents64(3, 0x0002DDF8, 1048)           = 488
getdents64(3, 0x0002DDF8, 1048)           = 224
getdents64(3, 0x0002DDF8, 1048)           = 200
getdents64(3, 0x0002DDF8, 1048)           = 120
getdents64(3, 0x0002DDF8, 1048)           = 40
getdents64(3, 0x0002DDF8, 1048)           = 0
close(3)                                     = 0
ioctl(1, TCGETA, 0xEFFFD434)                = 0
admin          fmfilesvisited      pcs          st370
write(1, " a d m i n          "..., 66)       = 66
core          focals                qms          tech
write(1, " c o r e          "..., 65)       = 65
write(1, " c o u d e v          "..., 70)     = 70
DeadLetters   lost+found            sa380
word_perfect.tar.Z
write(1, " D e a d L e t t e r s "..., 79)     = 79
desktop      Mail                  sa386_setup
write(1, " d e s k t o p          "..., 52)     = 52
employee     mail                  sp280
write(1, " e m p l o y e e          "..., 46)     = 46
fmdictionary ns_imap              st350
write(1, " f m d i c t i o n a r y"..., 46)     = 46
llseek(0, 0, SEEK_CUR)                   = 3463
_exit(0)

```





Sun Educational Services

## Interpreting `truss` Command Output

### General guidelines

- `exec()`
- `open()`
- `mmap()`
- `/dev/zero`

## *Interpreting `truss` Command Output*

### *General Guidelines*

There are general guidelines which are helpful for interpreting `truss` output. The first part of the output is similar for all programs, and usually includes the following:

- The `exec()` system call is invoked to open the program.
- The `open()` system call is invoked numerous times to open the standard library archive files that are needed by the program.
- The `mmap()` system call is invoked to map the program image into memory. The `munmap()` system call is subsequently also invoked.
- The `/dev/zero` file is opened to invoke the driver that zeroes out the BSS (block started by symbol) segment of the program image.



## Interpreting truss Command Output

### The `truss ls` example

- `ioctl()`
- `brk()`
- `lstat()`
- `getdents64()`
- `ioctl()` and `write()`

## Interpreting truss Command Output

### The `truss ls` Example

Other system calls executed by the `ls` code are also displayed. These include:

- The `ioctl()` system call, which is invoked to perform terminal input/output (I/O) control functions.
- The `brk()` system call, which is invoked to make a request for memory during program run.
- The `lstat()` system call, which is used to obtain file attributes; `fcntl()` performs control functions on open files, and `fstat()` obtains information about open files.
- The `getdents64()`, which reads bytes from the directory structure.
- The `ioctl()` and `write()` system calls, which are invoked for I/O control and to display information at the terminal screen.

## Example of `truss` Command Output With Errors

The following example shows the errors which `truss` displays when the `ls` program is passed a non-existent directory name as an argument:

```
# truss ls rep

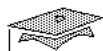
execve("/bin/ls", 0xEFFFFFF468, 0xEFFFFFF474)  argc = 2
open("/dev/zero", O_RDONLY)                  = 3
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF7C0000
open("/usr/openwin/lib/libc.so.1", O_RDONLY)
open("/usr/lib/libc.so.1", O_RDONLY)          = 4
fstat(4, 0xEFFFFFF01C)                       = 0
mmap(0x00000000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) =
0xEF7B0000
mmap(0x00000000, 700416, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) =
0xEF700000
munmap(0xEF793000, 61440)                    = 0
mmap(0xEF7A2000, 27680, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 4, 598016) = 0xEF7A2000
mmap(0xEF7A9000, 5936, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0xEF7A9000
close(4)                                     = 0
open("/usr/lib/libdl.so.1", O_RDONLY)        = 4
fstat(4, 0xEFFFFFF01C)                       = 0
mmap(0xEF7B0000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0)
= 0xEF7B0000
close(4)                                     = 0
open("/usr/platform/SUNW,SPARCstation-4/lib/libc_psr.so.1", O_RDONLY)
Err#2 ENOENT
close(3)                                     = 0
brk(0x00024720)                              = 0
brk(0x00026720)                              = 0
```

## Example of truss Command Output With Errors

```

open("/usr/lib/locale/en_US/en_US.so.1", O_RDONLY) = 3
fstat(3, 0xEFFFDFF4) = 0
mmap(0x00000000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF6F0000
mmap(0x00000000, 81920, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF6D0000
munmap(0xEF6D4000, 61440) = 0
mmap(0xEF6E3000, 2740, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 12288) = 0xEF6E3000
close(3) = 0
open("/dev/zero", O_RDONLY) = 3
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF6C0000
munmap(0xEF6C0000, 4096) = 0close(3)
= 0
munmap(0xEF6F0000, 4096) = 0
time() = 909175996
ioctl(1, TCGETA, 0xEFFFFFF39C) = 0
ioctl(1, TIOCGWINSZ, 0x000241B2) = 0
brk(0x00026720) = 0
brk(0x0002E720) = 0
lstat64("rep", 0xEFFFFFF318) Err#2 ENOENT
repwrite(2, " r e p", 3) = 3
: write(2, " : ", 2) = 2
No such file or directorywrite(2, " N o s u c h f i l e"., 25) =
25
write(2, "\n", 1) = 1
open("/dev/zero", O_RDONLY) = 3
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF6F0000
llseek(0, 0, SEEK_CUR) = 20650
_exit(2)

```



## Interpreting Errors in `truss` Command Output

### General guidelines

- Two main categories of errors:
  - System call error
  - Missing file errors
- Significant errors usually appear toward the end of the `truss` output

## *Interpreting Errors in `truss` Command Output*

### *General Guidelines*

There are two main categories of errors which `truss` reports:

- A *system call error*, often due to an invalid argument being passed to the system call. The man pages on the system calls are a helpful resource, as is the header file `/usr/include/sys/errno.h`.
- *Missing file errors*, often manifest with the `open()` system call statements. Usually, the condition is that the executing program needs to open a file which cannot be found, or for which the contents of the file are invalid or corrupt.

The most significant errors usually appear toward the lower part of the `truss` output, just before the program terminates with error. Therefore, it is often more efficient to work backward through the `truss` output.



## Interpreting Errors in `truss` Command Output

### The `truss ls` example

- The output text includes the string `Err#2 ENOENT`.
- `ENOENT` indicates that a named file or directory does not exist.
- The final invocation of the `write()` system call outputs the string `No such file or directory`.
- The `ls` program terminates with `llstat()` failing since the named file, `rep`, does not exist and cannot be found.

## *Interpreting Errors in `truss` Command Output*

### *The `truss ls` Example*

In the `ls` example, the system call activity in the example with errors is similar to the error-free example until `ls` encounters a non-existent file name. The following indications of error can be seen in the output:

- The output text includes the string `Err#2 ENOENT`.
- `ENOENT` indicates that a named file or directory does not exist.
- The final invocation of the `write()` system call outputs the string `No such file or directory`.
- The `ls` program terminates with `llseek()` failing since the named file, `rep`, does not exist and cannot be found.

## The `errno.h` Header File

An excerpt of the header file containing the main errors shown in the truss example is included here. This file can be examined on-line in the `/usr/include/sys` directory.

```
# cat /usr/include/sys/errno.h

/* Error codes */

#define EPERM 1 /* Not super-user*/
#define ENOENT 2 /* No such file or directory*/
#define ESRCH 3 /* No such process*/
#define EINTR 4 /* interrupted system call*/
#define EIO 5 /* I/O error*/
#define ENXIO 6 /* No such device or address*/
#define E2BIG 7 /* Arg list too long*/
#define ENOEXEC 8 /* Exec format error*/
#define EBADF 9 /* Bad file number*/
#define ECHILD 10 /* No children*/
#define EAGAIN 11 /* Resource temporarily unavailable*/
#define ENOMEM 12 /* Not enough core*/
#define EACCES 13 /* Permission denied*/
#define EFAULT 14 /* Bad address*/
#define ENOTBLK 15 /* Block device required*/
#define EBUSY 16 /* Mount device busy*/
#define EEXIST 17 /* File exists*/
#define EXDEV 18 /* Cross-device link*/
#define ENODEV 19 /* No such device*/
#define ENOTDIR 20 /* Not a directory*/
#define EISDIR 21 /* Is a directory*/
#define EINVAL 22 /* Invalid argument*/
#define ENFILE 23 /* File table overflow*/
#define EMFILE 24 /* Too many open files*/
#define ENOTTY 25 /* Inappropriate ioctl for device*/
#define ETXTBSY 26 /* Text file busy*/
#define EFBIG 27 /* File too large*/
#define ENOSPC 28 /* No space left on device*/
#define ESPIPE 29 /* Illegal seek*/
#define EROFS 30 /* Read only file system*/
#define EMLINK 31 /* Too many links*/
#define EPIPE 32 /* Broken pipe*/
#define EDOM 33 /* Math arg out of domain of func*/
#define ERANGE 34 /* Math result not representable*/
...
```

## Exercise: Using Solaris Troubleshooting Tools



**Exercise objective** – The objective of this exercise is to practice using some of the diagnostic tools and files discussed in this module.

### Preparation

A standard Solaris 7 installation with access to SunSolve information and the man pages is needed for this exercise. Part of this exercise is to complete one of the workshops from Appendix D. A good understanding of performing a fault analysis and diagnosis of a problem as discussed in Module 1 is also required.

### Tasks

Complete the following steps:

1. Use the `modinfo` command to list the driver modules which are loaded on your system. Is the `obpsym` module loaded?

\_\_\_\_\_

2. If the `obpsym` module is not loaded, use the `modload` command to configure it into your system.

\_\_\_\_\_

3. Use `prtconf` to determine the amount of memory which is configured on your system.

\_\_\_\_\_



---

## *Exercise: Using Solaris Troubleshooting Tools*

4. Run the `arch` and `arch -k` commands. What is the difference between the `arch` and the `arch -k` commands?

---

5. Use the `ifconfig` command to determine your Ethernet hardware address. Check the IP address next to the keyword `inet` and ensure it matches the value for your system specified in `/etc/hosts`.

---

6. Remove the symbolic links for the serial port devices in the directory `/dev/term`. Use `ls` to verify that they are gone. Use the `ports` utility to re-create the files without doing a reconfiguration boot.

## *Exercise: Using Solaris Troubleshooting Tools*

7. Enter the following command to determine which process on your system is consuming the most CPU time, and which process on your system is consuming the most memory:

```
# /usr/ucb/ps -aux
```

---

---

---

## *Exercise: Using Solaris Troubleshooting Tools*

8. Use diagnostic tools and on-line system files to answer the following questions regarding the state and configuration of your system:
  - ▼ What database does your system consult to resolve a host name?  
\_\_\_\_\_
  - ▼ How large is your swap partition?  
\_\_\_\_\_
  - ▼ Is your system currently sharing and NFS file systems?  
\_\_\_\_\_
  - ▼ Is the cron log enabled on your system?  
\_\_\_\_\_
  - ▼ Can remote systems log in as root on your system?  
\_\_\_\_\_
9. Work with your fault analysis work group to solve Workshop #11, in Appendix D. Use `truss` as an analysis tool.

## *Exercise: Using Solaris Troubleshooting Tools*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

---

## *Check Your Progress*

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Differentiate watchdog resets, panics, and system hangs
- Differentiate hardware and software problems
- Provide examples of fatal and non-fatal error conditions
- Identify a comprehensive set of Solaris commands and utilities which are useful in fault analysis
- Identify a comprehensive list of Solaris system files which contain information that is useful in fault analysis
- Describe the syntax, function, and relevance of each command or system file
- Use Solaris commands and files to determine system configuration and status information
- Solve workshop problems using Solaris utilities and system files

## *Think Beyond*

What on-line resources are at your disposal when seeking information on a system? What types of problems might you now be able to solve that you could not before?

## *Objectives*

Upon completion of this module, you should be able to

- Describe the capabilities and limitations of the POSTs in identifying and resolving system faults
- Describe different ways to view the POST
- Configure the file `/etc/remote` on a console server to enable the use of `tip` in a remote diagnostic session
- View and interpret POST output
- Describe the functionality of the `prtdiag` command

## Relevance



**Discussion** – POSTs are discussed in this module. Examples of POST screen displays are included, and with some dependencies on the classroom hardware, represent the type of display you will see in the exercise session.

Before beginning this module, consider the following questions:

- What sorts of hardware problems, if any, have you experienced with your Solaris systems?
- Have POST messages displayed to the screen during boot been helpful in troubleshooting problems on your Solaris systems?
- What other resources have been helpful to you in troubleshooting hardware problems?

## References



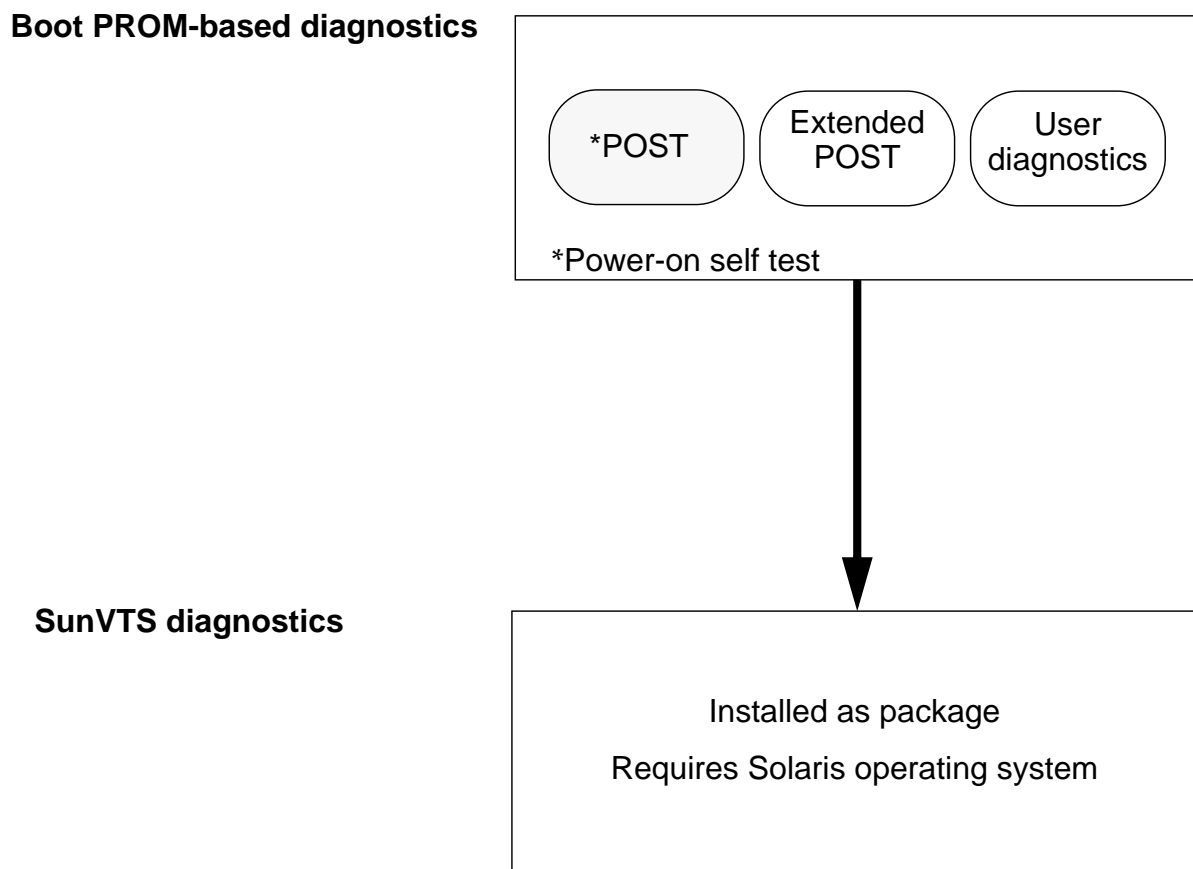
**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- *Field Engineer Handbook*, part numbers 800-4006-16 and 800-4247
- *OpenBoot Command Reference*, part number 800-6076
- *OpenBoot 2.x Quick Reference Card*, part number 802-1958
- *OpenBoot 3.x Quick Reference Card*, part number 802-3240
- <http://docs.sun.com>, the AnswerBook web site



## Diagnostics Overview

This section describes the importance, capabilities, and limitations of the POST in identifying and resolving system faults.



**Figure 3-1** Diagnostics Overview



## Boot PROM POST

- Is invoked automatically at the power-on sequence
- Differs slightly between workstation models
- Differs slightly between boot PROM revisions
- Conducts error detection and hardware verification for each system board
- Conducts all hardware bus probes, and saves information for the operating system's automatic reconfiguration

## *Boot PROM POST*

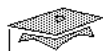
### *POSTs*

- Are invoked automatically at the power-on sequence
- Differ slightly between workstation models
- Differ slightly between boot PROM revisions
- Conduct error detection and hardware verification for each system board
- Conduct all hardware bus probes, and save information for the operating system's automatic reconfiguration (`boot -r`) and memory sizing

---

**Note** – A deliberate limitation of the boot PROM POST is that the I/O devices themselves are not tested, only those devices and buses required to access the boot device are tested.

---



*Sun Educational Services*

## SunVTS Diagnostics

- Requires Solaris operating environment
- Is installed as a package
- Is used for system verification
- Runs in a window or non-windowed environment

### *Boot PROM POST*

#### *SunVTS Diagnostics*

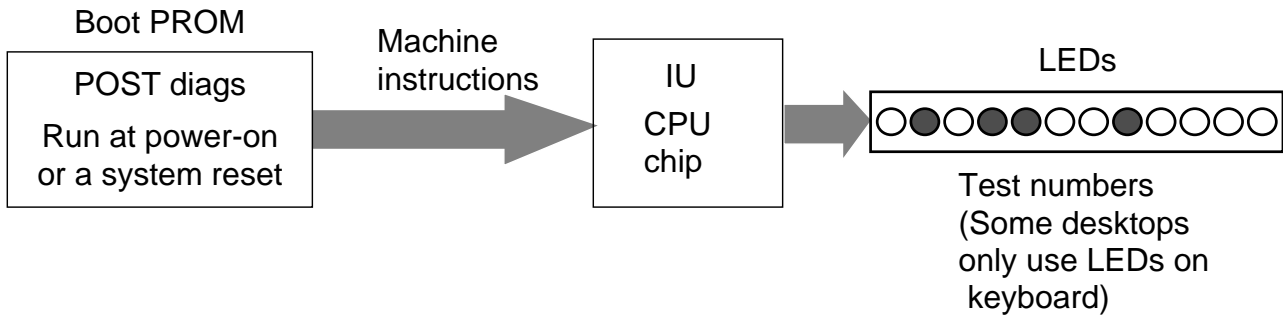
##### SunVTS

- Requires the Solaris operating environment to be operating
- Is installed as a package
- Is used for system verification
- Runs in a window or non-windowed environment

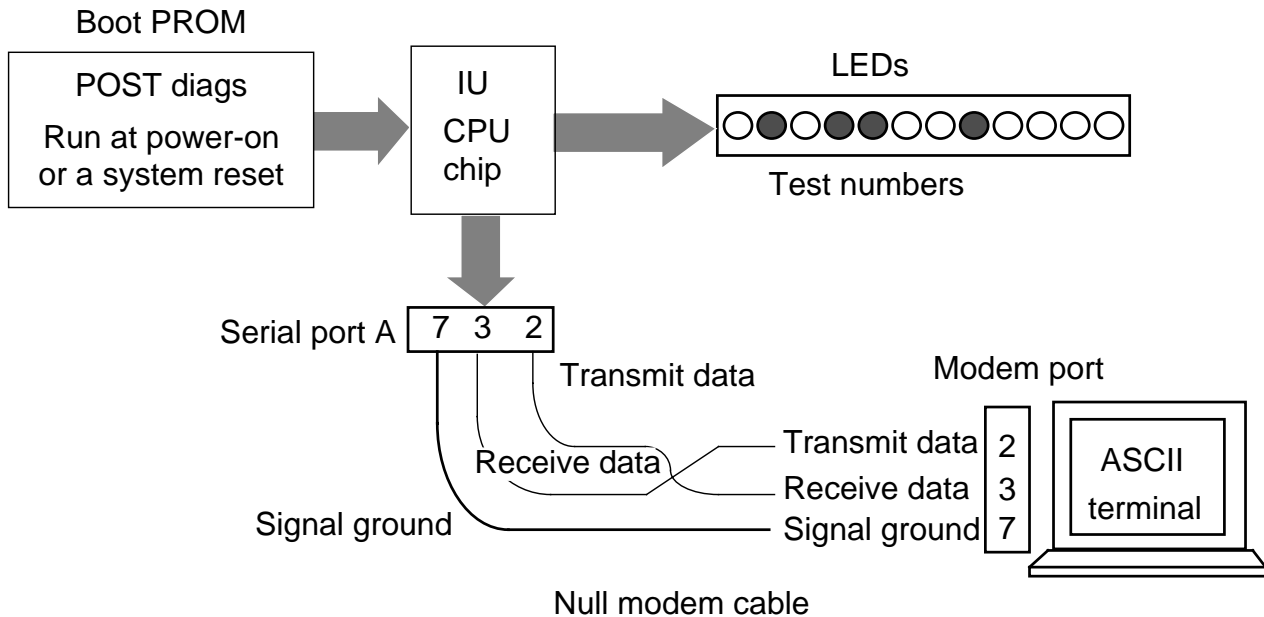
## POST Viewing Methods

POST is a set of boot PROM resident firmware programs that run independently of the Solaris operating environment.

Figure 3-2 and Figure 3-3 illustrate different ways to view POST.



**Figure 3-2** Viewing POST From the CPU Board LEDs



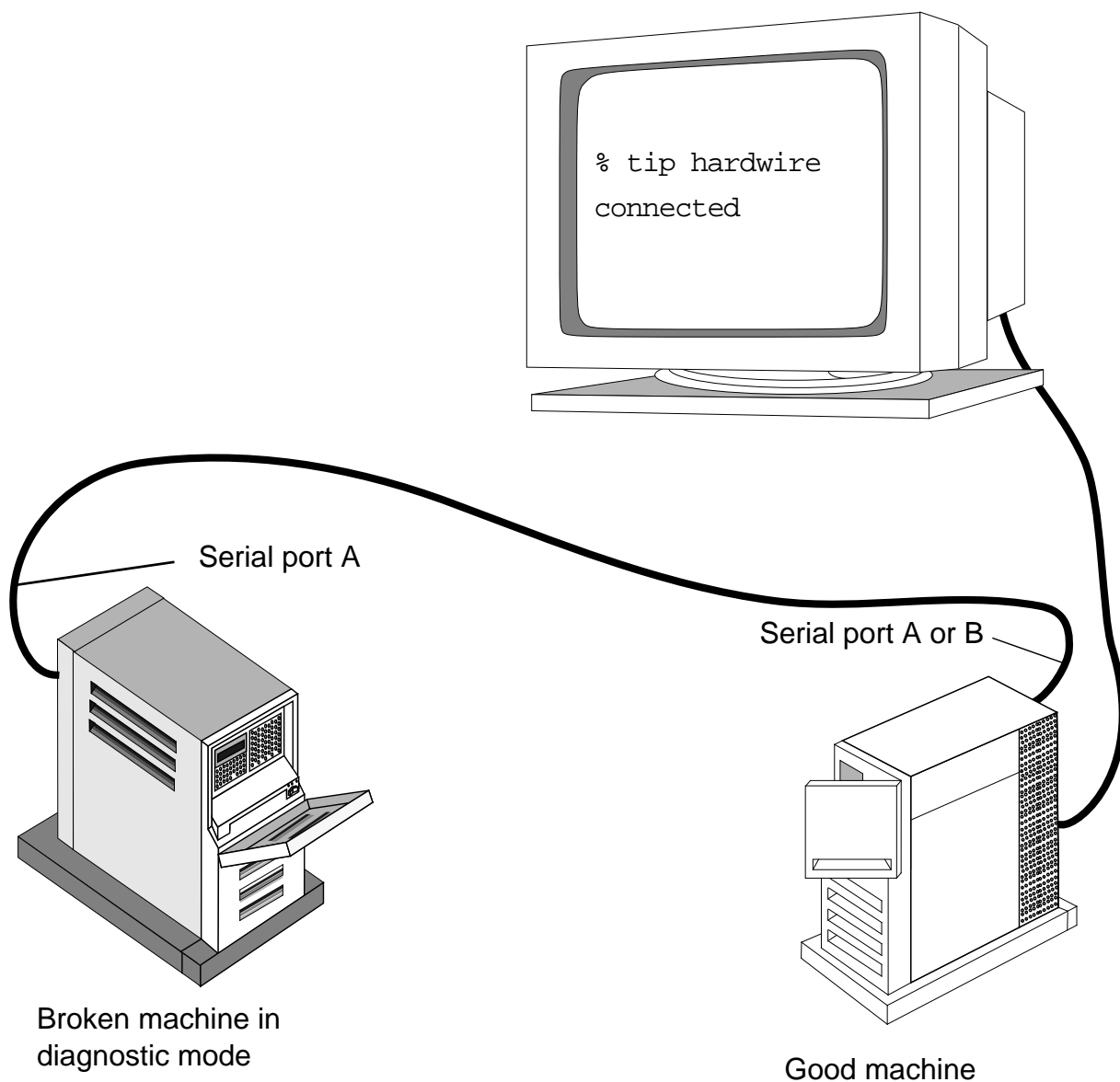
The ASCII terminal will list, in English text, the current executing POST diagnostic.

**Figure 3-3** View POST With a Serial Port Terminal

## Viewing POST Using the `tip` hardware Command

You can use the `tip` hardware command

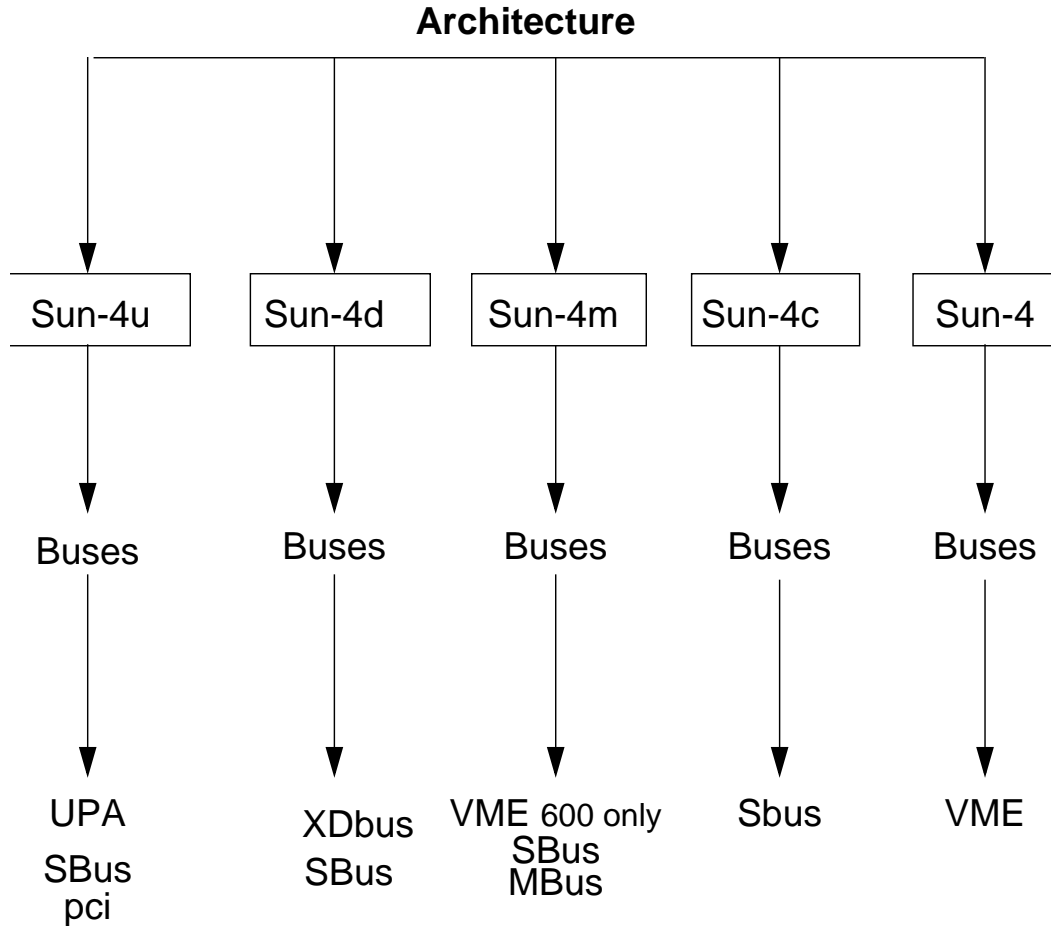
- When a serial port terminal is not available
- To analyze POST output in a Sun window



**Figure 3-4** Viewing POST Using the `tip` hardware Command

## Primary Buses

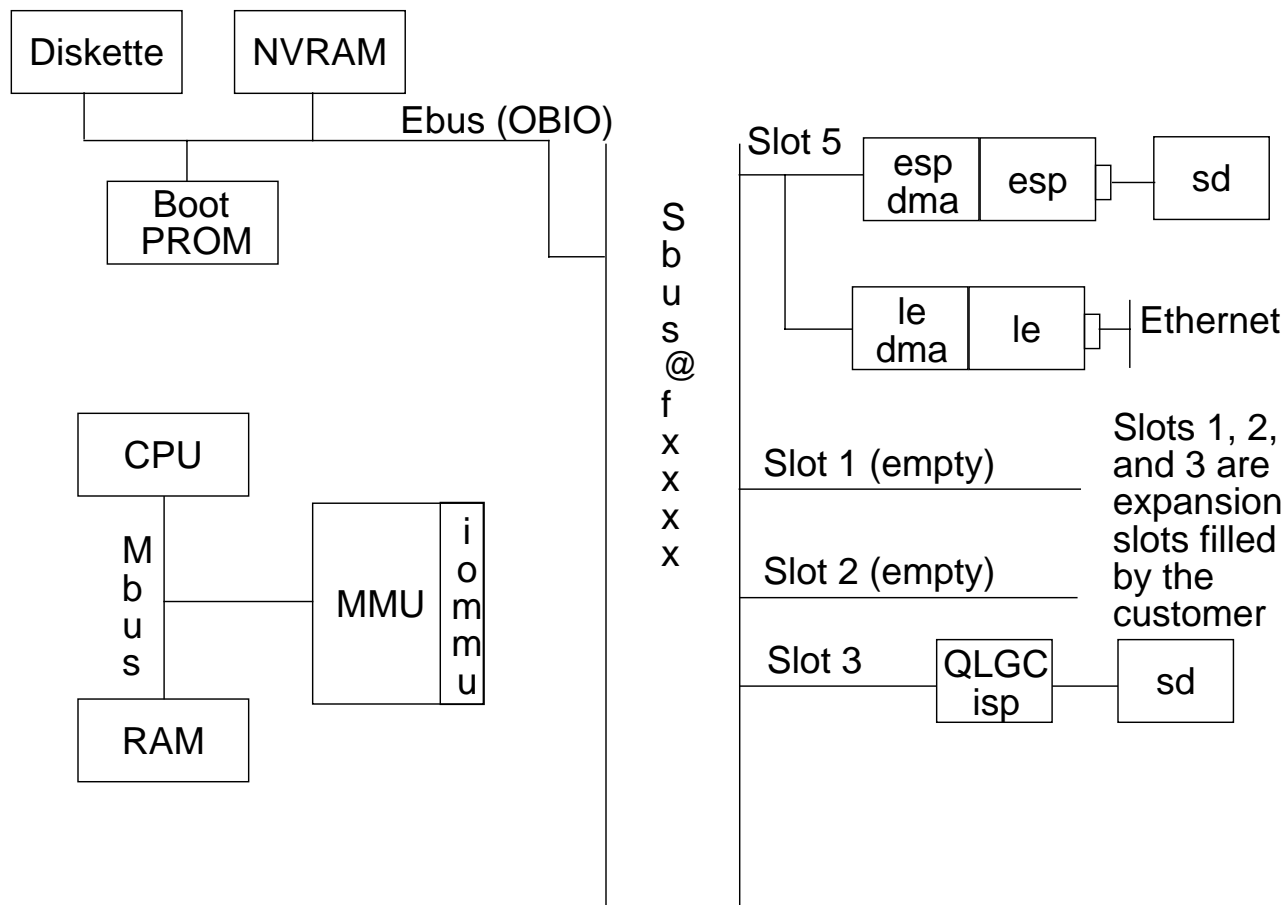
The following is a graphical representation of the major buses supported by SPARC architectures. Not shown is the onboard I/O (OBIO) for all architectures, which connects to chips on the system board as serial controllers do.



**Table 3-1** Sun Architecture

Architecture	Model
Sun-4	4/330, 4/370, 4/390, 4/470, 4/490
Sun-4c	SS1, SS1+, SS2, SLC, ELC, IPC, IPX
Sun-4m	SS5, SS10, SS20, 630, 670, 690, Classic, ClassicX, SSLX
Sun-4d	SC2000, SS1000
Sun-4u	140 ultra1, 170 creator, 170e creator3, ultra2, ultra5, ultra10, ultra 250 & 450, ultra 30 & 60, Enterprise 3000-6500

# Sun 4m Architecture



## *SPARCstation 5 POST Example With tip*

The following example represents a session performed on an SPARCstation™ 5. Output can vary according to the hardware configuration and problems detected.

```
# tip hardware
connected
Power-ON Reset
MB86907 POST 2.2.3 03SEP96
Probing system memory: 32 32 0 0 0 0 0 0
Config = 8800000A
512Kb ecache detected
1.1.1 mem ram walking ones Pass
1.1.2 mem ram address Pass
1.1.3 mem ram post r/w region Pass
1.1.4 mem ram obp r/w region Pass
1.2.1 mem control parity Pass
2.1.1 srmmu regs read/write Pass
2.2.1 srmmu ram io-tlb Pass
2.2.2 srmmu ram d-tlb Pass
2.2.3 srmmu ram pdt cache Pass
3.1.1 iommu reg read/write Pass
3.1.2 iommu reg flush individual Pass
3.1.3 iommu reg flush all Pass
3.2.1 iommu timeout ebus Pass
3.2.2 iommu timeout sbus Pass
4.1.1 fpu reg regfile Pass
4.1.2 fpu reg misalign Pass
4.1.3 fpu reg single precision Pass
4.1.4 fpu reg double precision Pass
4.2.2 fpu exceptions double precision Pass
5.1.1 cache dcache ram Pass
5.1.2 cache dcache address Pass
5.1.3 cache dcache tag Pass
5.1.4 cache dcache clear Pass
5.2.1 cache icache ram Pass
5.2.2 cache icache address Pass
5.2.3 cache icache tag Pass
5.2.4 cache icache clear Pass
5.3.1 cache ecache ram Pass
```



*SPARCstation 5 POST Example With tip*

```

5.3.2 cache ecache address Pass
5.3.3 cache ecache tag Pass
5.4.1 cache snoop ram Pass
6.1.1 i/o counters processor user timer Pass
6.1.2 i/o counters processor counter Pass
6.1.3 i/o counters system counter Pass
6.2.1 i/o lance getid Pass
6.2.2 i/o lance csr Pass
6.2.3 i/o lance rap Pass
6.2.4 i/o lance rdp Pass
6.3.1 i/o esp register r/w Pass
6.3.2 i/o esp config reg Pass
6.3.3 i/o esp fifo access Pass
6.3.4 i/o esp command reg Pass
6.4.1 i/o pp register access Pass
6.4.2 i/o pp io readback Pass
6.4.3 i/o pp tcr readback Pass
6.5.1 i/o tod regs test Pass
6.5.2 i/o tod nvram access Pass
7.1.1 intr regs sys Pass
7.1.2 intr regs proc Pass
7.2.1 intr software interrupts disabled Pass
7.2.2 intr software interrupts enabled Pass
7.2.3 intr software multi Pass
7.3.1 intr pp Interrupts Pass
7.4.1 intr timer system counter Pass
7.4.2 intr timer process counter Pass
8.1.1 dma apc bypass Pass

```

initializing TLB

initializing cache

Allocating SRMMU Context Table

Setting SRMMU Context Register

Setting SRMMU Context Table Pointer Register

Allocating SRMMU Level 1 Table

Mapping RAM

Mapping ROM

ttya initialized

Probing Memory Bank #0 32 Megabytes

Probing Memory Bank #1 32 Megabytes

Probing Memory Bank #2 32 Megabytes

Probing Memory Bank #3 32 Megabytes

Probing Memory Bank #4 Nothing there

## *SPARCstation 5 POST Example With tip*

```
Probing Memory Bank #5 Nothing there
Probing Memory Bank #6 Nothing there
Probing Memory Bank #7 Nothing there
Probing CPU FMI,MB86907
Probing /iommu@0,10000000/sbus@0,10001000 at 5,0  espdma esp sd st
SUNW,bpp ledma le
Probing /iommu@0,10000000/sbus@0,10001000 at 4,0  SUNW,CS4231 power-
management
Probing /iommu@0,10000000/sbus@0,10001000 at 1,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 2,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 3,0  cgsix
Probing /iommu@0,10000000/sbus@0,10001000 at 0,0  Nothing there
Probing Memory Bank #0 32 Megabytes
Probing Memory Bank #1 32 Megabytes
Probing Memory Bank #2 32 Megabytes
Probing Memory Bank #3 32 Megabytes
Probing Memory Bank #4 Nothing there
Probing Memory Bank #5 Nothing there
Probing Memory Bank #6 Nothing there
Probing Memory Bank #7 Nothing there
Probing CPU FMI,MB86907
Probing /iommu@0,10000000/sbus@0,10001000 at 5,0  espdma esp sd st
SUNW,bpp ledma le
Probing /iommu@0,10000000/sbus@0,10001000 at 4,0  SUNW,CS4231 power-
management
Probing /iommu@0,10000000/sbus@0,10001000 at 1,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 2,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 3,0  cgsix
Probing /iommu@0,10000000/sbus@0,10001000 at 0,0  Nothing there
```

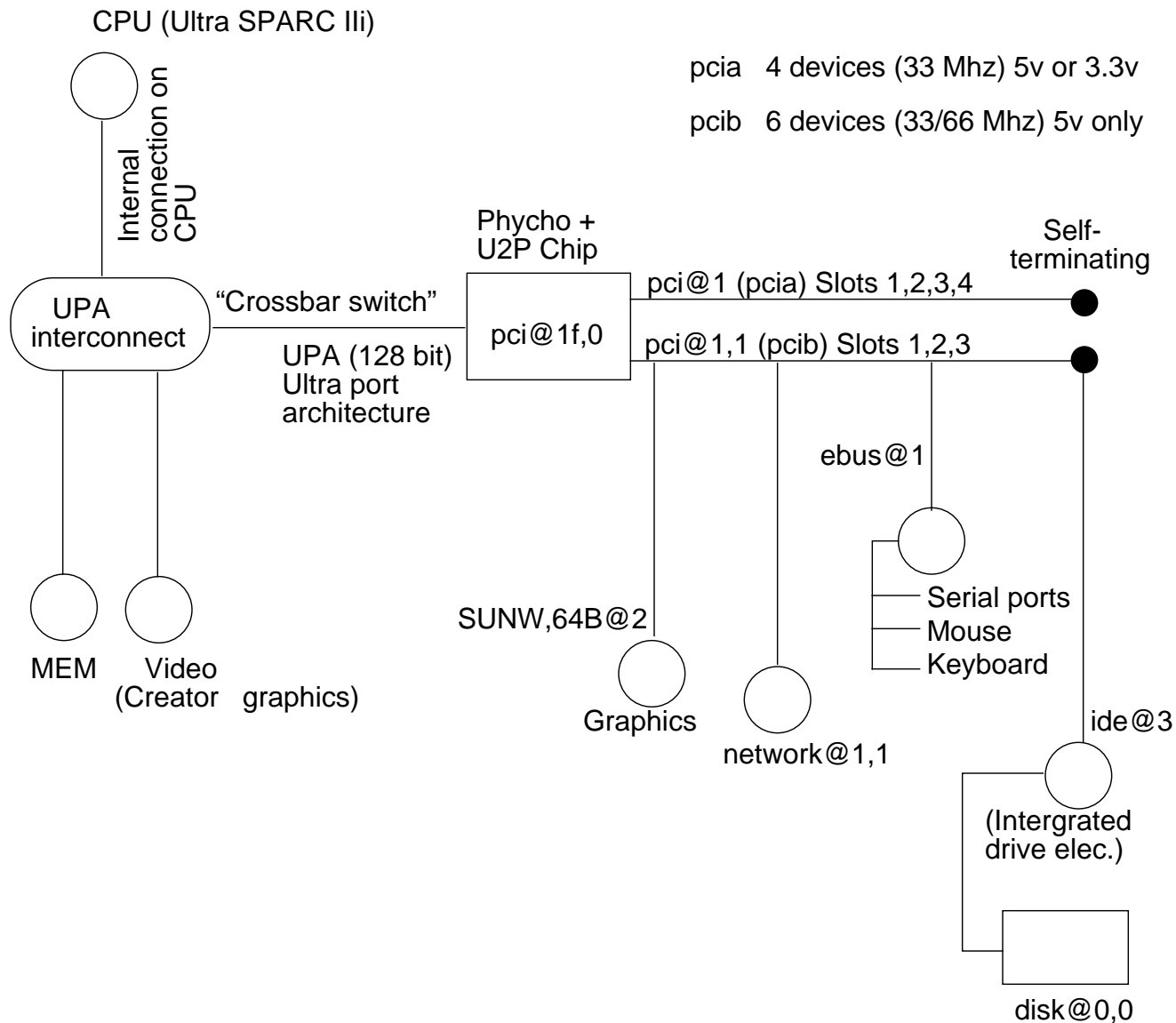
SPARCstation 5, No Keyboard

ROM Rev. 2.29, 128 MB memory installed, Serial #9453315.

Ethernet address 8:0:20:90:3f:3, Host ID: 80903f03.

# Ultra 5 and Ultra 10 Architecture

Figure 3-5 illustrates the architecture of Ultra 5 and Ultra 10 machines.

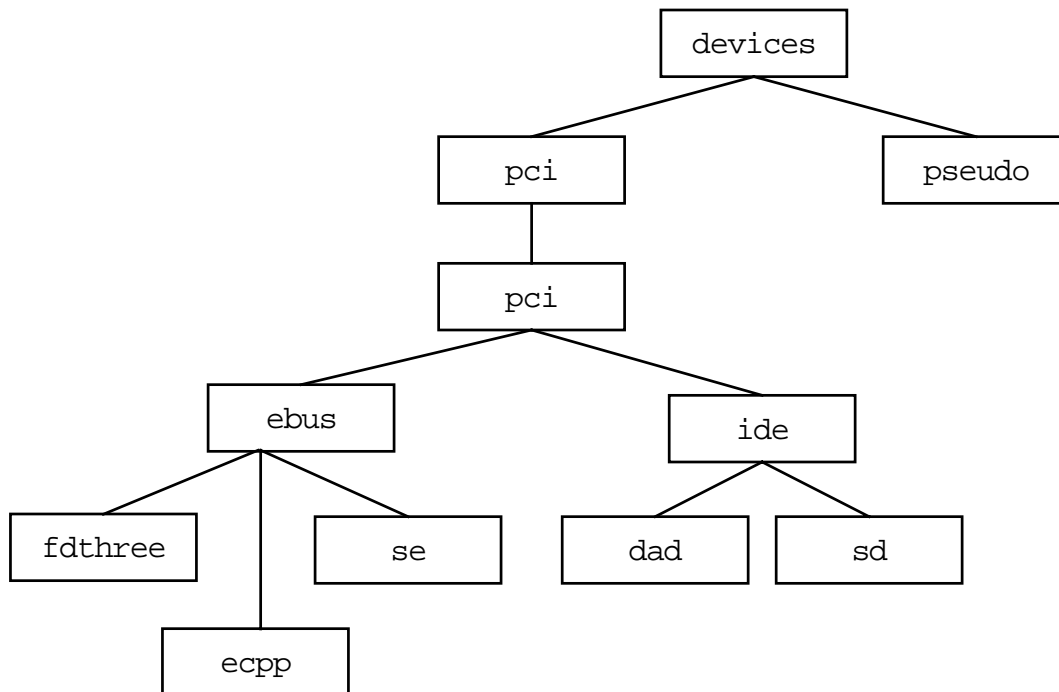


**Figure 3-5** Ultra 5 and Ultra 10 Architecture

## UltraSPARC pci-based Device Tree Hierarchy

The following discussion focuses on the architecture and board layout for the UltraSPARC™ pci-based systems which include Ultra5, Ultra10, Ultra30, Ultra60, Ultra250, and Ultra450 systems.

Figure 3-6 is an example of a device tree structure that shows how it is similar to a file system.



**Figure 3-6** Tree-like Hierarchy of Device Nodes

The OBP uses device nodes to represent devices. These device nodes are organized into a tree-like hierarchy. Controller boards represent the branches of the tree and the devices nodes are the leaves of the tree. An administrator can access a device by specifying the path used to reach the device.

## *UltraSPARC pci-based Example With tip*

```
# tip hardware
connected
Hardware Power ON
@(#) Sun Ultra 5/10 UPA/PCI 3.11 Version 12 created 1998/05/19 11:30
Probing keyboard Done
%o0 = 0000.0000.0000.4001
Executing Power On SelfTest
@(#) Sun Ultra 5/10 (Darwin) POST 2.2.9 (Build No. 502) 17:54 on 02/19/98
CPU: UltraSPARC-LC (MHz: 271 Ecache Size: 256KB)
Init POST BSS
  Init System BSS
  NVRAM
  NVRAM Battery Detect Test
  NVRAM Scratch Addr Test
  NVRAM Scratch Data Test
DMMU TLB Tags
  DMMU TLB Tag Access Test
DMMU TLB RAM
  DMMU TLB RAM Access Test
Probe Ecache
  Probe Ecache
Ecache Tests
  Ecache RAM Addr Test
  Ecache Tag Addr Test
  Ecache RAM Test
  Ecache Tag Test
All CPU Basic Tests
  V9 Instruction Test
  CPU Tick and Tick Compare Reg Test
  CPU Soft Trap Test
  CPU Softint Reg and Int Test
All Basic MMU Tests
```

## *UltraSPARC pci-based Example With tip*

```
DMMU Primary Context Reg Test
DMMU Secondary Context Reg TestDMMU TSB Reg Test
DMMU Tag Access Reg Test
DMMU VA Watchpoint Reg Test
DMMU PA Watchpoint Reg Test
IMMU TSB Reg Test
IMMU Tag Access Reg Test
All Basic Cache Tests
  Dcache RAM Test
  Dcache Tag Test
  Icache RAM Test
  Icache Tag Test
  Icache Next Test
  Icache Predecode Test
Sabre MCU Control & Status Regs Init and Tests
  Init Sabre MCU Control & Status Regs
    Initializing SC registers in SabreIO
Memory Probe and Init
  Probe Memory
    bank 0:  OMB
    INFO:    128MB Bank 2
INFO: MC0 = 0x00000000.80001484, MC1 = 0x00000000.06459acb
  Ecache Access Test
  Malloc Post Memory
  Memory Addr with Ecache
  Load Post In Memory
  Run POST from MEM
loaded POST in memory
  Map PROM/STACK/NVRAM in DMMU
  Update Master Stack/Frame Pointers
All FPU Basic Tests
  FPU Regs Test
  FPU Move Regs Test
  FPU State Reg Test
```

## *UltraSPARC pci-based Example With tip*

```
FPU Functional Test
FPU Trap Test
UPA Data Bus Line Test
Memory Tests
  Init Memory
    Memory Addr with Ecache Test
    ECC Memory Addr Test
    Block Memory Addr Test
    Block Memory Test
    Write 0x33333333.33333333 .....
Read .....
  ECC Blk Memory Test
Write 0xa5a5a5a5.a5a5a5a5 .....
Read .....
  All Basic Sabre MMU Tests
  Init Sabre
  PIO Decoder and BCT Test
  PCI Byte Enable Test
  Interrupt Map (short) Reg Test
  Interrupt Set/Clr Reg Test
  Sabre IOMMU Regs Test
  Sabre IOMMU RAM Address Test
  Sabre IOMMU CAM Address Test
  IOMMU TLB Compare Test
  IOMMU TLB Flush Test
  PBMA PCI Config Space Regs Test
  PBMA Control/Status Reg Test
  PBMA Diag Reg Test
  Sabre IO Regs Test
All Advanced CPU Tests
  DMMU Hit/Miss Test
  IMMU Hit/Miss Test
  DMMU Little Endian Test
  IU ASI Access Test
  FPU ASI Access Test
```

## *UltraSPARC pci-based Example With tip*

```
Ecache Thrash Test
All CPU Error Reporting Tests
CPU Data Access Trap Test
  CPU Addr Align Trap Test
  DMMU Access Priv Page Test
  DMMU Write Protected Page Test
All Advanced Sabre IOMMU Tests
  Init Sabre
  Consist DMA Rd, IOMMU miss Ebus Test
  Consist DMA Rd, IOMMU hit Ebus Test
  Consist DMA Wr, IOMMU miss Ebus Test
  Consist DMA Wr, IOMMU hit Ebus Test
  Pass-Thru DMA Rd, Ebus device Test
  Pass-Thru DMA Wr, Ebus device Test
  Consist DMA Rd, IOMMU LRU Lock Ebus Test
  Consist DMA Wr, IOMMU LRU Locked Ebus Test
All Basic Cheerio Tests
  Cheerio Ebus PCI Config Space Test
  Cheerio Ethernet PCI Config Space Test
  Cheerio Init
All Sabre IOMMU Error Reporting Tests
  Init Sabre
  PIO Read, Master Abort Test
  PIO Read, Target Abort Test
Status of this POST run:PASS
manufacturing mode=OFF
Time Stamp [hour:min:sec] 15:11:41 [month/date year] 12/06 1998
Power On Selftest Completed
Power ON
@(#) Sun Ultra 5/10 UPA/PCI 3.11 Version 12 created 1998/05/19 11:30
Clearing E$ Tags Done
Clearing I/D TLBs Done
Probing Memory Done
MEM BASE = 0000.0000.1000.0000
MEM SIZE = 0000.0000.0800.0000
```



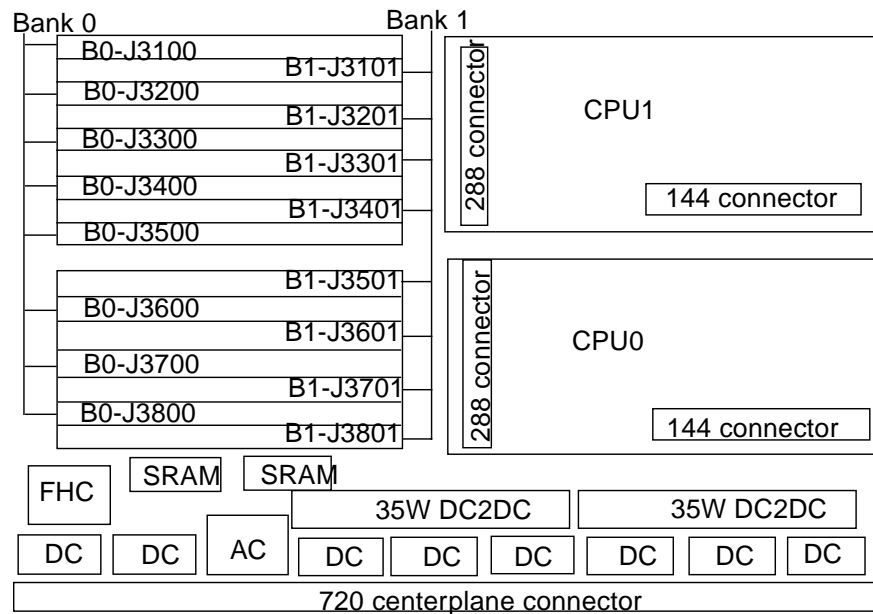
## *UltraSPARC pci-based Example With tip*

```
11-Column Mode Enabled
MMUs ON
Copy Done
PC = 0000.01ff.f000.200c
PC = 0000.0000.0000.2050
Decompressing into Memory Done
Size = 0000.0000.0006.e3e0
ttya initialized
Reset Control: BXIR:0 BPOR:0 SXIR:0 SPOR:1 POR:0
UltraSPARC-III 2-2-2 module
Probing Memory Bank #0  0 +  0 :  0 Megabytes
Probing Memory Bank #2 64 + 64 : 128 Megabytes
Probing UPA Slot at 1e,0 Nothing There
Probing /pci@1f,0/pci@1,1 at Device 1 pci108e,1000 network
Probing /pci@1f,0/pci@1,1 at Device 2 SUNW,m64B
Probing /pci@1f,0/pci@1,1 at Device 3 ide disk cdrom
Probing /pci@1f,0/pci@1 at Device 1 scsi disk tape scsi disk tape
Probing /pci@1f,0/pci@1 at Device 2 Nothing there
Probing /pci@1f,0/pci@1 at Device 3 Nothing there
Probing /pci@1f,0/pci@1 at Device 4 Nothing there
Sun Ultra 5/10 UPA/PCI (UltraSPARC-III 270MHz), No Keyboard
OpenBoot 3.11, 128 MB memory installed, Serial #10416478.
Ethernet address 8:0:20:9e:f1:5e, Host ID: 809ef15e.
```

## UltraSPARC Sbus-based Architecture

The following discussion focuses on the architecture and board layout for the UltraSPARC Sbus-based systems which include Ultra1, Ultra2, and Enterprise 3000 through 6500 systems.

### CPU/Memory Board Layout

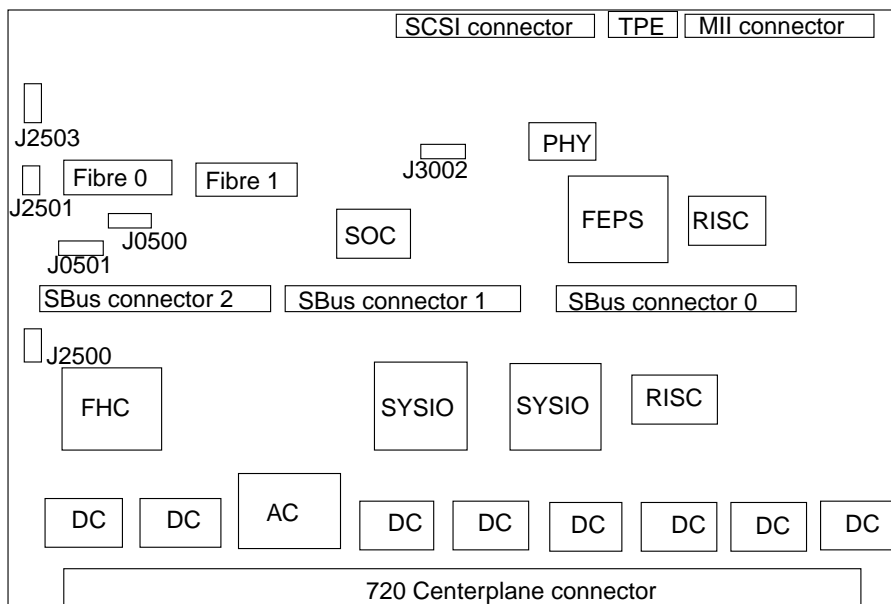


**Figure 3-7** CPU/Memory Board Layout for Sbus Architecture

The CPU/Memory board has 16 SIMM sockets, which are divided into two banks of 8 SIMMs each, Bank 0 and Bank 1. Bank 0 and Bank 1 SIMMs occupy alternate slot locations; Bank 0 SIMMs are in the even-numbered slots, and Bank 1 SIMMs are in odd-numbered slots.

# UltraSPARC Sbus-based Architecture

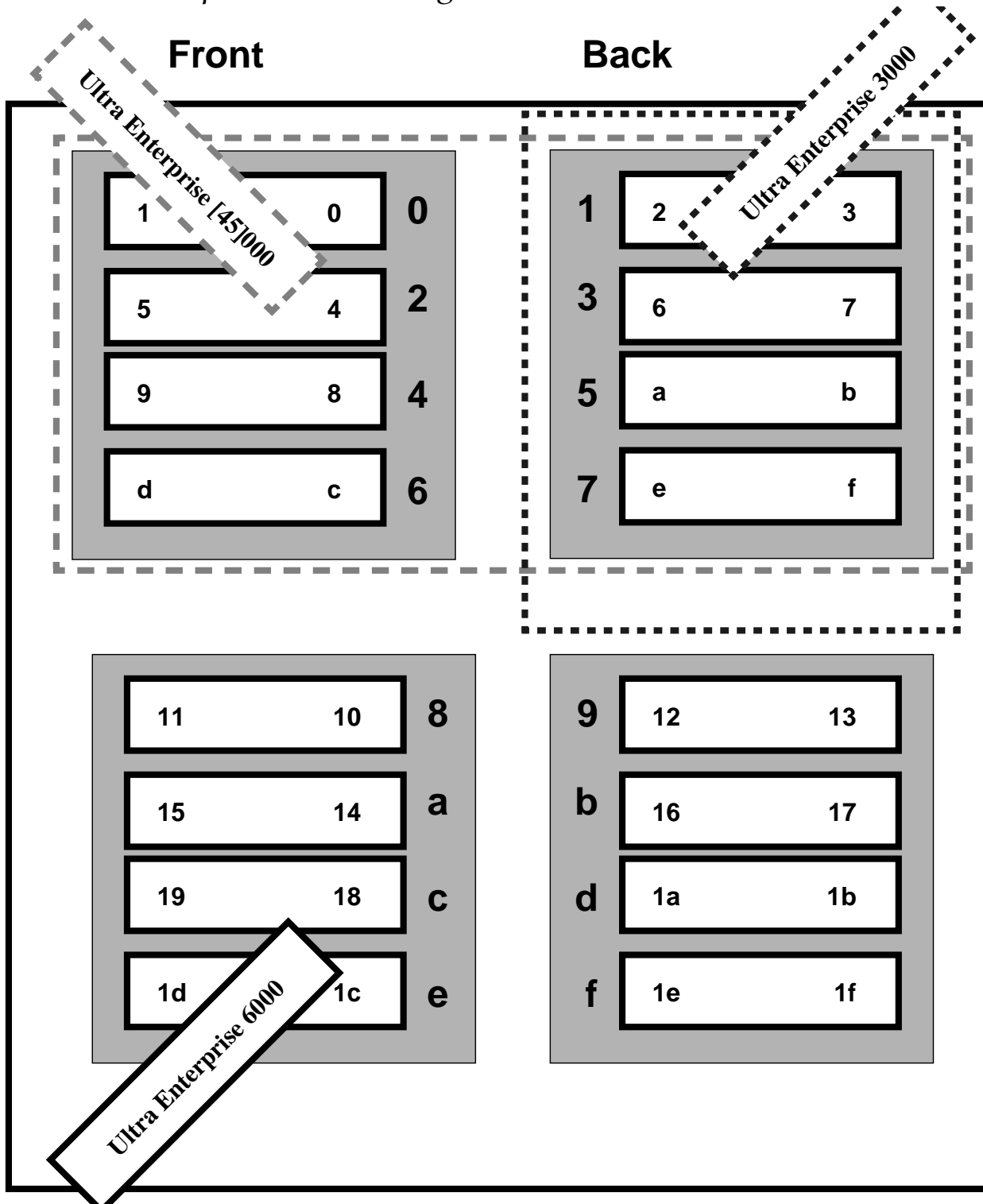
## I/O SBus Board Component Locations



**Figure 3-8** I/O Sbus board components

# UltraSPARC Sbus-based Architecture

## Centerplane Numbering Scheme



---

## UltraSPARC Sbus-based Architecture

### Reading Device Path

An Enterprise X000 displays the following device path when boot -v is used:

```
ssd20 is /sbus@3,0/SUNW,soc@d,10000/SUNW,pln@a0000000,78c6ff/ssd@1,4
```

This device path produces a wealth of information once decoded. For example:

- **ssd20** – Is the instance value assigned to a SPARC Storage Array
- **sbus@3** – Requires simple math and understanding of the two types of I/O cards:
  - ▼ I/O SBus card – Type 1
    - SYSIO (a) drives SOC (d) SBus (2) SBus (1)
    - SYSIO (b) drives SBus (0) SBus (3)
  - ▼ I/O Graphic card – Type 2
    - SYSIO (b) drives SOC (d) SBus (2) SBus (0) SBus (3)

Assume an I/O Sbus card is inserted in slot 3. During boot or POST, this card would report the following:

```
sbus@6 and sbus@7
```

Dividing 6 by 2 equals 3 with 0 remainder. The 3 represents slot location and remainder 0 indicates SYSIO (a).

Dividing 7 by 2 equals 3 with 1 remainder. The 3 represents slot location and remainder 1 indicates SYSIO (b).

---

**Note** – A Type 1, Sbus I/O, card will always report two Sbuses.

---

## *UltraSPARC Sbus-based Architecture*

### *Reading Device Path (Continued)*

Dividing 3 by 2 equals 1 with 1 remainder. The card is in Slot 1, but the type of I/O cannot be determined just yet. The soc on board chip is connected to it, which indicates a Type 2 I/O, graphic card. Remember, the 1 remainder indicates SYSI/O (b) ASIC chip, and soc can only be connected to SYSIO(b) on a graphic I/O card.

soc@d

The Serial Optic Card is connected to Sbus slot d in Enterprise cabinet slot 1.

```
/sbus@3,0/SUNW,soc@d,10000/SUNW,pln@a0000000,78c6ff/
```

Connected to the soc is a SPARC Storage Array with the World Wide number **a0000000,78c6ff**.

```
/sbus@3,0/SUNW,soc@d,10000/SUNW,pln@a0000000,78c6ff/ssd@1,4
```

The physical address of the disk is target 1 drive 4.

# UltraSPARC Sbus-based Architecture

## POST Sample Output

Hardware Power ON

@(#) Ultra Enterprise 3.1 Version 1  
Probing keyboard Done

Reset = 0000.01ff.f090.88c4  
%o0 = 0000.0000.0000.2001  
%o1 = 0000.0000.0000.2020  
%o2 = 0000.01ff.f090.88b2  
%o3 = 0000.01ff.f090.88c4

Calling POST

3,0>

3,0>@(#) POST 2.5.1 2/12/1996 05:24 PM

3,0>

SelfTest Initializing (Diag Level 10, ENV 00002001) IMPL 0010 MASK 22

3,1>

3,0>Board 3 CPU FEPROM Test

3,1>@(#) POST 2.5.1 2/12/1996 05:24 PM

3,0>Board 3 Basic CPU Test

3,1>

SelfTest Initializing (Diag Level 10, ENV 00000000) IMPL 0010 MASK 22

3,0> Set CPU UPA Config and Init SDB Data

3,1>Board 3 CPU FEPROM Test

3,1>Board 3 Basic CPU Test

3,0>Board 3 MMU Enable Test

3,1> Set CPU UPA Config and Init SDB Data

3,0> DMMU Init

3,1>Board 3 MMU Enable Test

3,1> DMMU Init

3,0> IMMU Init

3,1> IMMU Init

3,0> Mapping Selftest Enabling MMUs

3,1> Mapping Selftest Enabling MMUs

3,0>Board 3 Ecache Test

3,0> Ecache Probe

3,1>Board 3 Ecache Test

3,1> Ecache Probe

3,0> Ecache Tags

3,1> Ecache Tags

## *UltraSPARC Sbus-based Architecture*

### *POST Sample Output (Continued)*

```
3,0>   Ecache Quick Verify
3,1>   Ecache Quick Verify
3,0>   Ecache Init
3,1>   Ecache Init
3,0>   Ecache RAM
3,1>   Ecache RAM
3,0>   Ecache Address Line
3,1>   Ecache Address Line
3,0>Board 3 FPU Functional Test
3,0>   FPU Enable
3,1>Board 3 FPU Functional Test
3,1>   FPU Enable
3,0>Board 3 Board Master Select Test
3,0>   Selecting a Board Master
3,1>Board 3 Board Master Select Test
3,1>   Selecting a Board Master
3,0>Board 3 FireHose Devices Test
3,0>Board 3 Address Controller Test
3,0>   AC Initialization
3,0>   AC DTAG Init
3,0>Board 3 Dual Tags Test
3,0>   AC DTAG Init
3,0>Board 3 FireHose Controller Test
3,0>   FHC Initialization
3,0>Board 3 JTAG Test
3,0>   Verify System Board Scan Ring
3,0>Board 3 Centerplane Test
3,0>   Centerplane Join
3,0>Board 3 Setup Cache Size Test
3,0>   Setting Up Cache Size
3,0>Board 3 System Master Select Test
3,0>   Setting System Master
3,0>POST Master Selected (XIR,JTAG,CENTRAL)
3,0>Board 16 Clock Board Serial Ports Test
3,0>Board 16 NVRAM Devices Test
3,0>   M48T59 (TOD) Init
3,0>Board 3 System Board Probe Test
3,0>   Probing all CPU/Memory BDA3,0>   Probing System Boards
3,0>   Probing CPU Module JTAG Rings
```



## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```
3,0>Setting System Clock Frequency
3,0>   CPU mid 6 (167Mhz) Checked in OK
3,0>   CPU mid 7 (167Mhz) Checked in OK
3,0>   CPU mid 10 (167Mhz) Checked in OK
3,0>   CPU mid 11 (167Mhz) Checked in OK
3,0>   CPU mid 14 (167Mhz) Checked in OK
3,0>   CPU mid 15 (167Mhz) Checked in OK
3,0>   CPU mid 10 (167Mhz) Checked in OK
3,0>   CPU mid 11 (167Mhz) Checked in OK
3,0>   CPU mid 14 (167Mhz) Checked in OK
3,0>   CPU mid 15 (167Mhz) Checked in OK
3,0>   Setting CPU frequency from 125Mhz (f=0x000) to 167Mhz
3,0> ***** Clock Reset - retesting
3,0>
3,0>@(#) POST 2.5.1 2/12/1996 05:24 PM
3,0>
   SelfTest Initializing (Diag Level 20, ENV 00002081) IMPL 0010 MASK 22
3,1>
3,0>Board 3 CPU FPRM Test
3,1>@(#) POST 2.5.1 2/12/1996 05:24 PM
3,0>Board 3 Basic CPU Test
3,1>
   SelfTest Initializing (Diag Level 20, ENV 00002081) IMPL 0010 MASK 22
3,0>   FPU Registers and Data Path Test
3,1>Board 3 CPU FPRM Test
3,1>Board 3 Basic CPU Test
3,1>   FPU Registers and Data Path Test
3,0>   Instruction Cache Tag RAM Test
3,1>   Instruction Cache Tag RAM Test
3,0>   Instruction Cache Instruction RAM Test
3,1>   Instruction Cache Instruction RAM Test
3,0>   Instruction Cache Next Field RAM Test
3,1>   Instruction Cache Next Field RAM Test
3,0>   Instruction Cache Pre-decode RAM Test
3,1>   Instruction Cache Pre-decode RAM Test
3,0>   Data Cache RAM Test3,1>   Data Cache RAM Test
3,0>   Data Cache Tags Test
3,1>   Data Cache Tags Test
3,0>   DMMU Registers Access Test
```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```
3,1> DMMU Registers Access Test
3,0> DMMU TLB DATA RAM Access Test
3,1> DMMU TLB DATA RAM Access Test
3,0> DMMU TLB TAGS Access Test
3,1> DMMU TLB TAGS Access Test
3,0> IMMU Registers Access Test
3,1> IMMU Registers Access Test
3,0> IMMU TLB DATA RAM Access Test
3,1> IMMU TLB DATA RAM Access Test
3,0> IMMU TLB TAGS Access Test
3,1> IMMU TLB TAGS Access Test
3,0> Set CPU UPA Config and Init SDB Data
3,0>Board 3 MMU Enable Test
3,0> DMMU Init
3,1> Set CPU UPA Config and Init SDB Data
3,0> IMMU Init
3,1>Board 3 MMU Enable Test
3,1> DMMU Init
3,0> Mapping Selftest Enabling MMUs
3,1> IMMU Init
3,0>Board 3 Ecache Test
3,0> Ecache Probe
3,1> Mapping Selftest Enabling MMUs
3,0> Ecache Tags
3,1>Board 3 Ecache Test
3,1> Ecache Probe
3,1> Ecache Tags
3,0> Ecache Quick Verify
3,1> Ecache Quick Verify
3,0> Ecache Init
3,1> Ecache Init
3,0> Ecache RAM
3,1> Ecache RAM
3,0> Ecache Address Line
3,0>Board 3 FPU Functional Test
3,0> FPU Enable
3,1> Ecache Address Line3,0>Board 3 Board Master Select Test
3,0> Selecting a Board Master
3,1>Board 3 FPU Functional Test
```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```
3,1>    FPU Enable
3,0>Board 3 FireHose Devices Test
3,0>Board 3 Address Controller Test
3,1>Board 3 Board Master Select Test
3,0>    AC Registers Test
3,1>    Selecting a Board Master
3,0>    AC Initialization
3,0>    Memory Registers Initialization Test
3,0>    AC DTAG Init
3,0>Board 3 Dual Tags Test
3,0>    AC DTAG Init
3,0>Board 3 FireHose Controller Test
3,0>    FHC Initialization
3,0>Board 3 JTAG Test
3,0>    Verify System Board Scan Ring
3,0>Board 3 Centerplane Test
3,0>    Centerplane and Arbiter Check Test
3,0>Setting JTAG Master
3,0>Clear JTAG Master
3,0>    Centerplane Join
3,0>Board 3 Setup Cache Size Test
3,0>    Setting Up Cache Size
3,0>Board 3 System Master Select Test
3,0>    Setting System Master
3,0>POST Master Selected (XIR,JTAG,CENTRAL)
3,0>Board 16 Clock Board Serial Ports Test
3,0>Board 16 NVRAM Devices Test
3,0>    M48T59 (TOD) Init
3,0>Board 3 System Board Probe Test
3,0>    Probing all CPU/Memory BDA
3,0>    Probing System Boards
3,0>    Probing CPU Module JTAG Rings
3,0>Setting System Clock Frequency
3,0>    CPU mid 6 (167Mhz) Checked in OK
3,0>    CPU mid 7 (167Mhz) Checked in OK
3,0>    CPU mid 10 (167Mhz) Checked in OK
3,0>    CPU mid 11 (167Mhz) Checked in OK
3,0>    CPU mid 14 (167Mhz) Checked in OK3,0>    CPU mid 15 (167Mhz)
Checked in OK
```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```

3,0>System Frequency, fcpu = 167Mhz (6ns) fsys = 83Mhz (12ns)
3,0> Remote Console is enabled
3,0>TESTING BOARD 1
3,0>Board 1 JTAG Test
3,0> Verify System Board Scan Ring
3,0>Board 1 Centerplane Test
3,0> Centerplane Check
3,0>Board 1 Address Controller Test
3,0> AC Registers Test
3,0> AC Initialization
3,0> Memory Registers Initialization Test
3,0> AC DTAG Init
3,0>Board 1 FireHose Controller Test
3,0> FHC Initialization
3,0>Board 1 I/O FEPROM Test
3,0>Board 1 NVRAM Devices Test
3,0> M48T59 (TOD) Init
3,0>Re-mapping to Local Device Space
3,0>Enable AC Control Parity
3,0>Init Counters for Hotplug
3,0>Begin Central Space Serial Port access
3,0>Board 3 Cross Calls Test
3,0>Board 3 Environmental Probe Test
3,0> Environmental Probe
3,0>Checking Power Supply Configuration
3,0>Power is more than adequate, load 4 ps 3
3,0>Board 3 Probing Memory SIMMS Test
3,0> Probe SIMMID
3,0> Populated Memory Bank Status
3,0> bd # Size Address Way Status
3,0> 3 256 Normal
3,0> 3 256 Normal
3,0> 5 256 Normal
3,0> 5 256 Normal
3,0> 7 256 Normal
3,0> 7 256 Normal
3,0>Board 3 Memory Configuration Test
3,0> Memory Interleaving
3,0> Total banks with 8MB SIMMs = 0

```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```
3,0> Total banks with 32MB SIMMs = 6
3,0> Total banks with 128MB SIMMs = 0
3,0> Overall memory default speed = 60ns
3,0>Do OPTIMAL INTLV
3,0> Board 3 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Board 5 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Board 7 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Board 3 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Board 5 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Board 7 AC rev 4 RCTIME = 0 (Tras 71 Fcpu 167 Ratio 2)
3,0> Memory Refresh Enable
3,0>System Frequency, fcpu = 167Mhz (6ns) fsys = 83Mhz (12ns)
3,0>Board 3 SIMMs Test
3,0> MP Memory SIMM Clear
3,0> Memory Size is 1536Mbytes
3,0> CPU MID 7 clearing 00000000.00004000 to 00000000.10000000
3,0> CPU MID 10 clearing 00000000.10000000 to 00000000.20000000
3,0> CPU MID 11 clearing 00000000.20000000 to 00000000.30000000
3,0> CPU MID 14 clearing 00000000.30000000 to 00000000.40000000
3,0> CPU MID 15 clearing 00000000.40000000 to 00000000.50000000
3,0> CPU MID 6 clearing 00000000.50000000 to 00000000.600000003,0>
CPU MID 6 clearing 00000000.00000000 to 00000000.00004000
3,0> MP Memory SIMM (6N RAM Patterns) Test
3,0> Memory Size is 8Mbytes
3,0> CPU MID 7 testing 00000000.00000000 to 00000000.00100000
3,0> CPU MID 10 testing 00000000.00100000 to 00000000.00200000
3,0> CPU MID 11 testing 00000000.00200000 to 00000000.00300000
3,0> CPU MID 14 testing 00000000.00300000 to 00000000.00400000
3,0> CPU MID 15 testing 00000000.00400000 to 00000000.00500000
3,0> CPU MID 6 testing 00000000.00500000 to 00000000.00800000
3,0>TESTING IO BOARD 1
3,0>@(#) iPOST 1.1.4 1/23/1996 06:28 PM
3,0> TESTING IO BOARD 1 ASICs
3,0> TESTING SysIO Port 1
3,0>Board 1 SysIO Registers Test
3,0> SysIO Register Initialization
3,0> SysIO RAM Initialization
3,0>Board 1 SysIO Functional Test
```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```
3,0> Clear Interrupt Map and State Registers
3,0> SysIO Register Initialization
3,0> SysIO RAM Initialization
3,0>Board 1 OnBoard IO Chipset (FEPS) Test
3,0>Board 1 OnBoard IO Chipset (SOC) Test
3,0> PROBING FFB
3,0>Board 1 FFB Probe Test
3,0>IO BOARD 1 TESTED
3,0>Probing for Disk System boards
3,0>
3,0> System Board Status
3,0>-----
3,0> Slot      Board Status      Board Type      Failures
3,0>-----3,0>
0 | Not installed |          |
3,0> 1 | Normal        | IO Type 2     |
3,0> 2 | Not installed |              |
3,0> 3 | Normal        | CPU/Memory    |
3,0> 4 | Not installed |              |
3,0> 5 | Normal        | CPU/Memory    |
3,0> 6 | Not installed |              |
3,0> 7 | Normal        | CPU/Memory    |
3,0> 16 | Normal       | Clock Board   |
3,0>-----
3,0>
```

## UltraSPARC Sbus-based Architecture

### POST Sample Output (Continued)

```

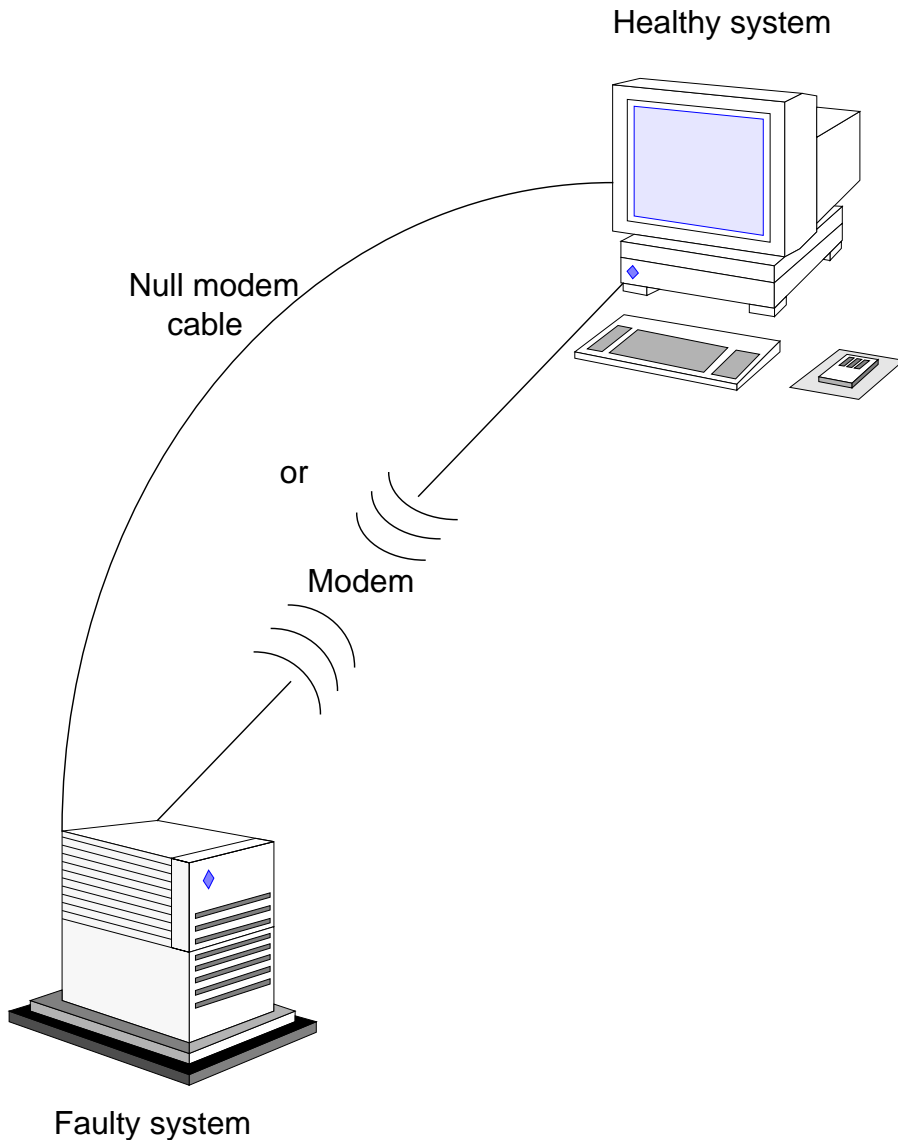
3,0> CPU Module Status
3,0>-----
3,0> MID  OK  Cache  Speed  Version
3,0>-----
3,0>  6  | y  | 512  | 167  | 00170010.22000507
3,0>  7  | y  | 512  | 167  | 00170010.22000507
3,0> 10  | y  | 512  | 167  | 00170010.22000507
3,0> 11  | y  | 512  | 167  | 00170010.22000507
3,0> 14  | y  | 512  | 167  | 00170010.22000507
3,0> 15  | y  | 512  | 167  | 00170010.22000507
3,0>-----
3,0>System Frequency, fcpu = 167Mhz (6ns) fsys = 83Mhz (12ns)
3,0> Populated Memory Bank Status
3,0>          bd #      Size      Address Way      Status
3,0>          3         256         0         4         Normal
3,0>          3         256         3         4         Normal
3,0>          5         256         1         4         Normal
3,0>          5         256         0         2         Normal
3,0>          7         256         2         4         Normal3,0>          7
256      1         2         Normal
3,0>
3,0>
3,0>          POST COMPLETE
3,0>Entering OBP

```

# POST Diagnostic Workshop Using tip

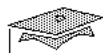
## Using Terminal Interface Protocol (TIP) for Remote Diagnostics

You can use a null modem cable or a modem with TIP to remotely troubleshoot a faulty system.



**Figure 3-9** Using TIP for Remote Diagnostics





Sun Educational Services

## POST tip Commands

- To send a break through the `tip` window, type  
~#
- To interrupt a test, press Control-c.
- To exit from `tip`, type  
~.  
or  
~^D (tilde Control-d)
- To see a list of `tip` commands, type  
~?

## POST tip Commands



---

**Warning** – Do not exit a `tip` window by killing processes, quitting the shell, or by pressing Stop-a (L1-a); these actions disable future `tip` functions.

---

To send a break through the `tip` window (Stop-a or L1-a key remote equivalent), type

~#

To interrupt a test, press Control-c.

To exit from `tip`, type

~.

## *POST tip Commands*

or

`~^D` (tilde Control-d)

To see a list of `tip` commands, type

`~?`

For more information on the `tip` command, refer to the on-line man pages.

## *Exercise: Using `tip` to Observe POST Diagnostics*



**Discussion** – The objective of this tutorial lab session is to perform diagnostics remotely between two classroom machines. The procedure and the skills used here can be applied to any situation that warrants a remote diagnostic approach to fault analysis.

### *Preparation*

The instructor breaks the class into small groups. Two machines and a null modem cable are needed for each group. It may be helpful to review the OBP variables `diag-switch?`, `ttya-mode`, and `diag-device` which are set or referenced within the remote diagnostic procedure. (See the `eeprom` man page.)

A console monitor or an American Standard Code for Information Interchange (ASCII) terminal can be used for remote diagnostic sessions. This lab procedure can be performed on both `sun4m` and `sun4u` architectures.

---

## Exercise: Using `tip` to Observe POST Diagnostics

### Tasks

Use the following procedure to set up a remote diagnostic session:

---

**Note** – Before you begin, make sure that the healthy system has the Solaris operating environment booted to multiuser mode and has a window system running.

---

1. Connect an RS-232C null modem cable to port B of the functional workstation.
2. Connect the other end of the cable to port A of the faulty machine.
3. Halt the faulty machine by pressing the Stop-a (L1-a) key sequence.
4. Set the `diag-switch?` parameters to true on the faulty machine. If you cannot type, this parameter can also be set to true by pressing the Stop-d keys while turning on the power.

---

**Note** – SS1000, SC2000, 600mp, and Sun-4 systems have a hardware diagnostic switch.

---

5. Use the commands

```
ok setenv diag-switch? true
ok reset
```
6. Turn off the faulty system to prevent blowing the keyboard fuse on some systems.
7. Disconnect the keyboard from the back of the faulty system (output to `ttya`). Remember to turn power off when you reconnect the keyboard.
8. Start your windows environment on the functional machine if not already started, and bring up a Shell Tool from the Programs menu. (You can run the `tip` command in a non-windowed environment, but there is a danger that if `tip` hangs, there will be no way to get into the system to release or kill it.)

---

## Exercise: Using `tip` to Observe POST Diagnostics

---

**Note** – The hardware argument for `ttya-mode` in OBP specifies that the `tip` command expects 9600 baud, 8 data bits, and 1 stop bit at port B on the CPU board. It is not a coincidence that these are the parameters set for port A when a machine is turned on without a keyboard.

---

9. If you have a SPARC 5 with two serial ports on your functional system, no edits are needed in `/etc/remote`. If your system is an Ultra 5, or if port A is the only available port, edit the `/etc/remote` file for port A on “good” system.

Change

```
:dv=/dev/term/b:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D
```

to

```
:dv=/dev/term/a:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D
```

10. In the Shell Tool window, type the following command:

```
# tip hardware
```

---

**Note** – The system should respond with *connected*. If it does not, some *likely causes* are:

- Wrong port selected, physically or logically in `/etc/remote`.
  - Selected port is already active. (Bring up `admintool` and make sure the port is disabled.)
  - A `/var/spool/locks/LCK` file exists from a previous `tip` or `uucp` session (often because someone did not properly exit `tip` with a `^D` or `~.`).
-

---

## Exercise: Using `tip` to Observe POST Diagnostics

11. Turn on the faulty system. At this point, you should observe the power-on diagnostic messages in the Shell Tool window of the healthy system. If not, some *likely causes* are
  - ▼ Wrong physical or logical port selected at either end
  - ▼ Faulty null modem cable
  - ▼ Machine is not in `diag` mode or still has keyboard plugged in
12. When the diagnostics finish, note any errors (see note).

---

**Note** – If the systems in your classroom are connected to a jump server, the system locates the server as it tries to boot over the net, which automatically becomes the default when `diag-switch?` is set to true. Usually, this invokes an installation procedure, which should be aborted. If the classroom systems are not connected to a server, an error can occur when the attempt to boot over the network fails.

---

13. Press `~.` to end the `tip` session. (See “POST `tip` Commands.”)
14. If your classroom machine is an Ultra, view the saved POST results. Type

```
prtdiag -v
```
15. Bring the faulty machine back to a running state:
  - a. Turn off the system and plug in the keyboard.
  - b. Turn on the system and wait for the `ok` prompt.
  - c. Run the following commands:

```
ok setenv diag-switch? false
ok reset
```
  - d. Verify that the machine boots fully again.

---

## *Exercise: Using tip to Observe POST Diagnostics*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

## *Check Your Progress*

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Describe the capabilities and limitations of the POSTs in identifying and resolving system faults
- Describe different ways to view the POST
- Configure the `/etc/remote` file on a console server to enable the use of `tip` in a remote diagnostic session
- View and interpret POST output
- Describe the function of the `prtdiag -v` command



---

## *Think Beyond*

Based on your experience with POST diagnostics and your background, are there hardware conditions that cannot be analyzed through POST diagnostics? Is it possible to run the remote diagnostic procedure presented in this module with an ASCII terminal instead of a console monitor?



## *Objectives*

Upon completion of this module, you should be able to use OBP commands to do the following:

- Gather general information about the system
- Define the meaning of the non-volatile read access memory (NVRAM) variables
- Display and capture the names of the devices in the system device tree, and display their attributes
- Test devices using the device path, node name, and device alias
- Generate and test a PROM device alias
- Alter and display NVRAM settings, and reset to the defaults
- Use the `eeprom` command to examine and define NVRAM

## Relevance



**Discussion** – This module covers the commands which are available in OBP and useful in fault analysis. OBP commands that display, test, and change information can be used to diagnose faults related to hardware or devices.

Before beginning this module, consider the following questions in relation to the work you do on Solaris systems:

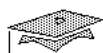
- What OBP commands do you find most useful?
- Are there administrative tasks that cannot be accomplished in UNIX and must be done in the OBP environment?
- What revision of OBP are you currently running?
- Are there any OBP variables that you do not fully understand?
- Have you experienced any boot failures? If so, how did you troubleshoot and repair the problem?

## References



**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- *Field Engineers Handbook*, part number 800-4006-16
- *OpenBoot Command Reference*, part number 800-6076
- *OpenBoot 2.x Quick Reference Card*, part number 802-1958
- *OpenBoot 3.x Quick Reference Card*, part number 802-3240
- *Solaris 7 Sun Hardware Platform Guide*, part number 805-4456-10



*Sun Educational Services*

## Functions and Capabilities of OBP

Two chips on each system board:

- The boot PROM itself
- A non-volatile random access memory (NVRAM)

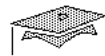
### *Functions and Capabilities of Open Boot PROM*

The OBP consists of two chips on each system board:

- The boot PROM itself
- A non-volatile random access memory (NVRAM)

The boot PROM has extensive firmware and FORTH-code writing capabilities that allows access to user-written boot drivers and extended diagnostics (Figure 4-1).

The NVRAM has user-definable system parameters and writable areas for user-controlled diagnostics, macros, or useful settings such as device aliases. The NVRAM also contains system-identification information as shown in Figure 4-1 (Host ID database).



## OBP Features

- Ability to read plug-in drivers and diagnostics from probed devices
- F(ORTH) code interpreter
- Device tree hierarchy
- Diagnostic, informational commands, and PROM variables
- Restricted monitor
- System initialization
- Power-On Self Tests (POSTs)

## *Functions and Capabilities of the Open Boot PROM*

### *OBP Features*

The OBP has the following features:

- The ability to read plug-in device drivers and diagnostics from probed devices. (Early Sun machines required all boot drivers and diagnostics to be completely written in the boot PROM.)
- A F(ORTH) code interpreter to facilitate writing and downloading drivers, diagnostics, and parameters
- A device tree with a data structure hierarchy, similar to UNIX, for locating device addresses
- Diagnostic and informational commands, and system configuration parameters (PROM variables)

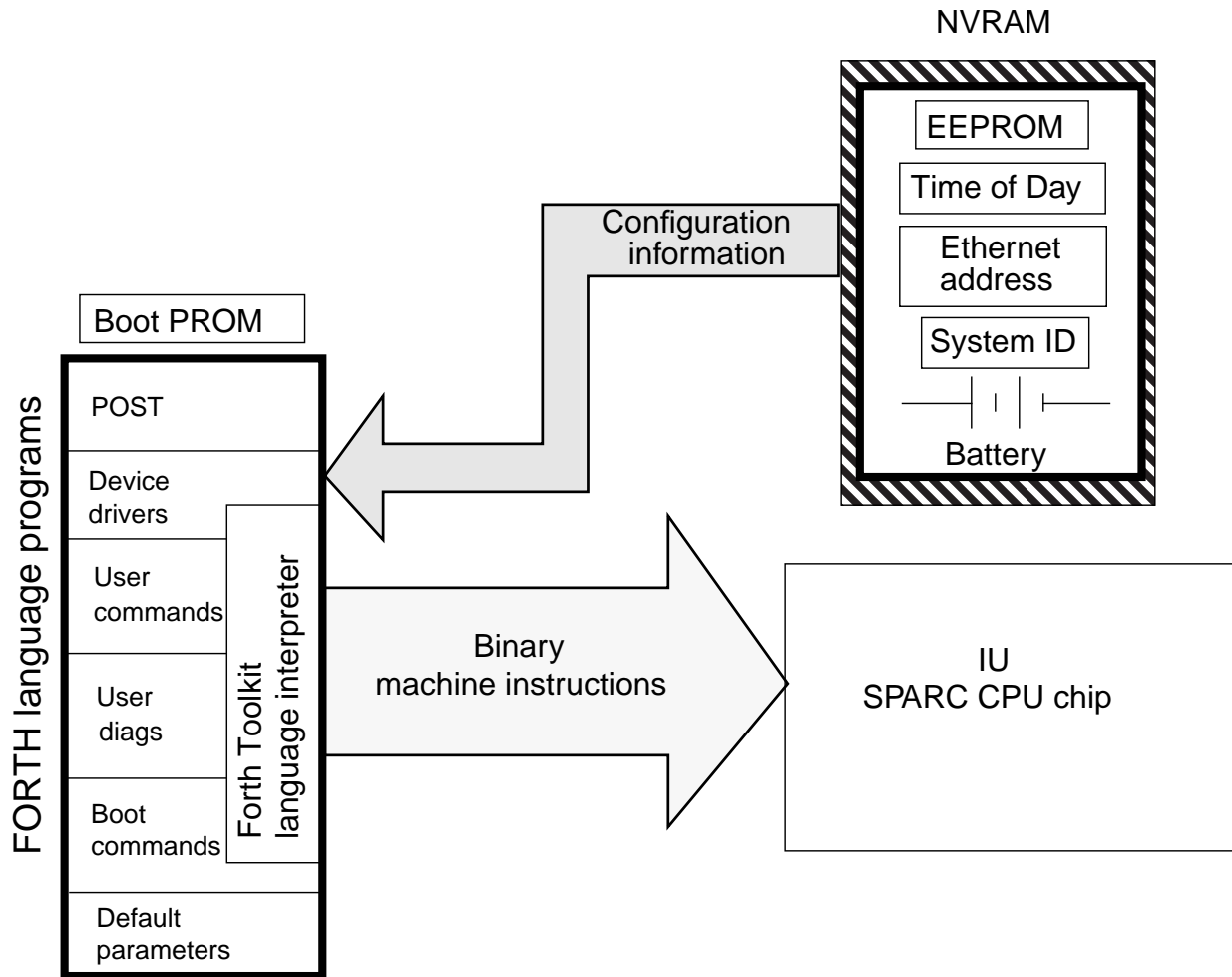
---

## *Functions and Capabilities of the Open Boot PROM*

### *OBP Features (Continued)*

- A restricted monitor (passworded security to limit or disallow system access from the ok prompt)
- System initialization (passing hardware configuration to and loads the operating system)
- Power-on self tests (POSTs)

# Open Boot PROM



**Figure 4-1** Information in OBP and NVRAM

The following information is contained in the Host ID:

- A 48-bit hardware Ethernet address
- CPU-type code
- Host serial number



## NVRAM Contents – Sun4m Architecture

You can use the `printenv` command at the monitor prompt to see the various NVRAM parameters (variables) and default values.

```
ok printenv
```

Parameter Name	Value	Default Value
tpe-link-test?	true	true
output-device	screen	screen
input-device	keyboard	keyboard
keyboard-click?	false	false
keymap		
ttyb-rts-dtr-off	false	false
ttyb-ignore-cd	true	true
ttya-rts-dtr-off	false	false
ttya-ignore-cd	true	true
ttyb-mode	9600,8,n,1,-	9600,8,n,1,-
ttya-mode	9600,8,n,1,-	9600,8,n,1,-
fcode-debug?	false	false
local-mac-address?	false	false
screen-#columns	80	80
screen-#rows	34	34
selftest-#megs	1	1
scsi-initiator-id	7	7
silent-mode?	false	false
auto-boot?	true	true
watchdog-reboot?	false	false
diag-file		
diag-device	net	net
boot-file		
boot-device	disk net	disk net
sbus-probe-list	541230	541230
use-nvramrc?	false	false
nvramrc		
sunmon-compat?	false	false
security-mode	none	none
security-password		
security-#badlogins	0	<no default>
diag-level	min	min
oem-logo		<no default>
oem-logo?	false	false
oem-banner		<no default>
oem-banner?	false	false
hardware-revision		<no default>

## *NVRAM Contents – Sun4m Architecture*

last-hardware-update		<no default>
testarea	0	0
mfg-switch?	false	false
diag-switch?	true	false

## NVRAM Contents – Ultra Workstations

You can use the `printenv` command at the monitor prompt to see the various NVRAM parameters (variables) and default values.

```
<#2>ok printenv
```

<u>Variable Name</u>	<u>Value</u>	<u>Default Value</u>
scsi-initiator-id	7	7
tpe-link-test	true	true
keyboard-click?	false	false
keymap		
ttyb-rts-dtr-off	false	false
ttyb-ignore-cd	true	true
ttya-ignore-cd	true	true
ttyb-mode	9600,8,n,1,-	9600,8,n,1,-
ttya-mode	9600,8,n,1,-	9600,8,n,1,-
pcia-probe-list	1,2,3,4	1,2,3,4
pcib-probe-list	1,2,3	1,2,3
mfg-mode	off	off
diag-level	max	max
#power-cycles	10	
system-board-serial#	5013082149391	
system-board-date	356b9fd2	
fcode-debug?	false	false
output-device	screen	screen
input-device	keyboard	keyboard
load-base	16384	16384

---

## NVRAM Contents – Ultra Workstations

<u>Variable Name</u>	<u>Value</u>	<u>Default Value</u>
boot-command	boot	boot
auto-boot?	true	true
watchdog-reboot?	false	false
diag-file		
diag-device	net	net
boot-file		
boot-device	disk net	disk net
local-mac-address?	false	false
ansi-terminal?	true	true
screen-#columns	80	80
screen-#rows	34	34
silent-mode?	false	false
use-nvramrc?	false	false
nvramrc		
security-mode	none	
security-passwd		
security-#badlogins	0	
oem-logo		
oem-logo?	false	false
oem-banner		
oem-banner?	false	false
hardware-revision		
last-hardware-update		
diag-switch	false	false

---

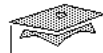
**Note** – On Ultra 1 and 2 systems, you have the entry `sbus-probe-list` instead of the entries `pcia-probe-list` and `pcib-probe-list`.

---

## The eeprom Command

The superuser can display and change PROM variable settings using the `eeprom` command. Some of the settings and their associated meanings are

Parameter	Meaning
<code>auto-boot?</code>	If true, boot automatically after power-on or reset. Default is true.
<code>diag-device</code>	Diagnostic boot device to use when <code>diag-switch</code> is set to true. Default is <code>net</code> .
<code>diag-switch?</code>	If true, run in diagnostic mode. Defaults to false.
<code>oem-logo?</code>	If true, use custom other equipment manufacturer (OEM) logo. Default is false (use Sun logo).
<code>sbus-probe-list</code> <code>pcia-probe-list</code> <code>pcia-probe-list</code>	Which system bus slots to probe, in what order. Defaults to 0123.
<code>security-mode</code>	Firmware security level. If set to <code>command</code> or <code>full</code> , the system prompts for PROM security password.
<code>selftest#megs</code>	Mbytes of random access memory (RAM) to test (in POST, not extended POST). Default is one.
<code>tpe-link-test?</code>	Enable 10baseT link test for built-in twisted pair Ethernet. Default is true.
<code>watchdog-reboot?</code>	If true, automatically reboot after a watchdog reset. Default is false.



## Changing PROM Variables With eeprom

- `/usr/sbin/eeprom` (*lists current settings*)
- `/usr/sbin/eeprom diag-switch?=true`  
(*changes settings*)

### *The eeprom Command*

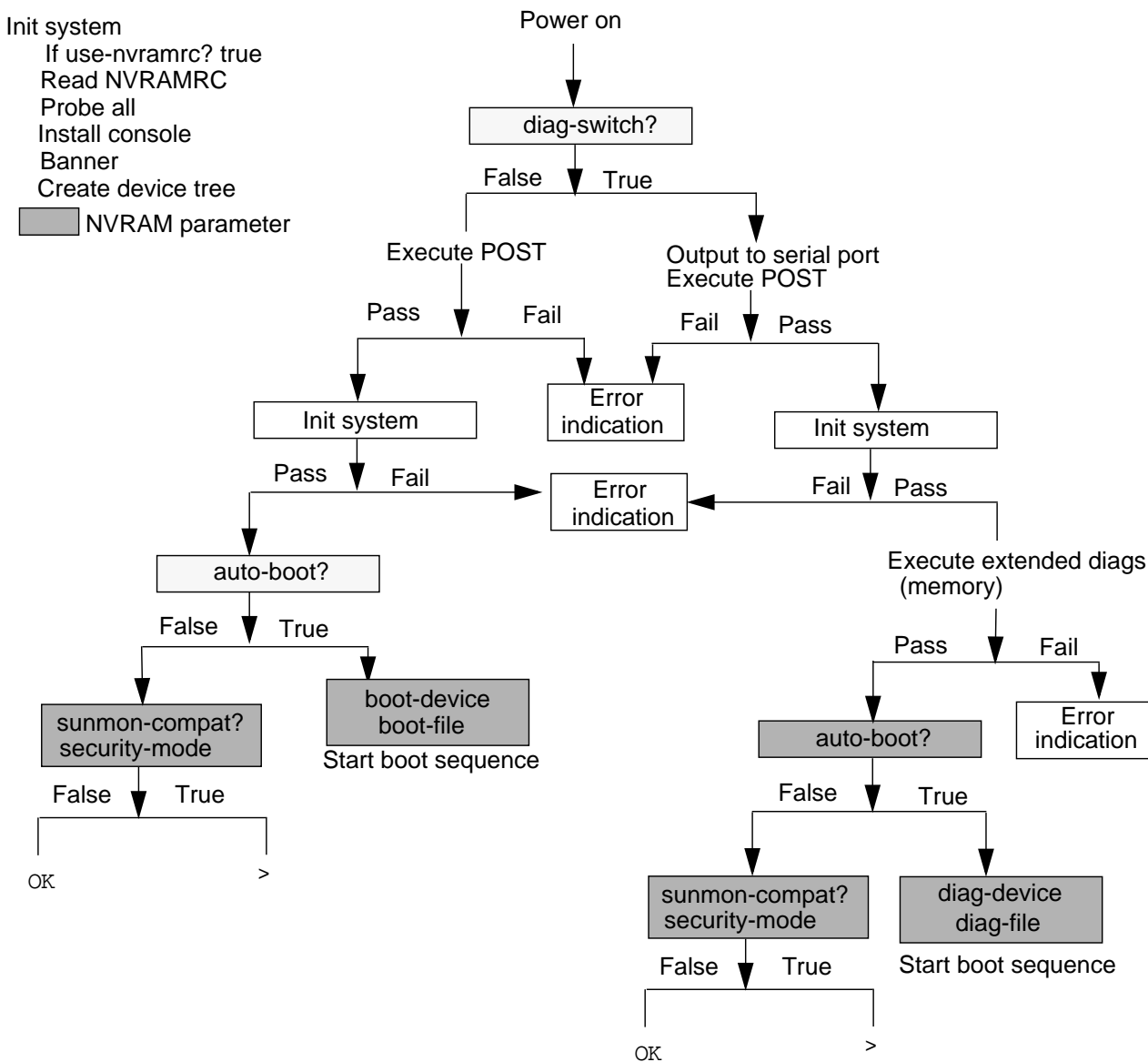
#### *Changing PROM Variables With eeprom*

The following two examples demonstrate the syntax to use with `eeprom` when changing parameters:

- `/usr/sbin/eeprom`
- `/usr/sbin/eeprom diag-switch?=true`

## Events During Power On

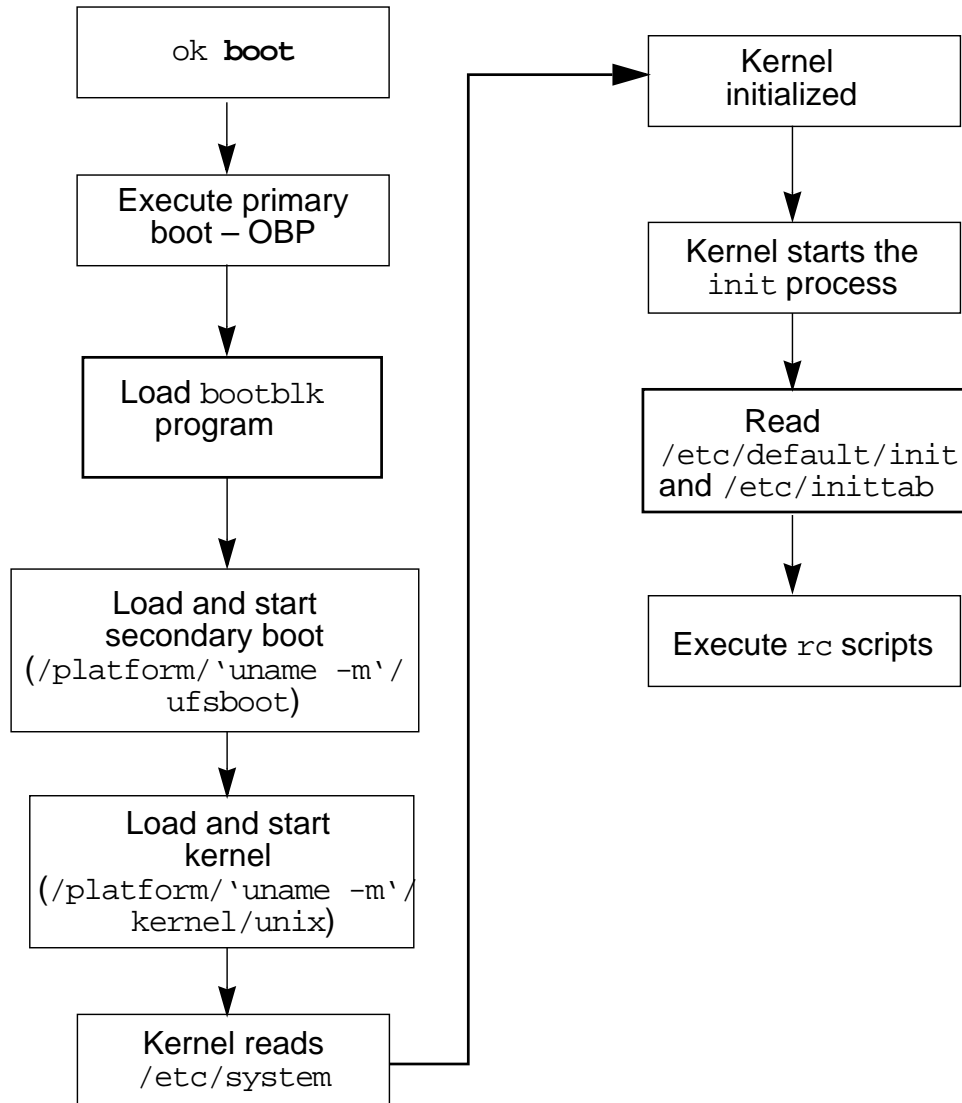
Figure 4-2 illustrates the various parameters that are evaluated during the powering on of a system.



**Figure 4-2** Events During Power On

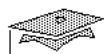
## Default Boot Sequence

Figure 4-3 illustrates a flow chart of the default boot sequence.



**Figure 4-3** Default Boot Sequence





*Sun Educational Services*

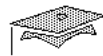
## Default Boot Sequence

1. Enter the `boot` command at the `ok` prompt.
2. Use the `boot` command to locate the primary boot code.
3. Use the `ufsboot` program to load kernel core image files.
  - ▼ 32-bit SPARC and X86 –  
`/platform/platform_name/kernel/unix` and  
`/kernel/genunix`
  - ▼ 64-bit SPARC –  
`/platform/`uname -m`/kernel/sparcv9/unix` and  
`/kernel/genunix`

## Default Boot Sequence

The following steps explain in more detail the sequential operations represented in the flow chart in Figure 4-3.

1. Entering the `boot` command at the `ok` prompt causes the system to find the definition for the NVRAM variable `boot-device` and attempt to boot from the defined device.
2. The `boot` command locates the primary boot code, installed in sectors 1–15 of the system disk by the `installboot` utility. This primary boot code contains the instructions to execute the second level boot program, `/platform/arch_name/ufsboot`.
3. The `ufsboot` program loads the following kernel core image files:
  - ▼ 32-bit SPARC and X86  
`/platform/platform_name/kernel/unix` and  
`/kernel/genunix`
  - ▼ 64-bit SPARC  
`/platform/`uname -m`/kernel/sparcv9/unix` and  
`/kernel/genunix`



## Default Boot Sequence

4. Read the `/etc/system` file as part of kernel initialization.
5. After creating the `sched` process (PID 0), the kernel creates process PID 1, which is `init`. The `init` daemon reads `/etc/inittab` and proceeds according to instructions in that file.
6. Use the `rc` scripts contain instructions to run additional scripts in the `rc` directories `/etc/rc0`, `/etc/rcS`, `/etc/rc2`, and `/etc/rc3`.

### *Default Boot Sequence*

4. The `/etc/system` file is read as part of kernel initialization. If no variables are explicitly set in this file, defaults as defined in the kernel are used. The following variables can be customized in `/etc/system`:
  - ▼ `moddir` – Change or modify paths of loadable kernel modules
  - ▼ `forceload` – Force a module to be loaded during boot
  - ▼ `exclude` – Force the exclusion of a particular kernel module at boot
  - ▼ `rootfs` – Set the file system type for the root file system
  - ▼ `rootdev` – Specify the path of the physical device for root
  - ▼ Kernel variables – Set values for tunable kernel parameters

---

## *Default Boot Sequence*

5. After creating the sched process and the process identification (PID) 0, the kernel creates process PID 1, which is `init`. The `init` daemon reads `/etc/inittab` and proceeds according to instructions in that file. The system initialization run level and the instructions to execute the `rc(run control)` scripts in `/sbin` are defined in the `inittab` file.
6. The `rc` scripts contain instructions to run additional scripts in the `rc` directories `/etc/rc0`, `/etc/rcS`, `/etc/rc2`, and `/etc/rc3`. A list of the scripts standardly provided in these directories is on the next page.

## The rc Files and Directories

The init daemon runs the scripts in `/etc/rcS.d`, `/etc/rc2.d`, and `/etc/rc3.d` toward the end of the boot procedure.

```
# ls /etc/rcS.d
```

```
K07dmi          K39lp          K41rpc         S35cacheos.sh
K07snmpdx       K39spc         K42inetsvc    S40standardmounts
K10dtlogin      K40cron        K43inet        S41cachesfs.root
K28nfs.server   K40nscd        README         S50drvconfig
K33audit        K40syslog      S10inipcmcia  S60devlinks
K35volmgt       K40xntpd       S30rootusr.sh S60pcmcia
K36sendmail     K41autofs      S33keymap.sh  S70buildmnttab.sh
K36utmpd
```

The `/etc/rc2.d` directory contains files used to start processes that should be running in run level 2. (It also contains files used to stop processes that should not be running at run level 2. These files are not shown here.)

```
# ls /etc/rc2.d
```

```
K07dmi          S69inet        S74syslog      S88sendmail
K07snmpdx       S70uucp        S74xntpd       S88utmpd
K28nfs.server   S71rpc         S75cron        S89bdconfig
ReadME          S71sysid.sys   S75savecore    S92volmgt
S01MOUNTFSYS    S72autoinstall S76nscd        S93cacheos.finish
S05RMTMPFILES   S72inetsvc     S80lp          S99audit
S20syssetup     S73cachefs.daemon S80PRESERVE    S99dtlogin
S21perf         S73nfs.client  S80spc
S30sysid.net    S74autofs
```

## The rc Directories and Files

Run level 3 is the default run level. The files in the `/etc/rc3.d` directory are executed when the system begins run level 3.

```
# ls /etc/rc3.d
```

```
README          S15nfs.server  S32pcnfs      S76snmpdx     S77dmi
```

The files in the `/etc/rc0.d` directory are executed during shutdown procedures, including a shutdown to single-user mode.

```
# ls /etc/rc0.d
```

```
K00ANNOUNCE     K35volmgt     K40cron       K41nfs.client
K07dmi          K36sendmail   K40nscd       K41rpc
K07snmpdx       K36utmpd      K40syslog     K42inetsvc
K10dtlogin      K39lp         K40xntpd      K43inet
K28nfs.server   K39spc        K41autofs     K78pcmcia
K33audit
```



## OBP Device Tree Navigation – UltraSPARC Workstation

- The `cd`, `ls`, and `pwd` commands are used to move around the device tree structure.
- The `dev device_pathname` command selects a node for examination.
- The `.properties` command shows the properties that define a particular node
- The `device-end` command unselects a device node.

### *OBP Device Tree Navigation – UltraSPARC Workstation*

The following points help to clarify the OBP commands used in the example to navigate the device tree on an Ultra-5 workstation.

- The `cd`, `ls`, and `pwd` commands are used to move around the device tree structure.
- The `dev device_pathname` command selects a node for examination.
- The `.properties` command shows the properties that define a particular node, and are passed to the OS at boot time.

## *OBP Device Tree Navigation – UltraSPARC Workstation*

- The device-end command unselects a device node.

```
<#0>ok cd /
<#0>ok ls
f006ce08 SUNW,UltraSPARC-IIia0,0
f005f59c pci@1f,0
f004d448 virtual-memory
foo4cd68 memory@0,10000000
f002cc3c aliases
f002cb04 openprom
f002ca98 chosen
f002ca28 package
```

```
<#0>cd pcia1f,0
<#0>ok ls
f0060378 pci@1
f005fd90 pci@1,1
```

```
<#0>ok cd pci@1
<#0>ok pwd
/pcia1f,0/pci@1
<#0>ok ls
f0083ac8 scsi@1,1
f007db18 scsi@1
```

---

## *OBP Device Tree Navigation – UltraSPARC Workstation*

```
<#0>cd pci@1,1
<#0>ok ls
f007a128 ide@3
f0075828 SUNW,m64b@2
f006d820 network@1
f00610e0 ebus@1

<#0>ok cd ide@3
<#0>ok pwd
/pcialf,0/pci@1,1/ide@3
<#0>ok ls
f007d0a4 cdrom
f007c9f8 disk
<#0>ok cd ..
<#0>ok cd ebus@1
<#0>ok pwd
/pcialf,0/pci@1,1/ebus@1

<#0>ok ls
f0066838 SUNW,CS4231@14,200000
f006677c flashprom@10,0
f0066694 eeprom@14,0
f0064ba4 fdthree@14,3023f0f006
f0064aa4 ecpp@14,3043bc
f0064988 su@14,3062f8
f0063168 su@14,3083f8
f0061840 se@14,400000
f00617ac SUNW,pll@14,504000
f00616dc power@14,724000
f006161c auxio@14,726000

<#0>ok
```



## *OBP Device Tree Navigation – SPARCstation 1000 System*

The following example shows how to navigate through the `sd` and `st` devices on a SPARCstation 1000:

```
<#0>ok cd /
<#0>ok ls
ffda476c io-unit@f,e1200000
ffd91c10 io-unit@f,e0200000
ffd8d2f4 mem-unit@f,e1100000
ffd8d210 mem-unit@f,e0100000
ffd8cebc cpu-unit@f,e1800000
ffd8cb68 cpu-unit@f,e1000000
ffd8c814 cpu-unit@f,e0800000
ffd8c4c0 cpu-unit@f,e0000000
ffd839a8 boards
ffd712fc openprom
ffd702bc virtual-memory@0,0
ffd7016c memory@0,0
ffd625cc aliases
ffd6257c options
ffd6252c packages

<#0>ok cd io-unit@f,e1200000
<#0>ok ls
ffda4d20 sbi@0,0

<#0>ok cd sbi
<#0>ok ls
ffdb0ffc lebuffer@1,40000
ffdac1f4 dma@1,81000
ffda9ff4 lebuffer@0,40000
ffda51ec dma@0,81000

<#0>ok cd dma@1,81000
<#0>ok ls
ffdac878 esp@1,80000
<#0>ok cd esp@1,80000
<#0>ok ls
ffdb05b4 st
ffaefef4 sd
```



## OBP Features – Ultra Enterprise Servers

- Hardware management
- Environmental monitoring
- Clock copy
- Externally initiated reset (XIR)
  - The clock board button
  - The key sequence <CR> <CR> <Ctrl-Shift-x>
  - The `.xir-state-all` command displays CPU status
- POST control

### *OBP Features – Ultra Enterprise Servers*

The following OBP features and commands are available on the Ultra Enterprise™ server family (3x00, 4x00, 5x00, and 6x00):

- Hardware management
- Environmental monitoring
- Clock copy
- Externally initiated Reset (XIR)
  - ▼ The clock board button
  - ▼ The key sequence <CR> <CR> <Ctrl-Shift-x> from a remote console
  - ▼ The `.xir-state-all` command
- POST control

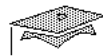
---

## *NVRAM Parameters – Ultra Enterprise Servers*

Some Ultra Enterprise features in OBP are controlled using the following parameters:

Parameter	Meaning
<code>disabled-board-list</code>	Specifies numerically identified boards to be disabled by OBP
<code>disabled memory-list</code>	Specifies numerically identified boards for which memory is disabled
<code>memory-interleave</code>	Enables or disables memory interleave
<code>disable-environmental-monitor</code>	Turns off monitoring of power supplies, board temperatures, and hot-plugged boards
<code>copy-clock-tod-to-io-boards</code>	Copies the time-of-day (TOD) clock from the clock board to an I/O board
<code>copy-io-board-tod-to-clock-tod</code>	Copies the backup of the TOD clock from an I/O board to the clock board
<code>configuration-policy</code>	Specifies how devices that fail POST are handled (disable system, board, or component level)

---



## Running OBP Commands With Key Sequences

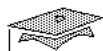
- The L1-f
- The L1-n
- The L1-d
- The L1 key runs POST in `INIT` mode, independent of security mode settings

### *Running OBP Commands With Key Sequences*

There are some useful functions that can be executed through the use of key sequences. These are helpful if NVRAM contents are corrupt, when typing at the keyboard is not possible, or when there are hardware problems.

The following key sequences can be used after power-up and until the keyboard flashes for the second time to perform the listed function. These keys are disabled if security is enabled.

- The L1-f sequence invokes FORTH command mode on `ttys` before probing the hardware. Use `fexit` to continue with initialization.
- The L1-n sequence resets NVRAM contents to defaults.
- The L1-d sequence sets the `diag-switch?` parameter to true, and enables verbose output during POST.
- The L1 key runs POST in `INIT` mode, independent of security mode settings.



Sun Educational Services

## Flash PROM Update for Sun4u Architectures

### Systems and firmware

- Ultra™-1, if firmware is lower than Revision 3.11.1, use patch number 104881.
- Ultra-2, if the firmware is lower than Revision 3.11.2, use patch number 104169.
- Ultra-4, if the firmware is lower than Revision 3.7.107, use patch number 106122.
- Ultra Enterprise™ systems, if the firmware is lower than revision 3.2.16, use patch number 103346.

## *Flash PROM Update for Sun4u Architectures*

### *Systems and Firmware*

The following Sun4u system types require the flash PROM update procedure on Solaris 7 in order to enable the 64-bit architecture:

- For Ultra-1, if the firmware is lower than Revision 3.11.1, use patch number 104881.
- For Ultra-2, if the firmware is lower than Revision 3.11.2, use patch number 104169.
- For Ultra-4, if the firmware is lower than Revision 3.7.107, use patch number 106122.
- For Ultra Enterprise systems, if the firmware is lower than Revision 3.2.16, use patch number 103346.

---

**Note** – Check the README file of the patch for any previous revisions that might be necessary for proper installation.

---



## Flash PROM Update for Sun4u Architectures

1. Capture and save NVRAM variable settings.
2. Set the write-enable jumper, J2003, on Ultra-1 and Ultra-2 systems, or turn the front panel key switch to the Diagnostic position on Ultra Enterprise systems.
3. Perform the update procedure using the bootable CD-ROM, or use the update script, obtained by pulling the appropriate patch number from:

<http://sunsolve.sun.com/sunsolve/pubpatches/patches.html#Hardware-rec>

## *Flash PROM Update for Sun4u Architectures*

### *Summary of Procedure*

The following summarizes the flash update PROM procedure:

1. Capture and save NVRAM variable settings in order to restore them after the procedure is complete.
2. Set the write-enable jumper, J2003, on Ultra-1 and Ultra-2 systems, or turn the front panel key switch to the Diagnostic position on Ultra Enterprise systems.
3. Perform the update procedure using the bootable CD-ROM or the update script, which can be obtained by downloading the appropriate patch number from

<http://sunsolve.sun.com/sunsolve/pubpatches/patches.html#Hardware-rec>

## Ultra Workstation OBP Command Examples

ok **help diag**

test <device-specification> Run selftest for specific device

Examples:

test floppy

test net

test scsi

test-all - Execute test for all devices w/selftest enabled

watch-clock - Show ticks of real-time clock

watch-net - Monitor network broadcast packets

watch-net-all - Monitor broadcast packets on all network interfaces

probe-scsi - Show attached SCSI devices

probe-scsi-all - Show attached SCSI devices for all host adapters

ok **.version**

Release 3.11 Version 12 created 1999 01/19 11:30

OBP 3.11.12 1999 01/19 11:30

POST 2.2.9 1999 01/19 7:54

ok **test net**

Testing /pci@1f,0/pci@1,1/network@1

Internal loopback test -- succeeded

Transceiver check -- passed

ok **test floppy**

Testing /pci@1f,0/ebus@1/fdthree@14,3023f0

Testing floppy disk system. A formatted disk should be in the drive.

No diskette or incorrect format.

Selftest failed. Return code = -1

ok **.enet-address**

8:0:20:9e:06:c2

ok **.speed**

CPU Speed : 300.00MHz

UPA Speed : 100.00MHz

PCI Bus A : 33MHz

PCI Bus B : 33MHz

## *Ultra Workstation OBP Command Examples*

ok **.idprom**

Format/Type: / 80 Ethernet: 8 0 20 9e d6 c2 Date:  
Serial: 9ed6c2 Checksum: a9

ok **probe-scsi-all**

/pci@1f,0/pci@1/scsi@1,1

Target 0

Unit 0 Disk Conner CFP1080E SUN1.055150

Target 4

Unit 0 Removable Tape Archive PYTHON 28454-XXX.BL

pci@1f,0/pci@1/scsi@1

ok **banner**

Sun Ultra 5/10 UPA/PCI (UltraSPARC-II; 300 MHz), Keyboard present  
Open Boot 3.11 128MB memory installed, Serial # 10409666  
Ethernet address 8:0:20:9e:d6:c2, Host ID: 809ed6c2



## SPARCstation 20 OBP Command Examples

The following example highlights user diagnostics and commands:

```
<#0> ok help diag
  Category: Diag (diagnostic routines)
test  device-specifier ( -- ) run selftest method for specified device
  Examples:
    test /memory          - test memory
    test /iommu/sbus/ledma@f,400010/le    - test net
    test floppy          - test floppy disk drive
    test net             - test net (device-specifier is an alias)
    test scsi            - test scsi (device-specifier is an alias)
watch-clock      ( -- ) show ticks of real-time clock
watch-net        ( -- ) monitor broadcast packets using auto-selected
                    interface
watch-aui        ( -- ) monitor broadcast packets using AUI interface
watch-tpe        ( -- ) monitor broadcast packets using TPE interface
watch-net-all   ( -- ) monitor broadcast packets on all net interfaces
probe-scsi       ( -- ) show attached SCSI devices
probe-scsi-all  ( -- ) show attached SCSI devices for all host adapters
test-all        ( -- ) run test for all devices with selftest method
test-memory      ( -- ) test all memory if diag-switch? is true,
                    otherwise test memory specified by selftest-#megs

<#0> ok

<#0> ok show-sbus
SBus slot f SUNW,bpp ledma le espdma esp
SBus slot e SUNW,DBRIe
SBus slot 0
SBus slot 1
SBus slot 2 cgsix
SBus slot 3
```

---

## SPARCstation 20 OBP Command Examples

```
<#0> ok probe-scsi
Target 1
  Unit 0   Disk QUANTUM P105SS 910-10-94A.1 08/31/89009030144 GENERIC
Target 3
  Unit 0   Disk      SEAGATE ST31200W SUN1.05872400795741
              Copyright (c) 1994 Seagate
              All rights reserved 0000
Target 4
  Unit 0   Removable Tape ARCHIVE VIPER 150 21531-003 SUN-03.00.00
Target 6
  Unit 0   Removable Read Only device      TOSHIBA XM-
4101TASUNSLCD108404/18/94
```

```
<#0> ok module-info
Mbus   : 50.00 MHz
Sbus   : 25.00 MHz
CPU#0  : 50.00 MHz SuperSPARC
CPU#2  : 50.00 MHz SuperSPARC
```

```
<#0> ok 2 switch-cpu
<#2> ok 0 switch-cpu
<#2> ok 1 switch-cpu
Processor #1 is not present!
```

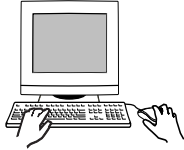
---

**Note** – Refer to the *Field Engineering Handbook*, Volume 1, “SPARC Processor Revision,” for information on the significance of these commands and their results. These commands enable you to determine the revision level of the SPARC processor and any required operating system patches.

---

---

## *Exercise: Using Ultra Workstation OBP Utilities*



**Exercise objective** – The objective of this exercise is to obtain information and perform diagnostic tests in the OBP environment. You will also display and change NVRAM variable settings.

### *Preparation*

Depending on the classroom lab equipment, select either the SPARC 5 lab or the Ultra Workstation lab, which immediately follows this section. Perform a shutdown procedure to access the OBP environments and enter commands from the `ok` prompt.

## *Exercise: Using Ultra Workstation OBP Utilities, Part 1*

### *Tasks*

In this exercise, you will use the `help` and `test` commands to perform basic diagnostic procedures on your workstation.

---

**Note** – Due to different PROM levels and architectures, the syntax for these labs can vary slightly. Refer to the OBP reference card if necessary.

---

Complete the following steps:

1. Return the machine safely to the `ok` prompt.
2. Use the `help` command to list some PROM level diagnostics.

```
ok help diag
```

3. Use the `test` command as follows:

```
ok test net
```

```
ok test-all
```

4. Run tests on devices listed with the `devalias` command. (Some devices will not have tests; that is important to know also.)

---

**Note** – If the `ok` prompt returns with no message, this means the self test found no errors.

---

---

## Exercise: Using Ultra Workstation OBP Utilities, Part 2

### Tasks

In this part,

- Gather general information about the system
- Alter necessary NVRAM settings, display the settings, and reset settings to the defaults

---

**Note** – Due to the capabilities of different architectures and PROM revision level differences, some of these suggested commands may not work on your particular lab machine. For example, `switch-cpu` probably will not work on a single CPU system.

---

1. Use the following console commands and observe the output. Determine if the results are useful.

OBP Command	Results
<code>banner</code>	
<code>help diag</code>	
<code>printenv</code>	
<code>show-tapes</code>	
<code>show-nets</code>	
<code>show-disks</code>	
<code>n switch-cpu</code> where <i>n</i> is the number 0, 1, 2, and so on.	
<code>devalias</code>	
<code>show-devs</code>	

---

## Exercise: Using Ultra Workstation OBP Utilities, Part 2

### Tasks (Continued)

<b>OBP Command</b>	<b>Results</b>
setenv diag-switch? true	
printenv diag-switch?	
set-default diag-switch?	
printenv diag-switch?	
setenv fcode-debug? true	
set-defaults	

<b>OBP Command</b>	<b>Results</b>
Optional, experiment from OBP reference card	

---

## Exercise: Using Ultra Workstation OBP Utilities, Part 3

### Tasks

In this part, complete the following steps:

1. Set the NVRAM parameters to defaults.

```
ok set-defaults (which you did previously)
```

or

During power on or after the ok reset, hold down the Stop (L1) and n keys simultaneously on the Sun keyboard. (There is no corresponding key to hold down to reset NVRAM to defaults from a port connection.)

2. As root, change the NVRAM settings.

```
# /usr/sbin/eeprom
```

(If no parameters are given, this shows the current settings, similar to the ok printenv command.)

The syntax to change a parameter is slightly different from the way this is done at the ok prompt; it includes the use of the = sign, an intolerance of spaces, and control characters.

3. Change the boot device.

```
# /usr/sbin/eeprom boot-device=disk1
```

- a. Perform a reset, power cycle, or boot from the ok prompt to make the change effective.
- b. Return to the ok prompt, and use the printenv command to verify that the boot-device was changed with the eeprom command.
- c. Use setenv to reset the boot-device to your system disk.

```
# setenv boot-device system_disk_name
```

---

## Exercise: Using Ultra Workstation OBP Utilities, Part 4

### Tasks

In this part of the lab exercise, you will display and capture the names of the devices in the system device tree and display their attributes. This is useful in isolating failures of Sun or third-party devices.

---

**Note** – This lab guides you through some of the device tree. Change to different devices as time allows.

---

1. Use the following commands:

```
ok cd /
ok ls
f006ce08 SUNW,UltraSPARC-IIia0,0
f005f59c pcia1f,0
f004d448 virtual-memory
foo4cd68 memory@0,10000000
.....
```

```
ok cd pci@1f,0
ok ls
f0060378 pci@1
f006fd90 pci@1,1
```

```
ok cd pci@1,1
ok ls
f007a128 ide@3
f0075828 SUNW,m64B@2
f006d820 network@1
f00610e0 ebus@1
```

```
ok cd ide@3
ok pwd
ok ls
f007d0a4 cdrom
f007c9f8 disk
ok .properties
.....
ok .version
.....
```



---

## Exercise: Using Ultra Workstation OBP Utilities, Part 5

### Tasks

In this part of the lab exercise, you will generate and test a PROM device alias. With the increased use of storage arrays and other variously addressed devices, it is important to be able to set a simple name for the device that the customer can boot from or otherwise use.

---

**Note** – If you re-create the tip hardwire session, you can cut and paste entries instead of typing them.

---

1. Use the following commands:

```
ok devalias
ok show-disks
a) /pci@1f/pci@1/scsi@1/disk
b) /pci@1f/pci@1/scsi@1,1/disk
c) /pci@1f/pci@1,1/ide@3/disk
d) /pci@1f/pci@1,1/ide@3/disk
e) /pci@1f,0/pci@1,1/ibus@1/fdthree@14,3023f0
q) NO SELECTION
```

```
Enter Selection, q to quit: d
/pci@1f,0/pci@1,1/ide@3/disk has been selected
```

```
Type ^Y ( Control-Y ) to insert it in the command line.
e.g. ok nvalias mydev ^Y
      for creating devalias mydev for
/pci@1f,0/pci@1,1/ide@3/disk has been selected
```

```
ok nvalias newdisk ^y
/pci@1f,0/pci@1,1/ide@3/disk
```

```
ok devalias
ok boot newdisk
```

---

**Note** – The boot fails here unless the alias specifies a bootable device.

---

## *Exercise: Using Ultra Workstation OBP Utilities, Part 5*

The following options can be used instead those shown on the previous page:

```
ok devalias
.....
ok cd /
ok ls (determine your location)
.....
ok cd pci@1f,0
ok ls
.....
ok cd pci@1,1
ok ls
.....
ok cd disk
ok pwd
/pci@1f,0/pci@1,1/ide@3/disk

ok nvedit
0: devalias newdisk /pci@1f,0/pci@1,1/ide@3/disk (Press
Return)
1:
(Press Control-c)

ok nvstore
ok setenv use-nvramrc? true
use-nvramrc? =          true

ok reset
```

---

## *Exercise: Working With SPARC 5 Workstation OBP Utilities*



**Exercise objective** – The objective of this exercise is to obtain information and perform diagnostic tests in the OBP environment. You will also display and change NVRAM variable settings.

### *Preparation*

---

**Note** – Due to different PROM levels and architectures, the syntax for these labs can vary slightly. Refer to the OBP reference card if necessary.

---

# Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 1

## Tasks

In this part, you will use the help and test commands to perform basic diagnostic procedures on your workstation.

1. Return the machine safely to the ok prompt.
2. Use help to list some PROM-level diagnostics.

```
ok help diag
  Category: Diag (diagnostic routines)
test  device-specifier ( -- ) run selftest method for specified device
  Examples:
    test /memory          - test memory
    test /iommu/sbus/ledma@5,8400010/le      -test net
    test.....
.....
ok setenv selftest-#megs 99 (setting up to test 99 Mbytes of memory)
ok test-memory
Testing memory \/
ok test net
```

Using AUI Ethernet Interface  
 Internal loopback test -- succeeded.  
 External loopback test -- Lost Carrier (transceiver cable problem?)  
 send failed. (This test failed because the machine is hooked only to a twisted pair net.)

Using TP Ethernet Interface  
 Internal loopback test -- succeeded.  
 External loopback test -- succeeded.  
 Run them all.

3. Run tests on devices listed in devalias. (Some devices will not have tests; that is important to know also.)

```
ok devalias
screen      /iommu@0,10000000/sbus@0,10001000/cgsix@3,0
newdisk
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/esp@5,8800000/sd@0,0
ttyb       /obio/zs@0,100000:b
ttya       /obio/zs@0,100000:a
keyboard!  /obio/zs@0,0:forcemode
.....
```

---

## *Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 1*

### *Tasks (Continued)*

ok **test keyboard**  
Keyboard Present

ok **test audio**  
CS4231 ASIC SelfTest Passed.  
L1A7192 DMA Loopback SelfTest Passed.

---

**Note** – If the ok prompt returns with no message, this means the self test found no errors.

---

4. Try other tests and commands as time allows.

## Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 2

### Tasks

In this part of the lab exercise, you will

- Gather general information about the system
- Alter NVRAM settings, display the settings, and reset to the defaults

---

**Note** – Due to the capabilities of different architectures and PROM revision level differences, some of these suggested commands may not work on your particular lab machine. For example, `switch-cpu` probably will not work on a single CPU system.

---

1. Use the following console commands and observe the output. Determine if the results are useful.

OBP Command	Results
<code>banner</code>	
<code>help diag</code> <code>help watch-tpe</code>	
<code>show boot-device</code>	
<code>show-hier</code>	
<code>show-ttys</code>	
<code>show-tapes</code>	
<code>show-nets</code>	
<code>show-disks</code>	
<code>n switch-cpu</code> where <i>n</i> is the number 0, 1, 2, and so on.	

<b>OBP Command</b>	<b>Results</b>
module-info	
devalias	
show-attrs	
show-devs	
printenv printenv diag-switch?	
setenv diag-switch? true	
show diag-switch?	
set-default diag-switch?	
show diag-switch?	
setenv fcode-debug? true	
set-defaults	
old-mode (> n)	

<b>OBP Command</b>	<b>Results</b>
Optional, experiment with the OBP reference card	

## *Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 3*

### *Tasks*

In this part, complete the following steps:

1. Set the NVRAM parameters to defaults.

```
ok set-defaults (which you did previously)
```

or

During power on or after the ok reset, hold down the Stop (L1) and n keys simultaneously on the Sun keyboard. (There is no corresponding key to hold down to reset NVRAM to defaults from a port connection.)

2. Change the NVRAM settings as root user.

```
# /usr/sbin/eeprom
```

(If no parameters are given, this shows the current settings, similar to the ok printenv command.)

The syntax to change a parameter is slightly different from the way this done at the ok prompt; it includes the use of the = sign, an intolerance of spaces, and control characters.

3. Change the boot device.

```
# /usr/sbin/eeprom boot-device=disk1
```

- a. Perform a reset, power cycle, or boot from the ok prompt to make the change effective.
- b. Return to the ok prompt, and use the printenv command to check that the boot-device was changed with the eeprom command.
- c. Use setenv to reset the boot-device to your system disk.

```
# setenv boot-device system_disk_name
```



## Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 4

### Tasks

In this part of the lab exercise, you will display and capture the names of the devices in the system device tree and display their attributes. This is useful in isolating failures of Sun or third-party devices.

---

**Note** – This lab guides you through some of the device tree. Change to different devices as time allows.

---

1. Use the following commands:

```
ok cd /
ok ls
ffd3c184 FMI,MB86904
ffd2d184 virtual-memory@0,0
ffd2d0c8 memory@0,0
ffd3c3fc obio
ffd2c128 iommu@0,10000000
.....
ok cd iommu@0,10000000
ok ls
ffd2c2c8 sbus@0,10001000
ok cd sbus@0,10001000
ok ls
ffd42504 cgsix@3,0
.....
ok cd cgsix@3,0
ok ls
ok .attributes
character-set ISO8859-1
intr 00000039 00000000
reg 00000003 00000000 01000000
dblbuf 00000000
v0,64125000,108000000,94500000
chiprev 0000000b
device_type display
model SUNW,501-2325 (look at this, the Sun part #!)
name cgsix
```

## Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 5

### Tasks

In this part of the lab exercise, you will generate and test a PROM device alias.

With the increased use of storage arrays and other variously addressed devices, it is important to be able to set a name for the device the customer can boot from or otherwise use.

---

**Note** – If you re-create the `tip` hardware session, you can cut and paste entries instead of typing them.

---

1. Use the following commands:

```
ok devalias (to show the format of devices aliases already)
tape1
/iommu/sbus/espdma@5,8400000/esp@5,8800000/st@5,0
disk3
/iommu/sbus/espdma@5,8400000/esp@5,8800000/sd@3,0
ok show-disks
a) /obio/SUNW,fdtwo@0,400000
b)
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd
q) NO SELECTION
```

```
Enter Selection, q to quit: b
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd has been selected.
```

Type `^y`( Control-Y ) to insert it in the command line.

```
e.g. ok nvalias mydev ^y
           for creating devalias mydev for
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd
ok nvalias newdisk ^y
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd@0,0
```

---

## Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 5

### Tasks (Continued)

```
ok devalias
newdisk
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd@0,0
screen
/iommu@0,10000000/sbus@0,10001000/cgsix@3,0
ttyb /obio/zs@0,100000:b
ok boot newdisk
```

---

**Note** – The boot will probably fail here unless a bootblock was placed on it. You will be setting up for alternate boots in a later module.

---

The following commands can be used instead of the commands shown on the previous page:

```
ok devalias
cd /
ok ls (just to find our way!)
ffd3c184 FMI,MB86904
ffd2d1e0 virtual-memory@0,0
ffd2d124 memory@0,0
ffd2c458 obio
ffd2c184 iommu@0,10000000
ok cd iommu@0,10000000
ok ls
ffd2c2c8 sbus@0,10001000
ok cd sbus@0,10001000
ok ls
ffd4242c cgsix@3,0
ffd423cc power-management@4,a000000
ffd41c80 SUNW,CS4231@4,c000000
ffd40024 ledma@5,8400010
ffd3ff98 SUNW,bpp@5,c800000
ffd3cea4 espdma@5,8400000
ok cd espdma@5,8400000
```

## *Exercise: Working With SPARC 5 Workstation OBP Utilities, Part 5*

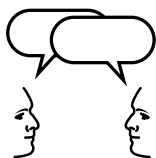
### *Tasks (Continued)*

```
ok ls
ffd3d280 esp@5,8800000
ok cd esp@5,8800000
ok ls
ffd3f854 st
ffd3f13c sd
ok cd sd
ok pwd
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd
ok nvedit
0: devalias newdisk
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/es
p@5,8800000/sd@0,1 (Carriage return)
1:
(Enter Control-c)
ok nvstore
ok setenv use-nvramrc? true
use-nvramrc? = true
ok reset
```

---

## *Exercise: Working With SPARC 5 Workstation OBP Utilities*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

## *Check Your Progress*

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Gather general information about the system
- Define the meaning of the NVRAM variables
- Display and capture the names of the devices in the system device tree, and display their attributes
- Test devices using the device path, node name, and device alias
- Generate and test a PROM device alias
- Alter and display NVRAM settings, and reset to the defaults
- Use the `eeeprom` command to examine and define NVRAM

---

## *Think Beyond*

Based on the device and OBP information in this module, how do you think you list disk device names for SPARC storage arrays? Why aren't storage array names visible from OpenBoot PROM?





## *Objectives*

Upon completion of this module, you should be able to

- Use the SunSolve database for fault analysis purposes
- Differentiate between the SunSolve CD-ROM™ and SunSolve Online™ databases
- Describe how to apply for a SunSolve Online account
- Install the SunSolve software and patches software on a server and share them correctly to the network
- Configure and use SunSolve software from an installed server or from the CD-ROM
- Display the installed patches on a Solaris system
- Display the current patch report for a given operating system
- Install and remove patches as needed on a Solaris system
- Solve a workshop exercise using SunSolve database information

## Relevance



**Discussion** – Sun users have requested a mechanism they could use to access information about systems and system problems. Sun’s solution centers needed an organized central database to report, track, and dispense information about system problems. The SunSolve system supports these needs, and is used to distribute operating system patches, important technical information, and problem workarounds for both customers and Sun support.

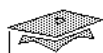
The SunSolve databases are intended to supplement, but not replace, the traditional human support interface.

- Are you a registered user of the SunSolve database?
- What kinds of information have you located within SunSolve databases?
- Have you ever configured a local SunSolve server at your site?

## References

**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- *SunSolve Online User’s Guide*
- *SunSolve User’s Guide*



*Sun Educational Services*

## Distribution

SunSolve Online is:

- Available to all Sun customers with any level of service agreement
- As a separate purchase through the 1-800-USA4SUN phone number.

To utilize SunSolve Online, establish an account using your service contract number. Sun employees use their employee ID number.

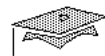
## *Distribution*

SunSolve information is available to all Sun customers (and Sun field offices or value-added reseller [VAR] accounts) with any level of service agreement, or as a separate purchase through the 1-800-USA4SUN phone number. Shipment of SunSolve CD-ROMs is one avenue of acquiring the product. The other avenue is through the web.

Additional or lost CD-ROM replacements can be arranged through your service provider or by using the 1-800-USA4SUN phone service. Updated CD-ROMs are sent out about 10 times a year and have information regarding all supported software, operating system levels, and hardware. SunSolve Online is updated nearly every business day.

In order to use SunSolve Online, you must establish an account using your service contract number (from your service provider). Sun employees use their employee ID number.

The information and search mechanisms for SunSolve Online and SunSolve CD-ROM are practically identical. The SunSolve Online information can be more up-to-date because of the logistics of shipping the CD-ROMs.



## SunSolve Online Account

To apply for a SunSolve Online account:

1. Use a browser to access one of the following web sites.
  - <http://sunsolve.sun.com>
  - <http://sunsolve1.sun.com>
  - <http://www.sun.com>
2. Click on the Register button.
3. Click on the Create new account button and answer the questions.

### *SunSolve Online Account*

A SunSolve Online account can be applied for by using a Web browser and visiting one of the following Web sites:

- <http://sunsolve.sun.com>
- <http://sunsolve1.sun.com>
- <http://www.sun.com>

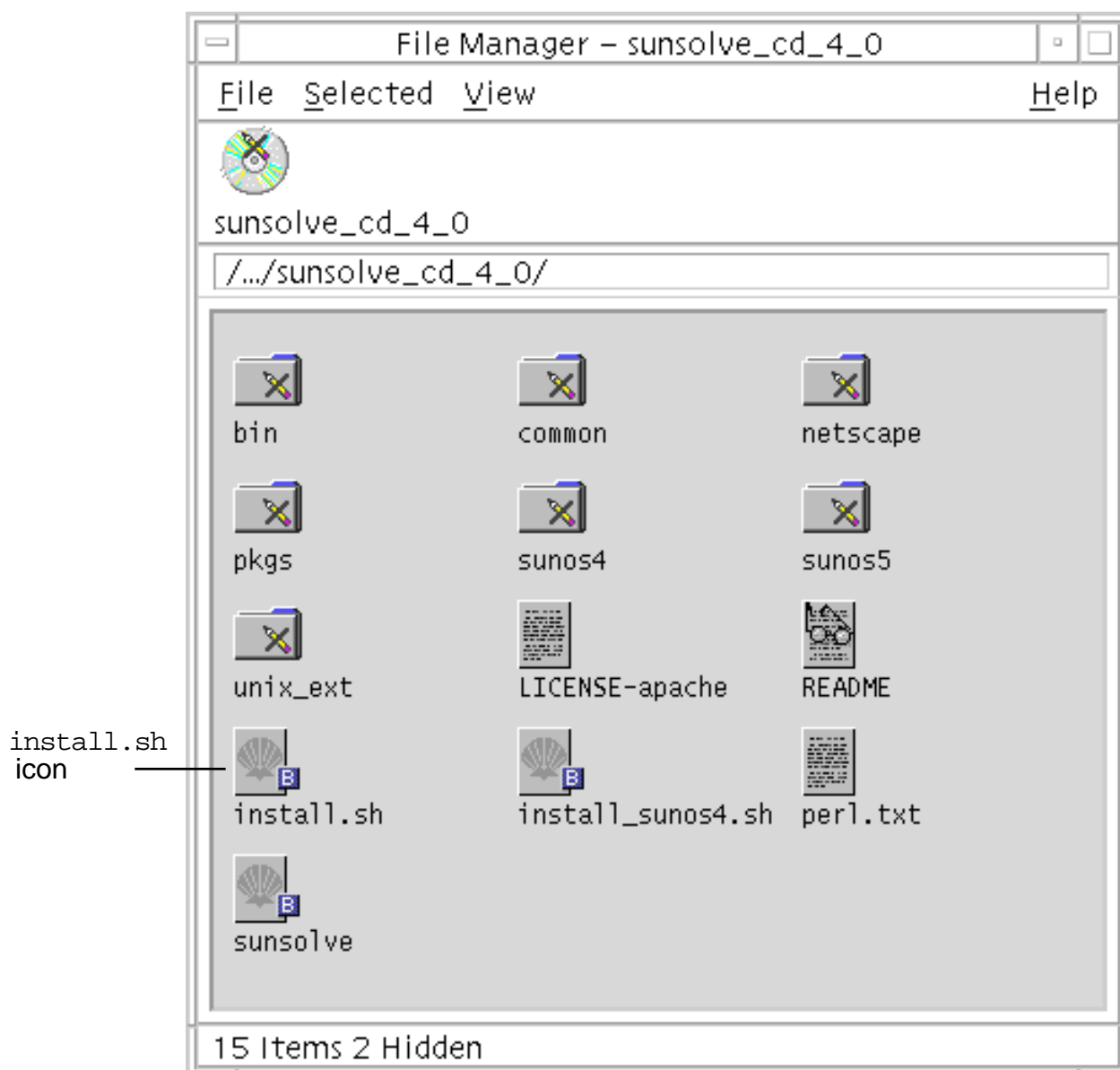
Follow these steps to apply for a SunSolve Online account:

1. Using a browser, access one of the above uniform resource locators (URLs).
2. Click on the Register button.
3. Click on the Create new account button and answer the questions. (You must have a Sun Service Spectrum Account number to register for a SunSolve Online account.) There is little or no wait in receiving an account once you submit the form.

For more details on a SunSolve Online account, refer to the *SunSolve Online Reference Guide*.

## Installing SunSolve Software

After inserting the SunSolve CD-ROM into the drive, you can use File Manager to install the software. Click on the `install.sh` icon and then provide answers to the prompts. A sample install session is shown on the following page. (The file names and paths on the CD-ROM may change with different releases of SunSolve software.)



**Figure 5-1** File Manager Display of CD-ROM Files

## *Installing SunSolve Software*

The `install.sh` script can also be run at the shell prompt. An example of this type of installation session follows.

```
# cd /cdrom/cdrom0
# ./install.sh
```

```
Script started on Thu Nov 05 10:44:51 1998
```

```
The following packages are available:
```

```
 1 SUNWss      SunSolve CD
                   (sparc) 4.0.0,REV=Final Release
```

```
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
```

```
Processing package instance <SUNWss> from
</cdrom/sunsolve_cd_4_0/pkg/SUNWss.sparc>
```

```
SunSolve CD
```

```
(sparc) 4.0.0,REV=Final Release
```

```
Copyright 1998 Sun Microsystems, Inc. All rights reserved.
```

```
Enter path to package base directory [?,q] /opt/SUNWss
```

```
The selected base directory </opt/SUNWss> must exist before
installation is attempted.
```

```
Do you want this directory created now [y,n,?,q] y
```

```
Using </opt/SUNWss> as the package base directory.
```

## *Installing SunSolve Software*

```
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

This package contains scripts which will be executed with super-user permission during the process of installing this package.

Do you want to continue with the installation of <SUNWss> [y,n,?] y

Installing SunSolve CD as <SUNWss>

```
## Executing preinstall script.
## Installing part 1 of 1.
```

```
/etc/rc2.d/S89httpd
/opt/SUNWss/sso/etc/K89httpd <symbolic link>
[ verifying class <none> ]
/etc/init.d/httpd_init <linked pathname>
/etc/rc1.d/K89httpd <linked pathname>
## Executing postinstall script.
installing bin...
```

...

< The script continues to echo the name of the package components as it installs, completing with the following message.>

Installation of SUNWss successful.



## Configuring a SunSolve Server

1. Insert the SunSolve CD-ROM on the server.
2. If needed, perform the installation procedure described previously, or run the software from the CD-ROM.
3. Share the software and, if needed, the CD-ROM.

```
# share -o ro SunSolve_install_dir
# share -o ro /cdrom/crom0
```

4. Add the following line to `/etc/dfs/dfstab`:

```
# share -o ro SunSolve_install_dir
# share -o ro /cdrom/cdrom0
```

## Configuring a SunSolve Server

The following procedure shows the steps needed to share the SunSolve software, and if needed, the CD-ROM from a server. The version number of SunSolve software under the `/cdrom` directory changes with progressive releases.

1. Insert the SunSolve CD-ROM into the CD-ROM drive on the server and verify that the `vold` daemon has mounted the software using the `mount` command.
2. If needed, perform the installation procedure described previously, or run the software from the CD-ROM.
3. Share the software and, if needed, the mounted CD-ROM.

```
# share -o ro SunSolve_install_dir
# share -o ro /cdrom/cdrom0
```

4. Add the following line to the `/etc/dfs/dfstab` file:

```
# share -o ro SunSolve_install_dir
# share -o ro /cdrom/cdrom0
```



---

## Configuring a SunSolve Server

5. Start the NFS server.

```
# /etc/init.d/nfs.server start
```

6. Check to see if the share command was successful.

```
# dfshares -F nfs server_name
```

7. On the clients, remotely mount the software or provide clients with the URL `http://servername`.

```
# mount server_name:/cdrom/cdrom0 /cdrom
```



## Running SunSolve Software

There are three ways you can run SunSolve software:

- From the SunSolve CD-ROM
- From a web site
- From an installed server at your site

### *Running SunSolve Software*

There are three ways you can run SunSolve software:

- From the SunSolve CD-ROM. Click on the sunsolve icon in the top-level CD-ROM File Manager window.
- From one of the web sites listed on page 5-4
- From an installed server at your site

Since the release of SunSolve 4.0, SunSolve automatically loads a web server, and can be accessed through Netscape Navigator™, as well as other browsers, using the URL

`http://server_name:port_number.`

The port number need not be specified if the default port is used. The installation, by default, tries to use port 80. If that port is already in use, port 3000 or the next available port is used.



## *The SunSolve Home Page Selections*

Once SunSolve software is installed, accessing the server site location provides the SunSolve home page. The main selections are described here, and are available through any of the execution methods mentioned previously.

- *Home Page* – Returns to the SunSolve home page.
- *Getting Started* – Provides a description of each main menu option with additional selections that provide more in-depth information for each.
- *About the CD* – Provides operating system and hardware requirements for running SunSolve, as well as Netscape and SunSolve version information.
- *Browse Collections* – Provides a browser for scanning any of SunSolve databases.
- *Patch Reports* – Provides options for finding a particular patch and for locating the current patch report for any release of Solaris.



## The SunSolve Online Home Page Selections

- Power Search
- Patch Diag Tool
- Crash Dump Analysis
- SunCourier

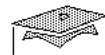
### *The SunSolve Home Page Selections*

- *Power Search* – Provides a menu of selectable database collections to search; these include
  - ▼ All SunSolve collections
  - ▼ Bug reports
  - ▼ Frequently asked questions (FAQs)
  - ▼ Patch descriptions
  - ▼ White papers/tech bulletins
  - ▼ Early notifier
  - ▼ Info docs
  - ▼ Symptoms and resolutions

---

## *The SunSolve Home Page Selections*

- *Patch Diag Tool* – Determines the patch level of your system compared to Sun’s recommended patch lists, both for the operating system and for particular patches. Also includes reference information and Patch Diag Tool installation instructions.
- *Crash Dump Analysis* – Displays how to load and run the ISCDA (Initial System Crash Dump Analysis) script. The text of the ISCDA page is included in the “Useful SunSolve Documents” section at the end of this module.
- *SunCourier* – Submits a service request to a Sun solution center. (This requires that sendmail is running.)



## Power Search Tool Overview

- To move from the initial Power Search screen to the display that allows search criteria to be specified, click on Continue.
- Once you have set the criteria, click on Search.
- Having located a document(s), enter a string to refine the search within a particular document.

### *Power Search Tool Overview*

To perform a refined and efficient search for information, click on the Power Search box on the SunSolve home page. This displays the screen shown in Figure 5-2.

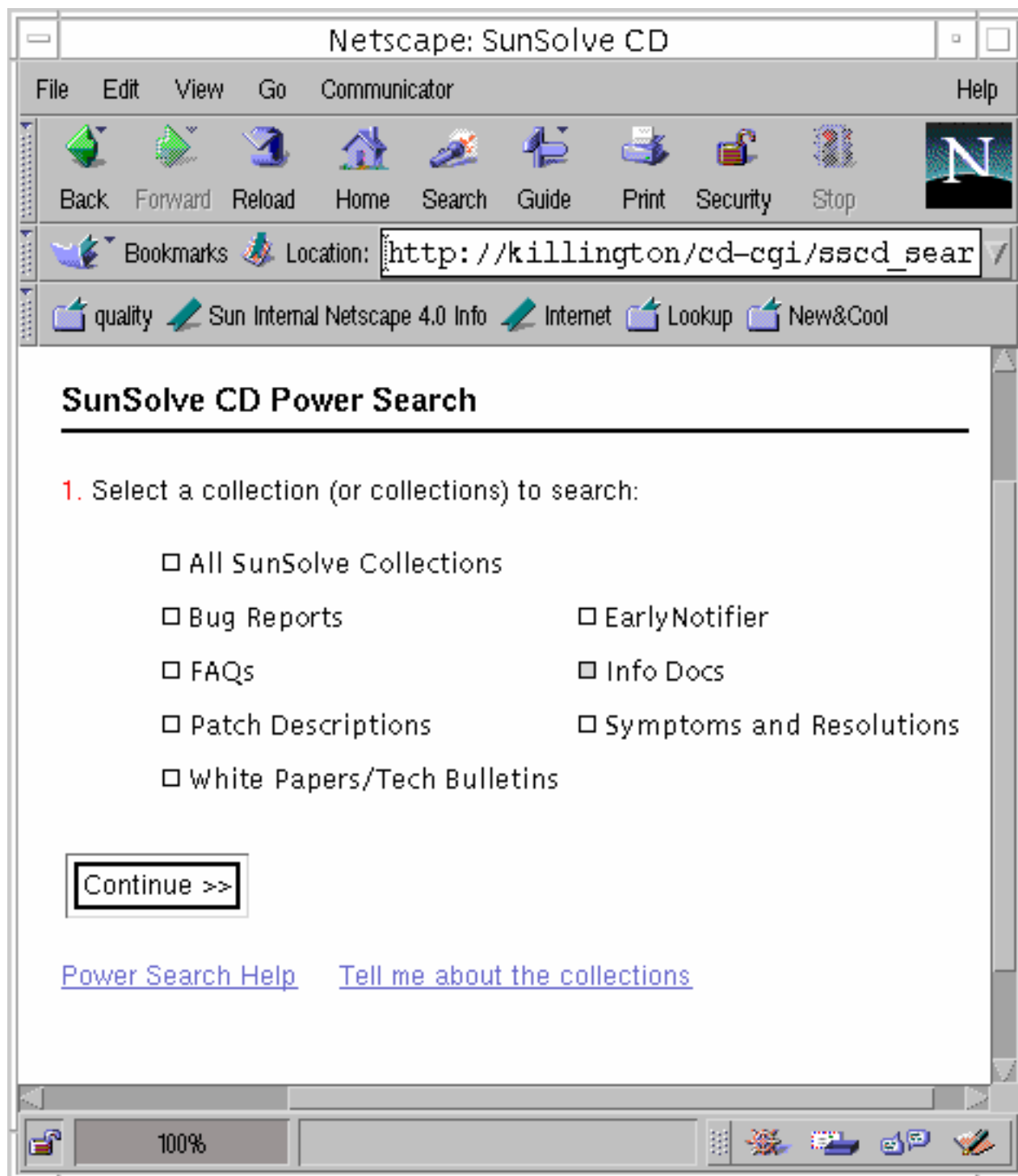
Two displays are available beneath this screen. One provides options that enable the search criteria to be refined. The other, once a document is located, enables a key search within the document.

The following describes how to progress from one display to the next:

- To move from the initial Power Search screen to the display that enables search criteria to be specified, click on the Continue button.
- Once you have set the criteria, click on Search.
- Having located a document(s), enter a string to refine the search within a particular document.

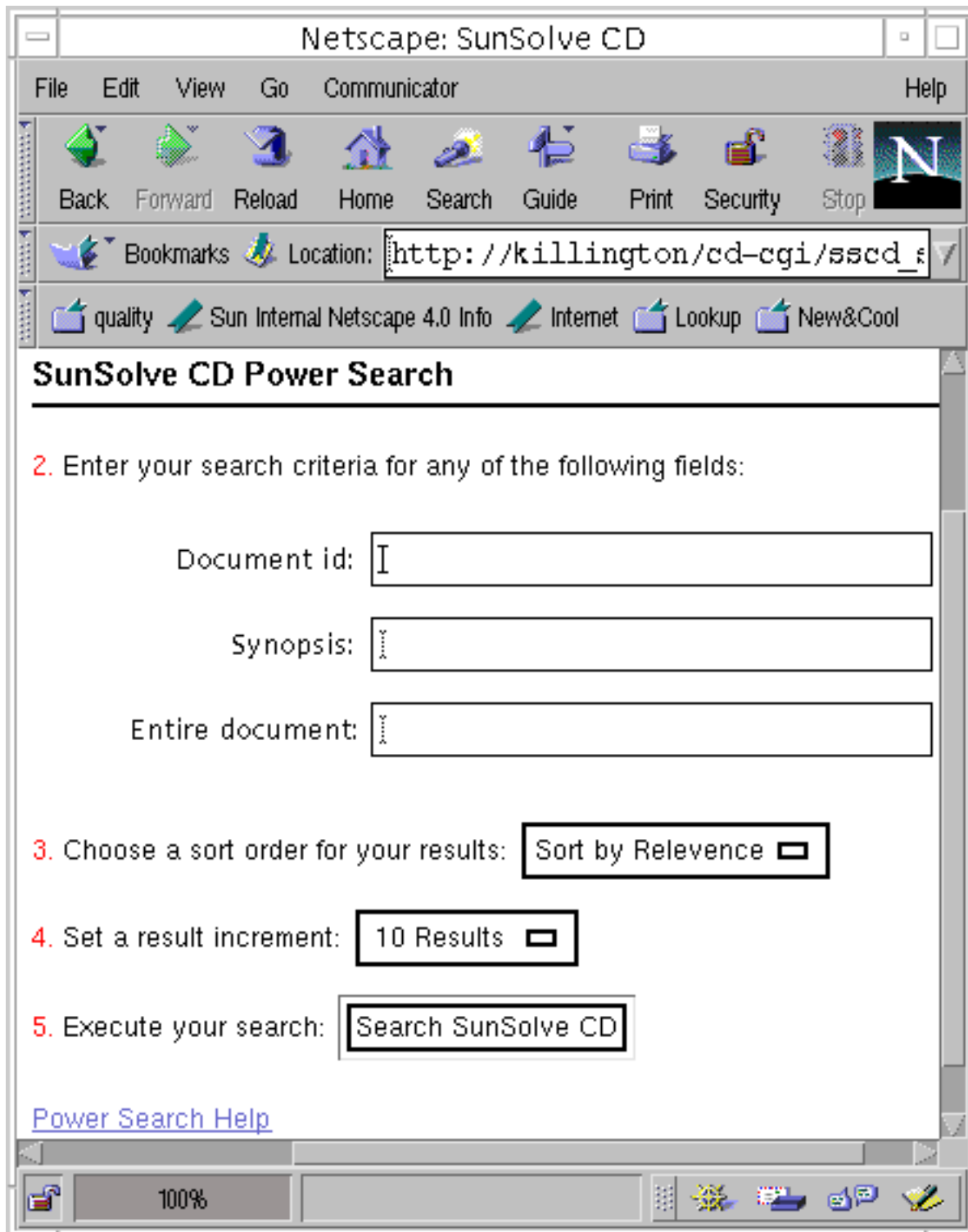
The next several pages show the main screens that display within Power Search Tool. Online help is available throughout the SunSolve screens.

## Power Search Collections



**Figure 5-2** SunSolve CD Power Search Collections

## Defining Search Criteria



**Figure 5-3** Defining Search Criteria



## Power Search Results

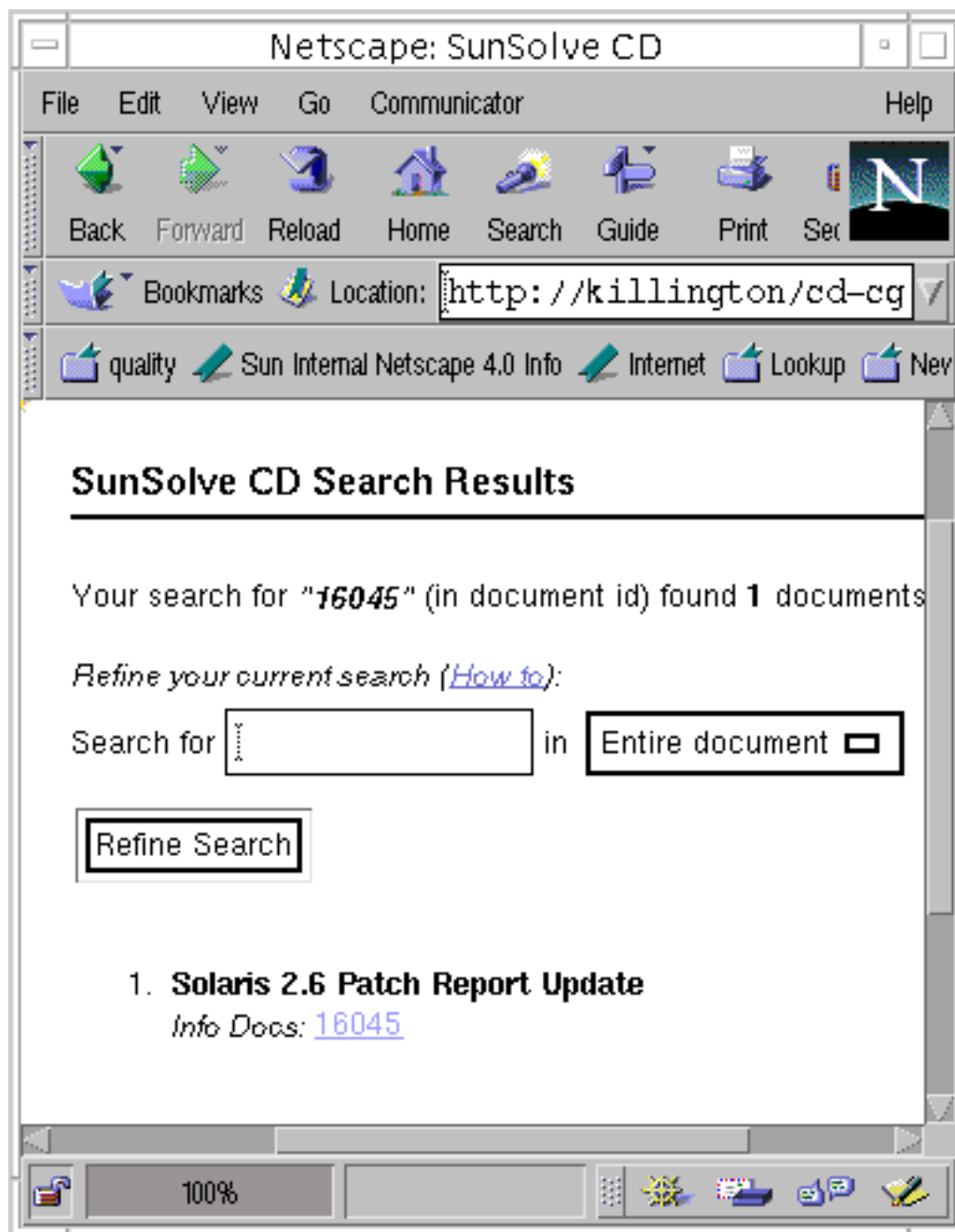


Figure 5-4 Power Search Results



## Boolean Search Syntax

- AND – This is the default operator when two terms are entered on a query line.
- OR – Curly braces ( { } ) are used to designate an OR operation.
- NOT – The exclamation point ( ! ) specifies the NOT operation.

### *Boolean Search Syntax*

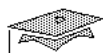
The use of Boolean search syntax can expedite searches through selected SunSolve databases. The following items summarize how to specify the major Boolean operators with text:

- AND – This is the default operator when two terms are entered on a query line. For example, the term *crash analysis* searches for documents that include both *crash* and *analysis*. Brackets can also be used for multiple terms; for example, the term [*crash analysis*] succeeds when every term in the brackets is found.
- OR – Curly braces are used to designate an OR operation. For example, the search string {*crash analysis panic*} returns documents that contain one or more of the terms within the braces.
- NOT – The exclamation point specifies the NOT operation. For example, the term [*crash ! watchdog*] returns documents that contain the word *crash*, but not the word *watchdog*.

---

**Note** – For more information, see “Searching Syntax” in Power Search Online Help.

---



*Sun Educational Services*

## The Current Patch Report

Click on the SunSolve Online home page. This provides screen areas with two main selections:

- Patch Report
- Enter a patch ID or keyword

### *The Current Patch Report*

To obtain a current patch report, click on Patch Report on the SunSolve home page. This provides screen areas with two main selections:

- Patch report
- Enter a patch ID or keyword

The Patch Report selection provides a comprehensive list of reports from Solaris 1.x through the current Solaris release. Click on the release you are interested in.

An excerpt of the 2.6 Patch Report is shown on the next page. Other information, including history, patch dependencies, and update information, is also available through this report mechanism.

You can also find the files which constitute the recommended patch reports on the SunSolve Patches CD-ROM.

---

**Note** – Searches are not case sensitive.

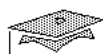
---

## *The Solaris 2.6 Patch Report*

Solaris 2.6 Recommended Patches:(excerpt)

-----

106125-05	SunOS 5.6:	Patch for patchadd and patchrm
105401-15	SunOS 5.6:	libnsl and NIS+ commands patch
105181-08	SunOS 5.6:	kernel update patch
105562-03	SunOS 5.6:	chkey and keylogin patch
105210-13	SunOS 5.6:	libc & watchmalloc patch
105216-03	SunOS 5.6:	/usr/sbin/rpcbind patch
105356-06	SunOS 5.6:	/kernel/drv/ssd patch
105357-02	SunOS 5.6:	/kernel/drv/ses patch
105375-09	SunOS 5.6:	sf & social driver patch
105379-05	SunOS 5.6:	/kernel/misc/nfssrv patch
105621-08	SunOS 5.6:	libbsm patch
105393-06	SunOS 5.6:	at & cron utility patch
105395-03	SunOS 5.6:	/usr/lib/sendmail patch
105407-01	SunOS 5.6:	/usr/bin/volrmmount patch
105552-02	SunOS 5.6:	/usr/sbin/rpc.nisd_resolv patch
105615-04	SunOS 5.6:	/usr/lib/nfs/mountd patch
105665-03	SunOS 5.6:	/usr/bin/login patch
105667-01	SunOS 5.6:	/usr/bin/rdist patch
105786-05	SunOS 5.6:	/kernel/drv/ip patch
105741-05	SunOS 5.6:	/kernel/drv/ecpp patch
105720-03	SunOS 5.6:	/kernel/fs/nfs patch
106049-01	SunOS 5.6:	/usr/sbin/in.telnetd patch
106235-02	SunOS 5.6:	lp patch
106257-04	SunOS 5.6:	/usr/lib/libpam.so.1 patch
106271-04	SunOS 5.6:	/usr/lib/security/pam_unix.so.1 patch
105755-06	SunOS 5.6:	libresolv, in.named, named-xfer, nslookup, nstest
105926-01	SunOS 5.6:	/usr/sbin/static/tar patch
106301-01	SunOS 5.6:	/usr/sbin/in.ftpd patch
106448-01	SunOS 5.6:	/usr/sbin/ping patch
105464-01	SunOS 5.6:	multiple xterm fixes
105490-05	SunOS 5.6:	linker patch
105580-07	SunOS 5.6:	/kernel/drv/glm patch
105797-03	SunOS 5.6:	/kernel/drv/sd patch
105600-06	SunOS 5.6:	/kernel/drv/isp patch
106226-01	SunOS 5.6:	/usr/sbin/format patch
105642-05	SunOS 5.6:	prtdiag patch
....		



*Sun Educational Services*

## Patch Diag Tool

- Is one of the newer options within SunSolve Online
- Provides a report indicating the status of installed patches on a system
- Provides
  - A sample report
  - Installation instructions
  - A user's guide

### *Patch Diag Tool*

One of the newer options within SunSolve software is Patch Diag Tool, which provides a report indicating the status of installed patches on a system, especially in comparison to Sun's Recommended Patch list.

Click on the Patch Diag Tool Selection on the SunSolve home page for access to a sample report, installation instructions, and a user's guide. If you have an installed server, the `patchdiag` command can be executed from the command line.

## Patch Diag Tool

### Patch Diag Report

The following output is a sample report, produced on a system with all recommended patches installed:

```
# cd /SunSolve_install_dir/sso/PatchDiag
# ./patchdiag.sparc
```

```
=====
System Name: killington   SunOS Vers: 5.7           Arch: sparc
Cross Reference File Date: 25/Sep/98
```

```
PatchDiag Version: 1.0.2
```

```
=====
Report Note:
```

Recommended patches are considered the most important and highly recommended patches that avoid the most critical system, user, or security related bugs which have been reported and fixed to date. A patch not listed on the recommended list does not imply that it should not be used if needed. Some patches listed in this report may have certain platform specific or application specific dependencies and thus may not be applicable to your system. It is important to carefully review the README file of each patch to fully determine the applicability of any patch with your system.

```
=====
INSTALLED PATCHES
Patch  Installed Latest  Synopsis
  ID   Revision  Revision
-----
=====
```

# Patch Diag Tool

## Patch Diag Report (Continued)

### UNINSTALLED RECOMMENDED PATCHES

Patch ID	Ins Rev	Lat Rev	Age	Require ID	Incomp ID	Synopsis
----------	---------	---------	-----	------------	-----------	----------

-----  
 All Recommended patches installed!  
 =====

### UNINSTALLED SECURITY PATCHES

NOTE: This list includes the Security patches that are also Recommended

Patch ID	Ins Rev	Lat Rev	Age	Require ID	Incomp ID	Synopsis
----------	---------	---------	-----	------------	-----------	----------

-----  
 All security patches installed!  
 =====

### UNINSTALLED Y2K PATCHES

NOTE: This list includes the Y2K patches that are also Recommended

Patch ID	Ins Rev	Lat Rev	Age	Require ID	Incomp ID	Synopsis
----------	---------	---------	-----	------------	-----------	----------

-----  
 All Y2K patches installed!

## Installing Patches

The SunSolve CD-ROM software usually includes two CD-ROMs which contain patches. Typically, CD-ROM 1 contains patches for the operating system, languages, and patch clusters. CD-ROM 2 contains patches for unbundled products.

To install a patch, place the CD-ROM in the drive, and run the `patchinstall` utility, specifying the number of the patch to load as an argument. This can be done from the shell prompt, or from a File Manager window by clicking on the `patchinstall` icon.

An example of a `patchinstall` session follows.

---

**Note** – Unless the installation procedure specifies otherwise, reboot your system after installing a patch.

---

```
# cd /cdrom/cdrom0
# ./patchinstall
=====
patchinstall - install a patch

    Copyright (c) 1993 Sun Microsystems, Inc. All Rights Reserved.
    Printed in the United States of America.
    2550 Garcia Avenue, Mountain View, California, 94043-1100 U.S.A.

Patches are distributed to SunService Contract Customers ONLY.
Redistribution to unauthorized parties is prohibited by the SunService
Contract. Installation of all patches is not suggested as some patches
may conflict with one another. Please make sure that the patch you are
installing is necessary before actually installing the patch.
=====

During the installation, default answers will be provided inside brackets
`[]' Pressing the <Return> key will select the default provided.

Press <Ctrl-C> at any time to stop the installation.
```



## *Installing Patches*

Continue with patch installation? [Y]

Installation of patches will use several temporary files, and may need several megabytes of space. Please enter the name of a directory that can be used for these temporary files (the directory must exist before running the installation!)

Where should I store temporary files? [/tmp]

The installation for Solaris 2.x patches provides the option of saving the original versions of the software being patched. Unfortunately, this occasionally will cause your /var/sadm/patch directory to grow too large. By default, this installation will NOT save your original versions of software.

Would you like to save the original versions of the software? [no]

Next you will be prompted for the patches you wish to install. If you would like to install the suggested patches, enter "suggested". Otherwise enter the patch id.

To see a list of the patches you have entered, type a '?' <Return>

To start the installation, press <Return> when prompted for a patch id.

Patch to install (patchid, suggested, ?): suggested

.....  
.....  
Installation complete!

#



## Removing a Patch

1. Display a list of installed patches on your system.  
`# showrev -p`
2. Find the installed location of the patch.  
`# find / -name 102044-01 -print`
3. Change directory to the location of the patch.  
`# cd /var/sadm/patch/102044-01`
4. Run the script to remove the patch, and reboot the system.  
`# ./backoutpatch .`

## Removing a Patch

To remove a specific patch, perform the following steps:

1. Display a list of installed patches on your system to find the exact name and revision level of the patch.  
`# showrev -p`
2. Find the installed location of the patch. (They are usually installed in the `/var/sadm/patch` directory.)  
`# find / -name 102044-01 -print`  
*(output omitted)*
3. Change directory to the location of the patch.  
`# cd /var/sadm/patch/102044-01`
4. Run the script to remove the patch.  
`# ./backoutpatch .`
5. Reboot the system.  
`# reboot`

---

## *Useful SunSolve Documents*

The following list shows some useful documents and categories you can locate in the SunSolve databases. The initial system crash dump analysis (ISCDA) document is discussed in Module 7.

- *Sun High Performance Computing White Paper* (1367)
- *Ultra 450 Architecture White Paper* (1396)
- *Sun Enterprise Cluster Architecture White Paper* (1380)
- *Troubleshooting Guide for Watchdog Resets* (1360)
- *Security in Practice* (1390)
- *ISCDA* (13089)
- *Information to Send to Sun for Analysis* (11835)
- Current patch reports – All releases
- Specific architectures, such as sun-4u or sun-4m
- Common Solaris error messages in alphabetical order

There are many other useful documents available depending on the system software or hardware and architecture that you are interested in. Spend some time browsing through the SunSolve database collections.

## *Other Useful SunSolve Documents*

Use the following space to note the documentation numbers of titles of other SunSolve documents you have found useful, or that you discover in the lab and workshop exercises.

- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

## *Exercise: Using SunSolve Fault Analysis*



**Exercise objective** – The objective of this exercise is to load, locate, and configure the SunSolve software, and then use the software as a fault analysis resource.

### *Preparation*

Consult your instructor as to server configuration in the classroom. If needed, configure a server in your group, or remotely mount the software or SunSolve CD-ROM from a designated classroom server.

## Exercise: Using SunSolve in Fault Analysis

### Tasks

Complete or answer the following:

1. Answer the following questions:

- ▼ Describe how SunSolve databases helps resolve system faults.

---

---

---

- ▼ Differentiate the CD-ROM and online SunSolve databases.

---

---

---

- ▼ Describe how to apply for an online SunSolve account.

---

---

---

2. Start SunSolve software and locate the documents listed by number in the "SunSolve Useful Documents" section of this module. Add any additional notes regarding these documents here.

---

---

## *Exercise: Using SunSolve in Fault Analysis*

### *Tasks (Continued)*

3. Display the installed patches on your system.

- 
4. Display the current patch report for a given OS release.

- 
5. In your workshop group, solve Workshop #41 in Appendix D. The solution for this workshop is found by locating information within SunSolve databases.

## *Exercise: Using SunSolve in Fault Analysis*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications



---

## Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Use the SunSolve database for fault analysis purposes
- Differentiate between the SunSolve CD-ROM and SunSolve Online databases
- Describe how to apply for a SunSolve Online account
- Install the SunSolve software and patches software on a server and share them correctly to the network
- Configure and use SunSolve software from an installed server or from the CD-ROM
- Display the installed patches on a Solaris system
- Display the current patch report for a given operating system
- Install and remove patches as needed on a Solaris system
- Solve a workshop exercise using SunSolve databases

## *Think Beyond*

Now that you have had some exposure to using SunSolve software, are there any issues at your site that should be analyzed by using the SunSolve database? Are there Solaris topics for which there would be no reference in SunSolve databases?

### *Objectives*

Upon completion of this module, you should be able to

- Install the SunVTS package on a system
- Select, set up, and run SunVTS diagnostic tests
- Run SunVTS over a network
- Run SunVTS in TTY mode without a frame buffer
- Analyze SunVTS test results

## Relevance



**Discussion** – This module presents the tools available in the SunVTS package for the analysis and diagnosis of hardware problems. The skills developed in this module expand the scope of fault analysis in the class so as to encompass both hardware and software problems.

Before beginning Module 6, consider the following questions related to your own experience with the Solaris operating system:

- Have you used the SunVTS software or its earlier version in the package SUNWdiag?
- What kind of hardware problems, if any, have you encountered with your Solaris systems?
- Have you experienced any hardware problems that were not detected by POST diagnostics at boot time?

## References



**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- *SunVTS 3.0 User's Guide*, part number 805-4442
- *SunVTS 3.0 Quick Reference Card*, part number 805-4444
- *SunVTS 3.0 Test Reference Manual*, part number 805-4443

---

**Note** – The documents are available on the *Solaris 7 Hardware AnswerBook* set, which is provided on the Solaris 7 Supplement CD-ROM, or at the uniform resource locator (URL)  
<http://docs.sun.com>.

---



## Introduction

- SunVTS software
  - Is an on-line validation test suite
  - Verifies functionality of Sun hardware devices
  - Is a successor to SunDiag™ diagnostics
  - Runs on Solaris 2.5 and later releases

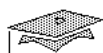
## *Introduction*

### *SunVTS Software Overview*

SunVTS is Sun's on-line validation test suite. Functionality of most Sun hardware devices can be verified. The SunVTS tests can be used to stress certain areas of the system as needed for diagnostic and troubleshooting purposes.

The SunVTS diagnostic software is the successor to SunDiag™ diagnostics, which is shipped with the Solaris 2.4 operating system or earlier releases. SunVTS runs on the Solaris 2.5 operating system and later releases.

Like its SunDiag predecessor, SunVTS software can run concurrently with customer applications and the Solaris operating system.



## Hardware and Software Requirements

- The Solaris 7 operating system
- The SunVTS 3.0 package
- Operating system kernel configured to support all peripherals that are to be tested
- Superuser access for installation and startup of SunVTS software

### *Introduction*

#### *Hardware and Software Requirements*

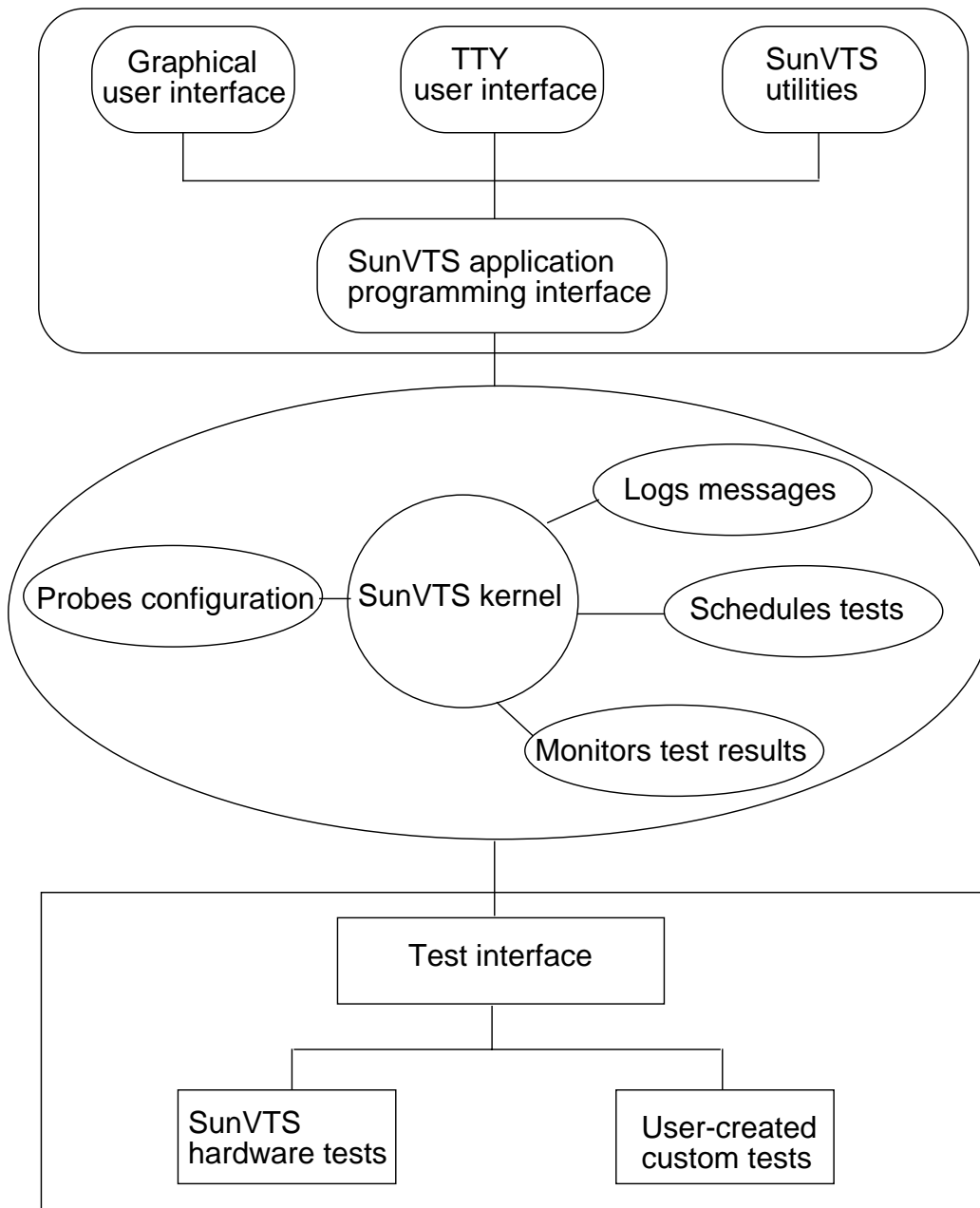
The requirements to run SunVTS Version 3.0 software successfully in the common desktop environment (CDE) environment are

- The Solaris 7 operating system
- The SunVTS 3.0 package
- Operating system kernel configured so as to support all peripherals that are to be tested
- Superuser access for both installation and startup of SunVTS software

## The SunVTS Architecture


The SunVTS architecture is divided into the

- User interfaces
- SunVTS kernel
- Hardware tests



**Figure 6-1** SunVTS Architecture





*Sun Educational Services*

---

## Interfaces

- User
- Kernel
- Hardware tests

## *Interfaces*

### *User*

SunVTS has graphical user interface for the Common Desktop Environment, OpenWindows™ (OpenLook), and a TTY version of the interface for a terminal sessions.

### *Kernel*

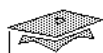
The kernel runs as a background process, a daemon. Upon startup of the SunVTS software, the SunVTS kernel probes the system kernel for installed hardware devices. Those devices are displayed on the SunVTS user interface.

Both the SunVTS kernel and the user interface must be started before testing can begin.

## *Interfaces*

### *Hardware Tests*

For each supported hardware device, a corresponding hardware test can validate its operation. Each test is a separate process from the SunVTS kernel process.



*Sun Educational Services*

## Installing SunVTS Software

The following packages comprise the SunVTS software:

- `SUNWvts`
- `SUNWvtsx`
- `SUNWodu`
- `SUNWvtsmn`

The SunVTS software can be installed using either

- The `pkgadd` command
- The installation script

### *Installing SunVTS Software*

The following packages represent a full installation of the SunVTS package, and require approximately 29 Mbytes of disk space in `/opt`:

- `SUNWvts` – The SunVTS kernel, user interfaces, and tests
- `SUNWvtsx` – The extension to support 64-bit architectures
- `SUNWodu` – On-line diagnostic utilities
- `SUNWvtsmn` – The SunVTS man page references

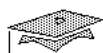
## *Installing SunVTS Software*

The SunVTS software can be installed using either of the following methods:

- Running pkgadd – Use cd and pkgadd commands:  

```
# cd /cdrom/cdrom0/Product  
# pkgadd -d . SUNWvts SUNWvtsx SUNWodu SUNWvtsmn
```
- Running the installer script – Run the installation script provided on the CD-ROM in /cdrom/cdrom0/installer. It can be run from the shell or through a File Manager window.

The procedure checks available disk space before proceeding with the installation. Either method allows for a custom installation of products available on the Solaris 7 Supplement CD-ROM, which also includes the *Hardware AnswerBook*, SUNWabh.



Sun Educational Services

## Starting the SunVTS Software

The SunVTS program is started using

- `sunvts`
- `sunvts -l`
- `sunvts -t`
- `sunvts -h host_name`

### *Starting the SunVTS Software*

- ▼ The SunVTS program is run when the superuser types one of the following commands. The `/opt/SUNWvts/bin` directory needs to be defined as part of the `PATH` variable.
- `sunvts` – Runs the SunVTS kernel and default graphical interface (CDE) on the local machine
- `sunvts -l` – Runs the SunVTS kernel and OpenLook graphical interface on the local machine
- `sunvts -t` – Runs the SunVTS kernel in TTY mode, `vtstty`
- `sunvts -h host_name` – Runs the graphical interface on the local machine while connecting and testing a remote machine

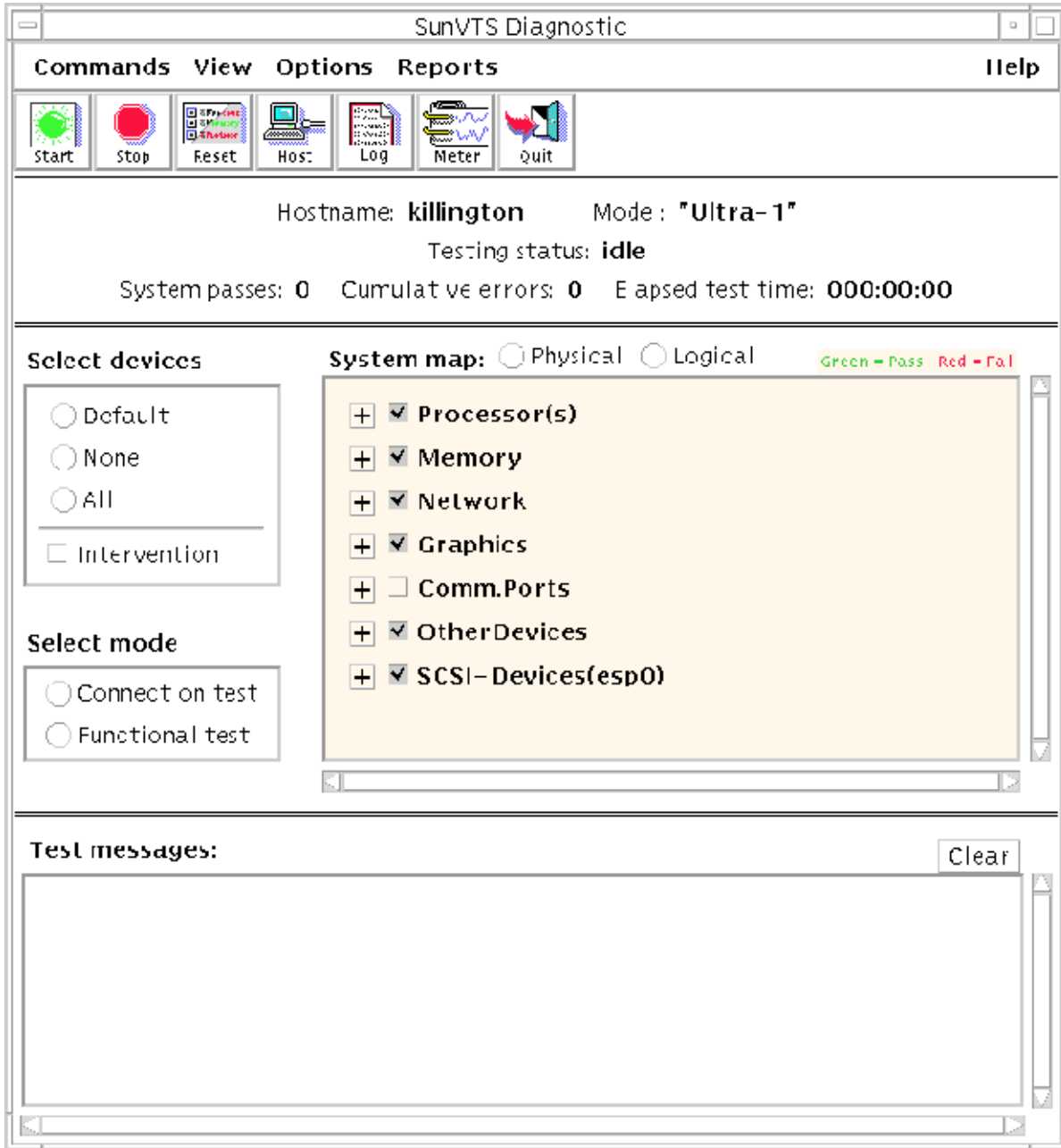
---

**Note** – The `SUNvts` package and, if needed, the `SUNvtsx` package must be installed on both local and remote machines to perform remote diagnostics.

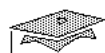
---

## The SunVTS Graphical Interface

The initial SunVTS graphic menu is shown in Figure 6-2.



**Figure 6-2** SunVTS Graphical Interface



Sun Educational Services

## The Sun VTS Window Panels

- System Status Panel
- The System Map
- Select Devices
- Select Mode
- Test Messages

### *The Sun VTS Window Panels*

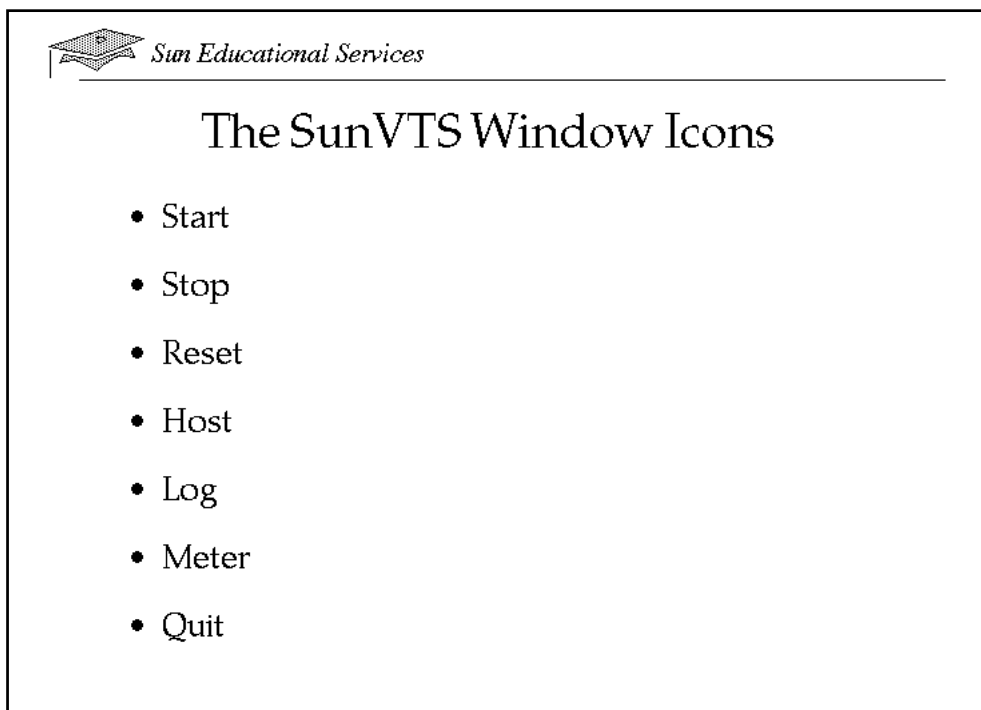
The five major panels of the SunVTS window are:

- *System Status Panel* – Test status, host name, model type, number of passes and errors, and elapsed time are displayed in the upper area of the SunVTS menu.
- *The System Map* – This area of the initial menu displays a logical device view consisting of a selectable list of devices to test by default. Each device test can be turned on or off by clicking on the check box. Particular devices, such as CPUs, network interfaces, or disks, can be selected by clicking on the plus sign box.
- *Select Devices* – This area of the SunVTS menu enables you to quickly select the devices to test, including a default set (shown in Figure 6-2).

## *The SunVTS Window Panels*

- *Select mode* – The following tests can only be run in Connection Test mode. All other tests support Functional Test mode only.
  - ▼ `cdtest` – Small computer system interface (SCSI) CD-ROM test
  - ▼ `disktest` – SCSI disk test
  - ▼ `fputest` – Floating point unit test
  - ▼ `mpptest` – Multiprocessor test
  - ▼ `pmemtest` – Physical memory test
  - ▼ `sptest` – Onboard serial port test
  - ▼ `tapetest` – SCSI tape test
  - ▼ `audiotest` – Audio device test
  - ▼ `nettest` – Network test
  - ▼ `plntest` – SPARCstorage™ array controller test
  - ▼ `ecpptest` – ECP1284 parallel port printer test
  - ▼ `bpptest` – Bidirectional parallel port printer test
  - ▼ `rsctest` – Enterprise 250 remote system control test
- *Test Messages* – This area displays any information or error messages that are issued during test executions.

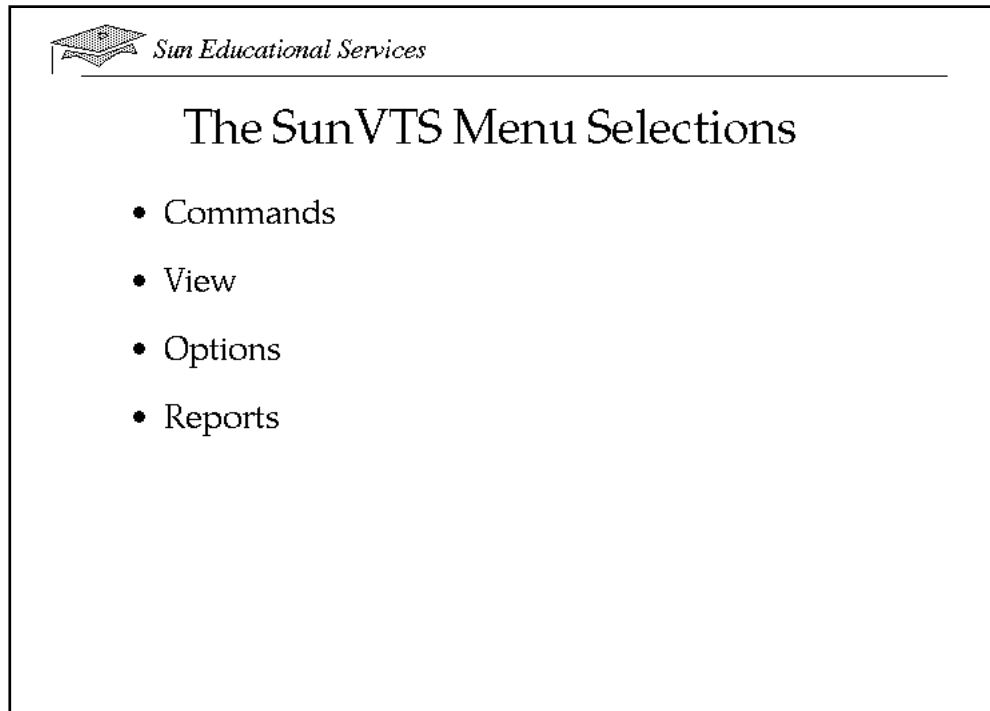




### *The SunVTS Window Icons*

Seven icons are provided at the top of the SunVTS menu. These are:

- Start – Begins the test, according to the selections made in the System map, Select devices, and Select mode areas. Progressive updates are displayed in the Information Panel during testing.
- Stop – Stops current testing, without exiting SunVTS.
- Reset – Sets the System map area to previous state.
- Host – Provides a submenu in which a remote host name to connect to can be entered. This host must be reachable, with SunVTS installed.
- Log – Displays the log file, and provides a menu to select the amount of information to log, including errors, information, and UNIX messages (`/var/adm/messages`).
- Meter – Invokes the performance monitor utility, which graphically displays system resource activity during testing.
- Quit – Exits the SunVTS program.



## *The SunVTS Menu Selections*

The top horizontal bar of the SunVTS menu has four selections with lists of associated submenus.

- Commands – This menu provides the following:
  - ▼ Start testing – Begin testing
  - ▼ Stop testing – Halt testing
  - ▼ Connect to host – Specify host name to connect to
  - ▼ Trace test – Select a test to trace, and a location for the output
  - ▼ Reprobe system – Probe the hardware
  - ▼ Quit VTS – Exit SunVTS

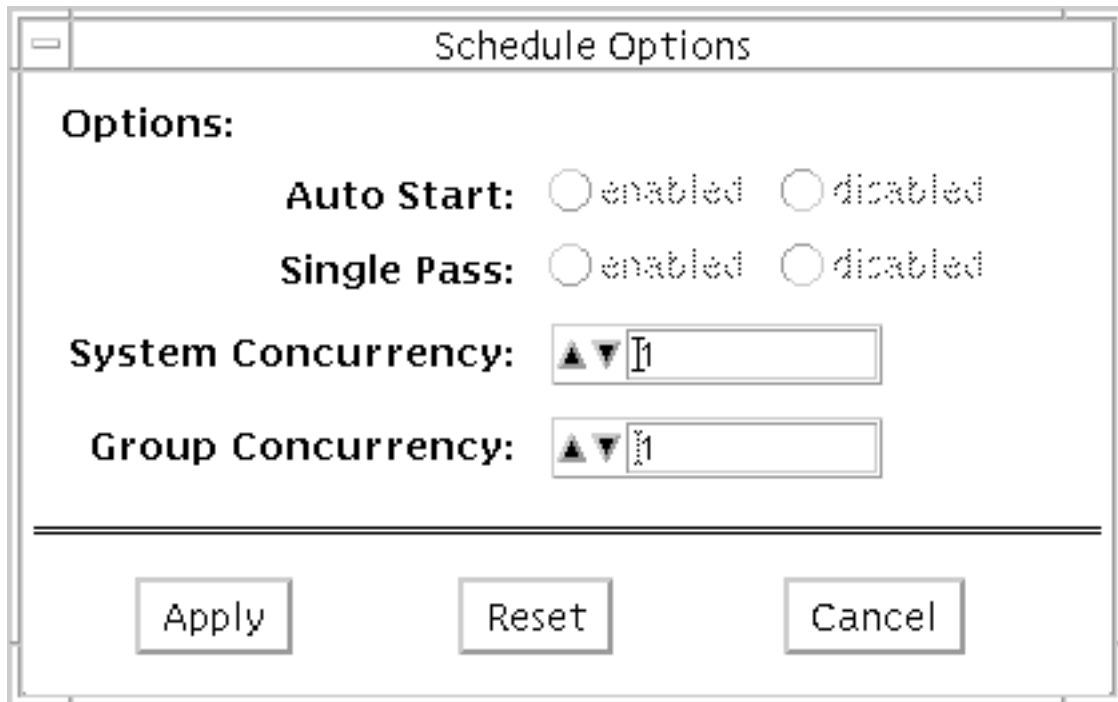
---

## *The SunVTS Menu Selections*

- View – This menu provides two options:
  - ▼ Open System map – Display full device selection list
  - ▼ Close System map – Display default device selection list
- Options – The following selections are available:
  - ▼ Thresholds – Specify number of passes, errors, and time to run
  - ▼ Notify – Specify a user to mail with test status information
  - ▼ Schedule, Test Execution, and Advanced – Run specified number of tests with stress, verbose, core file, or run on error option (see the next page)
  - ▼ Option files – Load, store, or remove a test options file
- Reports – Two selections are provided:
  - ▼ System configuration – Displays the system configuration report as obtained with the `prtconf` command
  - ▼ Log files – Displays the log file and allows selection of the level of information to log

## The Schedule Options Menu

Clicking on the Schedule option beneath the Options selection on the horizontal bar of the SunVTS window displays the window in Figure 6-3.



**Figure 6-3** Schedule Options Window

The available options are:

**Auto Start** – Run tests selected in a previously saved option file using a command-line specification when `sunvts` is invoked.

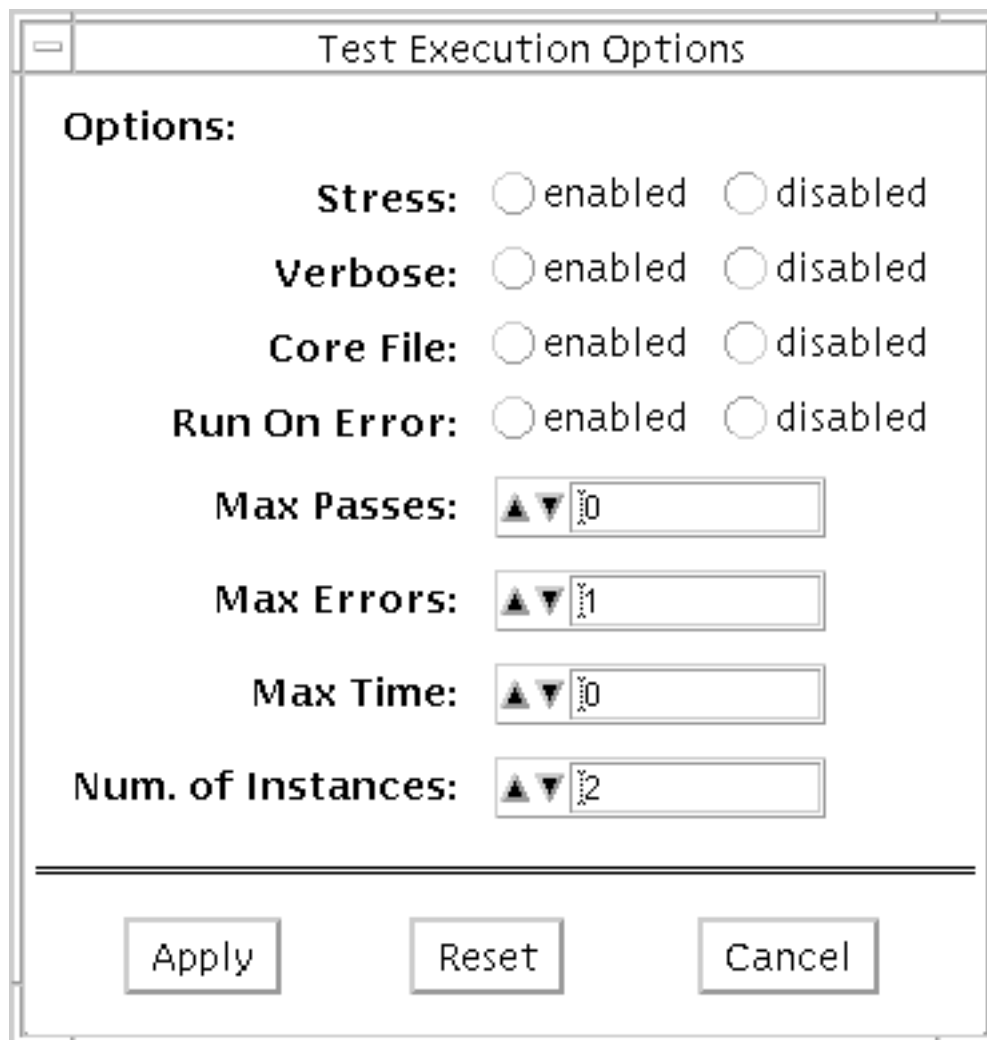
**Single Pass** – Run only one pass of each selected test.

**System Concurrency** – Specify the maximum number of tests that can be run concurrently on the machine being tested.

**Group Concurrency** – Specify the number of tests to be run at the same time in the same group.

## The Test Execution Menu

Clicking on Test Execution beneath the Options selection on the upper horizontal bar of the SunVTS menu displays the window in Figure 6-4.



**Figure 6-4** Text Execution Options Window



## The Test Execution Menu

- Stress
- Verbose
- Core file
- Run on Error
- Max Passes
- Max Errors
- Max Time
- Number of Instances

### *The Test Execution Menu*

The following is a list of options available in the Test Execution menu:

- Stress – Run certain tests in stress mode, working the system harder than normal.
- Verbose – Enable more information to be logged and displayed during testing.
- Core file – Allow core dump generation in the SunVTS bin directory when abnormal conditions occur. The core file name format is `core.testname.xxxxxx`.
- Run on Error – Continue testing until the `max_errors` value is reached.

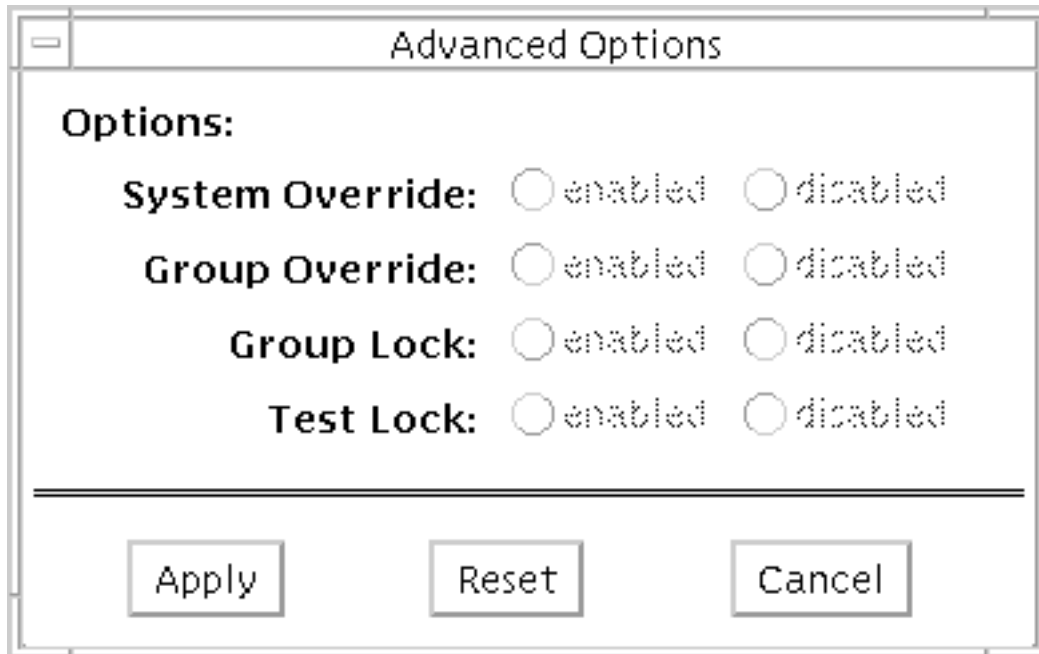
---

## *The Test Execution Menu*

- Max Passes – Specify the maximum number of passes that tests can run. A value of zero indicates no limit.
- Max Errors – State the maximum number of errors any test allows before stopping. A value of zero causes tests to continue regardless of errors.
- Max Time – Specify the maximum number of minutes tests are allowed to run. A value of zero indicates no limit.
- Number of Instances – Specify the number of tests to run for all tests that are scalable.

## The Advance Options Menu

Clicking on the Options selection on the topmost horizontal bar of the SunVTS window displays the window in Figure 6-5.

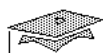


**Figure 6-5** Advanced Options Window

Available options are:

- System Override – Supersede group and test options in favor of the options selected in a Global Options window; set all options on all test group and test option menus.
- Group Override – Supersede specific test options in favor of the group options set in a Group Options window.
- Group Lock – Protect specific group options from being changed by the options set at the system level. (System Override supersedes this option.)
- Test Lock – Protect specific test options from being changed by options set at the group or system level. (System Override and Group Override supersede this option.)





Sun Educational Services

## Intervention Mode

Certain tests require that you intervene with

- Loopback connectors needed for certain tests
- Media needed to test tape, diskette, or CD-ROM drives

### *Intervention Mode*

Certain tests require that you intervene before you can run the test successfully. These include tests that require media or loopback connectors.

- Loopback connectors are required to run certain tests, such as serial port tests, successfully.

Refer to the *SunVTS Test Reference Manual* for more information about loopback connectors, and which tests need them.

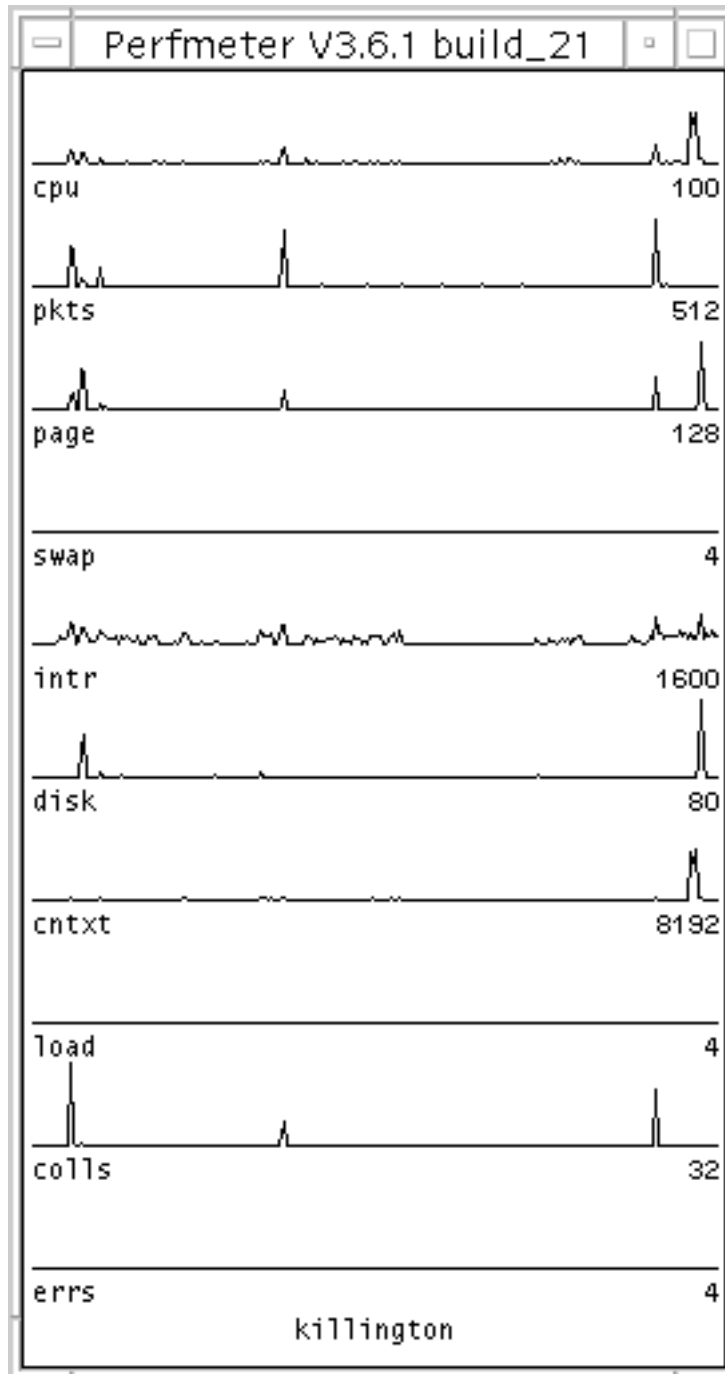
- Media (tapes, diskettes, or CD-ROMs) must be present in the drive(s) before the system is probed at SunVTS startup. If this is not done, the following error message is displayed:

Using old or damaged tapes and diskettes may cause errors in corresponding tests.

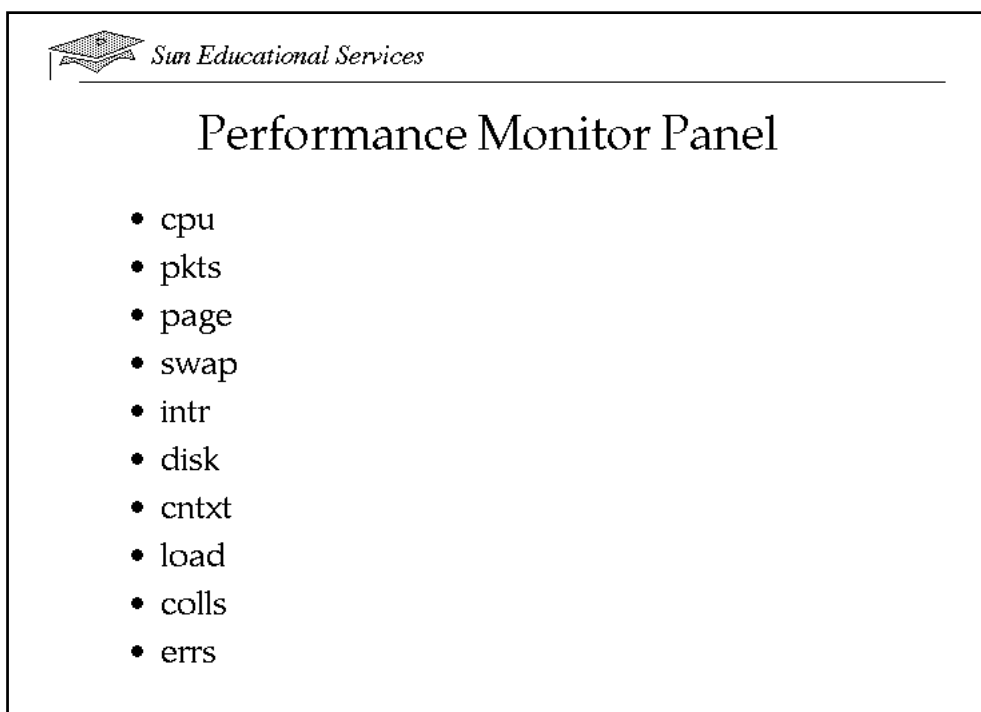
You cannot select these tests until you enable the intervention mode. This setting does not change the test function; it just serves as a reminder that you must intervene before the test can be successfully completed.

## Performance Monitor Panel

The performance monitor displays system resource activity. A brief description of each component is provided on the next page.



**Figure 6-6** Perfmeter Window



### *The Performance Monitor Panel*

The information displayed with the SunVTS Performance Monitor is the same as that displayed by the operating system `perfmeter` utility.

- `cpu` – Percentage of CPU used per second
- `pkts` – Ethernet packets per second
- `page` – Paging activity in pages per second
- `swap` – Jobs swapped per second
- `intr` – Number of device interrupts per second
- `disk` – Disk use in transfers per second
- `cntxt` – Number of context switches per second
- `load` – Average number of processes that have run over last minute
- `colls` – Collisions per second detected on the Ethernet
- `errs` – Errors per second on receiving packets

## Using SunVTS in TTY Mode

If you use the SunVTS software in TTY mode, no frame buffer is required. To run in TTY mode, perform the following steps:

1. Start the SunVTS kernel with the `vtstk` command.
 

```
# /opt/SUNWvts/bin/vtsk
```
2. Start the SunVTS TTY User Interface with the `vtstty` command
 

```
# /opt/SUNWvts/bin/vtstty
```

 or the `sunvts` command with the `-t` option.
 

```
# /opt/SUNWvts/bin/sunvts -t
```

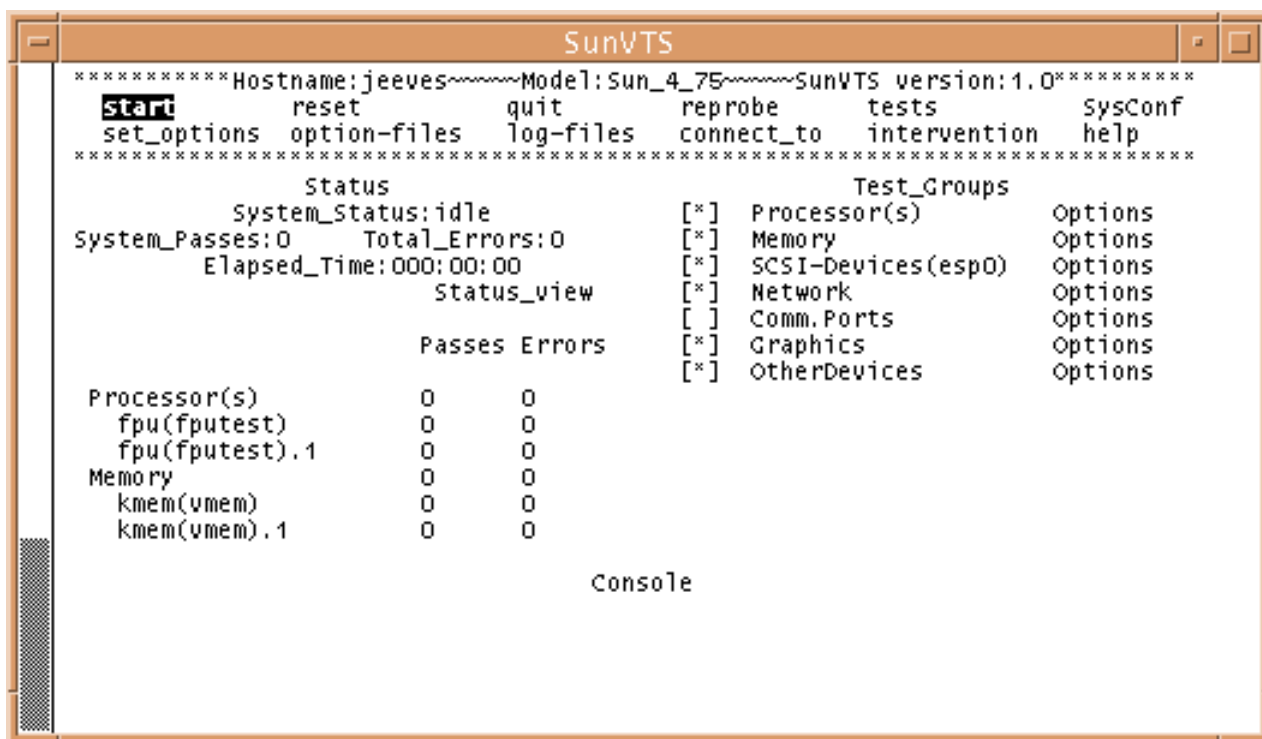


Figure 6-7 SunVTS Window

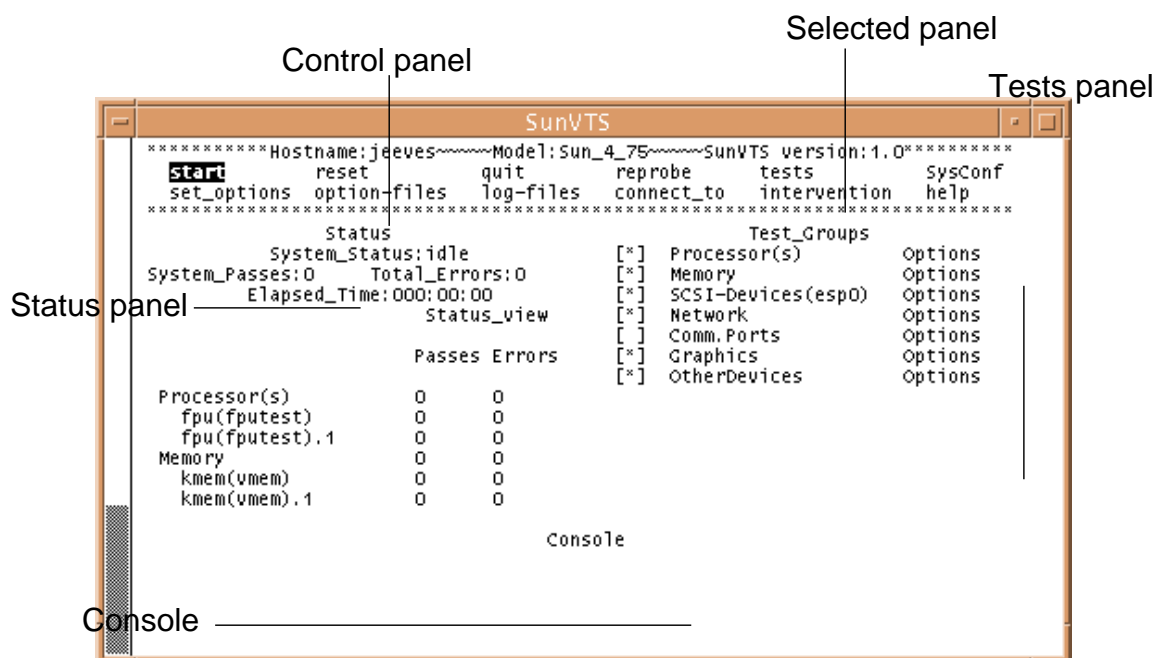
## Negotiating the SunVTS TTY Interface

The SunVTS TTY interface provides a screen with four working panels: *Console*, *Status*, *Control*, and *Tests*.

Messages pertaining to SunVTS tests are displayed in the Console panel. The asterisk highlights the selected working panel at a given time.

The following keys operate as follows with the TTY interface:

- Tab – Selects a screen panel for keyboard input
- Spacebar – Selects an option within a panel
- Arrows – Moves between the options in a panel
- Esc – Closes pop-up option windows
- Control-I – Refreshes the TTY window



**Figure 6-8** Various Working Panels of the SunVTS TTY Interface



## Running SunVTS Remotely

- Kernel interface
- User interface
- Graphical user interface

### *Running SunVTS Remotely*

A testing session can be run across a network or even a modem. Both the kernel and the user interface components are used in remote testing.

#### *Kernel Interface*

To test a remote system, it *must* have the kernel process `/opt/SUNWvts/bin/vtsk` running.

#### *User Interface*

To test local system, the user interface can be either TTY (teletype) or graphical.

---

## *Running SunVTS Remotely*

### *Graphical User Interface*

The graphical user interface (GUI) component *must* have the interface `/opt/SUNWvts/bin/sunvts` running as an active process. To connect while running SunVTS, perform the following steps:

1. Click on Commands in the upper horizontal bar of the SunVTS window.
2. Click on Connect to Host.
3. Type the name of the remote system to be tested.
4. Click on the Apply button.



## The User Interface

- Connecting directly to the remote computer

```
/opt/SUNWvts/bin/sunvts -h remote_hostname
```

- Using the TTY interface

```
/opt/SUNWvts/bin/vtstty
```

## *The User Interface*

### *Connecting Directly to the Remote Computer*

You can also connect directly to the remote computer running the SunVTS kernel when starting the graphical user interface.

```
/opt/SUNWvts/bin/sunvts -h remote_hostname
```

### *TTY interface*

The TTY interface process `/opt/SUNWvts/bin/vtstty` is run as a process from a terminal that is logged in to the computer to be tested.

Testing can now be carried out as described earlier in “The SunVTS Graphical User Interface” section.



## Exercise: Diagnosing Problems With SunVTS



**Exercise objective** – The objective of this exercise is to

- Install the SunVTS package on a system
- Review SunVTS options
- Monitor the tests as they run
- Stop the testing session
- Review the results

and optionally

- Set up SunVTS tests for local and remote testing
- Run the tests in graphical and TTY mode

### *Preparation*

To complete this lab, you will need

- At least two networked SPARC desktop workstations running the Solaris 7 operating system.
- Access to a copy of the `SUNWvts` package, either from a server on the lab network or on the *Software Supplement for the Solaris 7 Operating Environment* CD-ROM.

## Exercise: Diagnosing Problems With SunVTS

### Tasks

In this lab exercise, you are going to verify that all hardware on your lab system is functional. You will need the SunVTS software present on your system.

1. Log in to the system as root.
2. Change to the `/opt` directory, and list the contents.  

```
# cd /opt
# ls
```
3. If `SUNWvts` is already present, run the `pkgrm` command to remove the old `SUNWvts` package.  

```
# pkgrm SUNWvts
```
4. Install SunVTS using the `pkgadd` program or by running the installation script as described earlier in this module.
5. Change to the `/opt/SUNWvts/bin` directory, and start `sunvts`.
6. Display the configuration on your workstation using the Reports menu.
7. Enable the verbose option (Options ► Test Execution), and send yourself mail notification upon error (Options ► Notify).
8. Save the configuration by choosing Options ► Option Files.
9. Start the session, and observe.
10. Read the manual pages for
  - ▼ `sunvts`
  - ▼ `vtstty`
  - ▼ `vtsui`
  - ▼ `vtsk`
  - ▼ `vtsprobe`

## Exercise: Diagnosing Problems With SunVTS

Now that you have a general idea of how the diagnostics work, become more familiar with the features of SunVTS by completing the following steps:

1. Run tests on another machine in the lab.
2. Run an intervention test.

The instructor can supply you with media.

3. Try the TTY interface.
4. Run tests remotely using Connect options or by invoking the `sunvts` program with the `-h` option.  

```
# /opt/SUNWvts/bin/sunvts -h host_name
```
5. Kill the `sunvts` kernel process and try the previous two steps again.
6. Deselect all tests.
7. Run the audio test. Observe the different selections that are played depending on the machine you are testing.
8. Run only the frame buffer test.
9. Find the maximum number of passes allowed for the `fputest?`.
10. Attempt to force an error. If your instructor has any failed hardware attached, observe a test on that. Otherwise, do the following:
  - a. Run a test that requires intervention, and do not attach the necessary media or cables.
  - b. Disconnect a peripheral, and then try to test that peripheral.
  - c. Disconnect the Ethernet cable.
  - d. Run `nettest` on a non-existent machine.
11. View your email and peruse the log file information.
12. Complete Workshop #39 if time allows.

## *Exercise: Diagnosing Problems With SunVTS*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

---

## *Check Your Progress*

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- Install the SunVTS package on a system
- Select, set up, and run SunVTS diagnostic tests
- Run SunVTS over a network
- Run SunVTS in TTY mode without a frame buffer
- Analyze SunVTS test results

## *Think Beyond*

From your recent experiences using SunVTS and POST diagnostics, how do these differ? The SyMON software interfaces with SunVTS. Which SunSolve database documents are related to SunVTS and SyMON?

## *Objectives*

Upon completion of this module, you should be able to

- Describe how a process creates a system dump
- Configure a system to collect and store core files
- Differentiate a system panic condition from a system hang
- List the steps to perform an initial core dump analysis
- Use the adb and crash debuggers to analyze crash dumps

## Relevance



**Discussion** – This module provides the information and tools needed to analyze crash dumps. The skills developed in this module are useful in troubleshooting cases of system panics as well as hung systems.

Before beginning Module 7, consider the following questions in relation to your work with Solaris systems:

- Have you experienced the condition of a system panic on any of your Solaris systems, and if so, how did you analyze the problem and reach a solution?
- How do you configure a system so that it can successfully collect crash dumps?
- What software package discussed earlier in the course provides access to the ISCTA program?

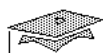
## References



**Additional resources** – The following references can provide additional details on the topics discussed in this module:

- SPARC International. *The SPARC Architecture Manual*.
- Goodheart and Cox. *The Magic Garden Explained*.
- C. Drake and K. Brown. *Panic! UNIX System Crash Dump Analysis*





## Introduction

### System panics

- Display panic message at the console
- Perform stack trace, listing routines that led to panic
- Dump “interesting” portions memory to swap
- Reboot, copying the core file from swap to a file system

## *Introduction*

### *System Panics*

Panics occur when unexpected, critical conditions are detected. Since certain conditions can cause data corruption, the system response is to stop processing, which prevents further damage.

There are two methods available in the code for calling the panic routine. While there is a kernel routine called `panic()` that can be directly invoked, the preferred method is the `cmn_err()` routine with the flag `CE_PANIC` specified as the error level.

The following events occur in response to a system panic:

- Display a panic message at the console
- Perform a stack trace, listing routines that led to the panic
- Dump “interesting” portions memory to the swap device
- Reboot and copy the core file from swap to a file system



## Introduction

### System hangs

- An application has hung.
- A terminal or window has hung.
- The system has hung.
- The system is overloaded and has a resource bottleneck.

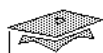
## *Introduction*

### *System Hangs*

A system that has hung appears to have stopped processing. A crash file is not automatically created in response to this condition, so debugging procedures are different than those used with panics.

The severity of the hang needs to be monitored by trying different methods of system access, and varies depending on the cause of the hang, which commonly include:

- An application has hung.
- A terminal or window has hung.
- The system has hung.
- The system is overloaded and has a resource bottleneck.



Sun Educational Services

## Configuring a System to Collect Crash Dumps

### The dumpadm utility

- `savecore` is invoked automatically by the Solaris 7 operating system.
- You can size the primary swap device and `/var/crash`.
- To view current configuration, use `dumpadm`.

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/killington
Savecore enabled: yes
```

## *Configuring a System to Collect Crash Dumps*

### *The dumpadm Utility*

The `savecore` function, which collects a crash dump from the swap device and copies it to the file system, is invoked automatically in Solaris 7. Earlier releases required configuration of the utility, as described on page 7-6.

The administrator needs to ensure that the size of the primary swap device is large enough to collect the crash dump, and that the size of `/var/crash` is large enough for `savecore` to successfully copy the crash dump from swap to the file system.

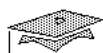
## Configuring a System to Collect Crash Dumps

### *The dumpadm Utility (Continued)*

Details of the crash dump creation and collection can be examined and configured with `dumpadm`. To view the current configuration, enter the `dumpadm` command at the shell prompt with no arguments.

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/killington
Savecore enabled: yes
```

The defaults are represented by the output shown, which include to enable `savecore`, collect kernel pages only, dump the file to the swap device, and save it to the `/var/crash/host_name` directory upon reboot.



Sun Educational Services

## The dumpadm Utility

- `-c content_type`
  - `-d dump_device`
  - `-m min`
  - `-n`
  - `-s savecore_dir`
  - `-u`
- ```
# dumpadm -c kernel -d swap -s /opt/crash/system1
```

### The dumpadm Utility

The options available with `dumpadm` to change the crash dump and savecore configuration details are:

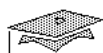
- `-c content_type` – Specify the contents of the crash dump using the key word *kernel*, for kernel pages only, or *all*, for all memory pages.
- `-d dump_device` – Specify the target device to which the crash dump is written in the case of a panic. The key word *swap* specifies the swap device. Disk device partitions can also be used.
- `-m min` – Specify the minimum amount of free space that savecore should leave in the file system where crash dumps are collected. The key word *min* is followed by *k* for Kbytes, *m* for Mbytes, or the percent sign (%) to indicate a particular percentage.
- `-n` – Suppress the automatic execution of savecore on reboot. (Not recommended.)

## *The dumpadm Utility*

- `-s savecore_dir` – Specify the absolute path of the directory into which `savecore` should copy the crash dump files. The default is `/var/crash/host_name`.
- `-u` – Update the configuration details based on the contents of `/etc/dumpadm.conf`. This file is consulted on reboot in order to restore the settings in effect at the time of the last boot.

The following command specifies collect the core file on swap, populate the file with kernel pages only, and save the core file to `/opt/crash/system1` on reboot:

```
# dumpadm -c kernel -d swap -s /opt/crash/system1
```



Sun Educational Services

## The savecore Utility

Excerpt from `/etc/init.d/sysetup`:

```
##
## Default is to not do a savecore
##
#if [ ! -d /var/crash/`uname -n` ]
#then mkdir -p /var/crash/`uname -n`
#fi
# echo `checking for crash dump...\c `
#savecore /var/crash/`uname -n`
# echo ``
```

### *The savecore Utility*

Systems running Solaris 2.6 or earlier releases need to configure the savecore utility so that it executes automatically.

When `savecore(1M)` runs, it makes a copy of the kernel symbol table (`/dev/ksyms`) that was running, called `unix.n`, and dumps physical memory to a file called `vmcore.n` in the specified directory (by default `/var/crash/machine_name`).

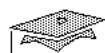
There must be enough space in `/var/crash` to contain the core dump or it will be truncated. The file appears larger than it actually is because it contains holes – so avoid copying it. Debuggers can be used on the core dump and the symbol table files.

## *The savecore Utility*

In the `/etc/init.d/syssetup` file, uncomment the following lines. If needed, the target directory name (`/var/crash`) can be changed.

```
##
## Default is to not do a savecore
##
#if [ ! -d /var/crash/`uname -n` ]
#then mkdir -p /var/crash/`uname -n`
#fi
# echo `checking for crash dump...\c `
#savecore /var/crash/`uname -n`
# echo ``
```





*Sun Educational Services*

## Debuggers

- adb
- crash
- kadb

### *Debuggers*

#### adb

The assembly debugger `adb` is an interactive, general-purpose utility. It can be used to examine files, and it provides a controlled environment for the execution of programs. `adb` reads commands from the standard input and displays responses on the standard output. By default, it does not supply a prompt.

#### crash

The functionality of `crash` is similar to `adb`, however, the command interface is different. The `crash` debugger is used to examine the system memory image of a running or a crashed system by formatting and printing control structures, tables, and other information.

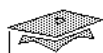
## *Debuggers*

### kadb

kadb is an interactive debugger with a user interface similar to that of adb. kadb must be loaded prior to the standalone program it is to debug. To run the kernel under the control of the kadb debugger, type

**boot kadb**

This mode is useful for setting breakpoints in kernel routines, and for analyzing system hang conditions.



Sun Educational Services

## Invoking the Debuggers

- adb

```
# cd crash_directory
# adb -k unix.n vmcore.n

# adb -kw /dev/ksyms /dev/mem
physmem 272a

# adb -kw -P "adb: " /dev/ksyms /dev/mem
physmem 272a
adb:
```

- crash

```
# crash vmcore.n unix.n
dumpfile = vmcore.0, namelist = unix.0, outfile = stdout
>
```

## Invoking the Debuggers

### adb

To invoke adb for crash dump analysis, type

```
# cd crash_directory
# adb -k unix.n vmcore.n
```

---

**Note** – *n* is a value starting at 0. Ensure that `vmcore` and `unix` both have the same *n* value within the command line.

---

To start adb on a live system, run the following command:

```
# adb -kw /dev/ksyms /dev/mem
physmem 272a
```

## Invoking the Debuggers

### adb (*Continued*)

The file `/dev/ksyms` invokes a special driver that provides an image of the kernel's symbol table. To examine the kernel symbol table, use the `/usr/ccs/bin/nm` command with `/dev/ksyms` as the argument.

When `adb(1)` responds with `physmem nnn`, it is ready for a command. If you prefer to work with a prompt, use the `-P` option:

```
# adb -kw -P "adb: " /dev/ksyms /dev/mem
physmem 272a
adb:
```

### crash

To invoke `crash` for core dump analysis, type

```
# crash vmcore.n unix.n
dumpfile = vmcore.0, namelist = unix.0, outfile = stdout
>
```

To examine a running kernel, type `crash` with no arguments.

---

## General Purpose adb Commands

Some commonly used j commands are:

|              |                                                                                           |
|--------------|-------------------------------------------------------------------------------------------|
| b            | Set a breakpoint (only possible in kadb for the kernel)                                   |
| \$b          | Display all breakpoints (kadb, adb on user programs)                                      |
| d            | Delete a breakpoint                                                                       |
| \$z          | Delete all breakpoints                                                                    |
| \$M          | Display built-in macros (kadb only)                                                       |
| cpu\$<cpus   | Display cpu0, which contains the address of the currently running thread                  |
| cpun \$< cpu | Display the cpu identified by the number <i>n</i>                                         |
| \$<msgbuf    | Display the msgbuf structure, which contains the console messages leading up to the panic |

## General Purpose adb Commands

|                  |                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| \$c              | Display the stack trace                                                                                                                          |
| \$C              | Show the call trace and arguments at the time of the crash as well as the saved frame pointer and the saved program counter for each stack frame |
| \$r              | Display the SPARC window registers, including the program counter and the stack pointer                                                          |
| <sp\$<stacktrace | Use the <i>sp</i> (stack pointer) address to locate and display a detailed stacktrace                                                            |
| \$q              | Quit                                                                                                                                             |
| w or W           | Write a 2- or 4-byte value to address                                                                                                            |
| Z <i>value</i>   | Write an 8-byte value to address                                                                                                                 |
| \$> <i>file</i>  | Redirect output to <i>file</i>                                                                                                                   |
| >                | Reset standard output to the default                                                                                                             |

## Format Specifications With `adb`

The default format used by `adb` is hexadecimal, which can be changed with the following specifiers. With the `?`, `/`, and `=` commands, output format specifiers can be used.

General usage is lowercase letters for 2-byte output and uppercase letters for 4-byte output.

|                                  |                                                                                                       |
|----------------------------------|-------------------------------------------------------------------------------------------------------|
| <code>o</code> or <code>O</code> | Display in octal (short, long)                                                                        |
| <code>q</code> or <code>Q</code> | Display in signed octal (short, long)                                                                 |
| <code>d</code> or <code>D</code> | Display in signed decimal                                                                             |
| <code>x</code> or <code>X</code> | Display in hex                                                                                        |
| <code>u</code> or <code>U</code> | Display unsigned decimal                                                                              |
| <code>f</code> or <code>F</code> | Display as floating point (long or double)                                                            |
| <code>K</code>                   | Print pointer or long in hex (4 bytes for 32-bit kernel; 8 bytes for 64-bit kernel)                   |
| <code>b</code>                   | Print as an octal byte                                                                                |
| <code>s</code>                   | Print as a null-terminated string                                                                     |
| <code>c</code>                   | Display as a single character                                                                         |
| <code>C</code>                   | Display a single character using escape conventions for nonprinting characters                        |
| <code>i</code>                   | Display as a disassembled instruction (mnemonic code)                                                 |
| <code>0t</code>                  | Specify base 10 or decimal for a value being input to <code>adb</code> ; hex is the default (base 16) |

## adb *Commands*

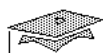
The general form of an adb(1M) command is

```
[ address ] [ ,count ] command [ ; ]
```

If *address* is omitted, the current location is used. (The dot [*.*] also stands for the current location.) The address can be a kernel symbol. If the *count* is omitted, it defaults to 1. Commands to adb consist of a verb followed by a modifier or list of modifiers. Verbs can be:

|        |                                                                                                                      |
|--------|----------------------------------------------------------------------------------------------------------------------|
| ?      | Enables you to examine code or variables in the object file (executable).                                            |
| /      | Enables you to examine data from the core file.                                                                      |
| =      | Prints values in different formats.                                                                                  |
| \$<    | For miscellaneous commands, includes macro invocations.                                                              |
| >      | Assigns a value to a variable or register.                                                                           |
| <      | Reads a value from a variable or register.                                                                           |
| Return | Repeats the previous command with a count of 1. Increments the current location represented by the dot ( <i>.</i> ). |





## Syntax Example

```
# f0cebacd,10/X
```

- The *address* is f0cebacd. adb goes to this location in memory
- The *count* is 10 separated by a comma from the address. The command is applied for a count of 10.
- The *command* is / (slash), which specifies to display data from the core file.
- The X specifies 4-byte hexadecimal output.

## Syntax Example

The following command illustrates the *address*, *count*, *command* format used with adb. The output of this command is to display sixteen 4-byte fields starting at address f0cebacd.

```
# f0cebacd,10/X
```

- The *address* is f0cebacd. adb goes to this location in memory
- The *count* is 10 separated by a comma from the address. The command is applied for a count of 10.
- The *command* is / (slash), which specifies to display data from the core file.
- The *x* specifies 4-byte hexadecimal output.

The output is sixteen 4-byte fields starting at address f0cebacd.

Why does adb display 16 fields (not 10) in this example?



## Register References in adb

- Eight general purpose registers – %g0 through %g7
- Eight input registers – %i0 through %i7
- Eight output registers – %o0 through %o7
- Eight local registers – %l0 through %l7

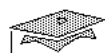
## *Register References in adb*

Registers contain machine status information at all times. In crash analysis, certain registers are examined to extract information that helps to determine the cause of the crash.

In adb, the percent sign (%) and the less-than sign (<) are used in register references. SPARC chips have over 150 registers. Of these, 32 general purpose registers are available to functions:

- Eight general-purpose registers – %g0 through %g7
- Eight input registers – %i0 through %i
- Eight output registers – %o0 through %o7
- Eight local registers – %l0 thorough %l7

For core dump analysis, the most important registers are the program counter (<o7 or pc), which holds the current instruction (in a crash dump, this is the instruction that caused the panic), and the stack pointer (<o6 or sp), which points to the routines currently on the stack (in a crash dump, these are the routines that led to the panic).



## Register References in adb

### Reserved registers

- o6 – Stack pointer (sp)
- o7 – Program counter (pc)
- i6 – Frame pointer (fp)
- g0 – Always zero
- g7 – Address of the current thread

## *Register References in adb*

Reserved registers within the 32 general purpose registers include

- o6 – Stack pointer (sp)
- o7 – Program counter (pc)
- i6 – Frame pointer (a way to trace through the stack to the previous function)
- g0 – Zero
- g7 – Address of the current thread



## Using Macros in adb

- A macro is a file that contains adb commands.
- Most macros display commonly used structures.
- Macros are bundled with the Solaris operating system.
- Search order for the macros is
  - Current directory
  - /usr/lib/adb
  - Directory specified with the -I option to adb

### *Using Macros in adb*

A macro is a file that contains a number of adb commands in the same format as would be entered interactively to the debugger. Over two hundred macros are available to facilitate working with the debugger.

Most macros display commonly referenced structures; for example, the `proc` macro displays the process structure, the `thread` macro displays the thread structure, and the `inode` macro displays the `inode` structure.

Macros reduce the need to communicate to adb strictly by using hexadecimal addresses, and provide annotation on output displays that facilitate interpretation of information.

The macros are bundled with the Solaris operating system, and can be found in the `/usr/lib/adb` directory. When adb is given a macro name, it searches the following locations to locate the macro name:

- The current directory
- The standard directory, `/usr/lib/adb`
- The directory specified on the command line using the `-I` option

## adb *Macros*

The following are available adb macros:

|                 |                |                 |                |
|-----------------|----------------|-----------------|----------------|
| as              | inode          | proc2u          | snode          |
| bootobj         | iocblk         | procthread      | stack          |
| buf             | iovec          | procthread.list | stackregs      |
| bufctl          | itimerval      | ptbltohme       | stacktrace     |
| bufctl_audit    | kmem_cache     | ptbltopte       | stacktrace.nxt |
| cache           | kosyminfo      | ptetoptbl       | stat           |
| cache           | ksiginfo       | putbuf          | stdata         |
| callout         | lwp            | putbuf.wrap     | strtab         |
| calltrace       | mblk           | qinit           | svcfh          |
| calltrace.nxt   | mblk.nxt       | qproc.info      | sysinfo        |
| cnode           | memlist        | qthread.info    | tcpcb          |
| cpu             | memlist.list   | queue           | tcpip          |
| cpun            | memlist.nxt    | regs            | thread         |
| cpus            | memseg         | rlimit          | thread.trace   |
| cpus.nxt        | mntinfo        | rnode           | threadlist     |
| cred            | modctl         | rpctimer        | threadlist.nxt |
| ctx             | modctl_list    | rwindow         | tmount         |
| dblk            | modinfo        | rwlock          | tmpnode        |
| devinfo         | modlinkage     | seg             | traceall.nxt   |
| dino            | module         | segdev          | tsdpent        |
| direct          | modules        | seglst          | tsproc         |
| disp            | modules.nxt    | seglst.nxt      | tune           |
| dispq           | msgbuf         | segmap          | u              |
| dispq.nxt       | msgbuf.wrap    | segn            | u.sizeof       |
| dispqtrace      | mutex          | sema            | ucalltrace     |
| dispqtrace.list | netbuf         | session         | ucalltrace.nxt |
| dispqtrace.nxt  | page           | setproc         | ufchunk        |
| dumphdr         | page2hme       | setproc.done    | ufchunk.nxt    |
| exdata          | page2hme.nxt   | setproc.nop     | uio            |
| file            | pathname       | setproc.nxt     | ustack         |
| filsys          | pcb            | sigaltstack     | utsname        |
| hat             | pid            | slab            | v              |
| hme             | pid.print      | sleepq          | v_call         |
| hme.sizeof      | pid2proc       | sleepq.nxt      | v_proc         |
| hmelist         | pid2proc.chain | slpqtrace       | vattr          |
| hmelist.nxt     | pollhead       | slpqtrace.list  | vfs            |
| hmetoptbl       | prgregset      | slpqtrace.nxt   | vnode          |
| ifnet           | proc           | smap            |                |



## Header Files and adb Macros

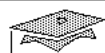
- adb macros display information about C programming structures.
- C structures are defined in header files.
- Header files can be used to determine what the information is that adb macros display.

### *Header Files and adb Macros*

As previously mentioned, many adb macros display information about actual C programming structures. Definitions for these structures can be found in various header files (files that have a `.h` suffix) on the system.

If you have some programming background, you can use these header files to help you determine what the different fields of information displayed by an adb macro are.

Header files are provided with the Solaris distribution. Each header defines one or more structures; for example, the `proc.h` header file defines exactly what is contained in a process structure.



## Header Files and adb Macros

- /usr/include/sys
- /usr/include/vm
- /usr/include/sys/fs
- /usr/platform/*arch\_name*/include/sys
- /usr/platform/*arch\_name*/include/vm
- /usr/include

### *Header Files and adb Macros*

Header files are found in the following locations:

- /usr/include/sys – Most of the system header files
- /usr/include/vm – Header files that describe virtual memory (vm) structures (page.h, seg.h, and so on.)
- /usr/include/sys/fs – Header files that describe file system structures and types
- /usr/platform/*arch\_name*/include/sys – Architecture dependent structures defined in the header files
- /usr/platform/*arch\_name*/include/vm (and sys) – Architecture dependent structures defined in the header files
- /usr/include – The net, nfs, rpc, protocols, inet, and netinet directories with headers that define networking data structures



## Frequently Used Header Files and adb Macros

- `/usr/include/sys/proc.h` and `proc`
- `/usr/include/sys/thread.h` and `thread`
- `/usr/include/sys/klwp.h` and `lwp`
- `/usr/include/sys/user.h` and `user`
- `/usr/include/sys/cred.h` and `cred`
- `/usr/include/vm/as.h` and `as`
- `/usr/include/vm/seg.h` and `seg`

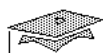
### *Header Files and adb Macros*

#### *Frequently Used Header Files and adb Macros*

The primary goal in crash analysis is to locate the process, the thread within the process, the function within the thread, and the instruction within the function that caused the panic. Other areas may also need to be examined depending on the nature of the panic. The most useful information for core dump analysis purposes is found in the following header files and adb macros:

- `/usr/include/sys/proc.h` and the `proc` macro
- `/usr/include/sys/thread.h` and the `thread` macro
- `/usr/include/sys/klwp.h` and the `lwp` macro
- `/usr/include/sys/user.h` and the `user` macro
- `/usr/include/sys/cred.h` and the `cred` macro
- `/usr/include/vm/as.h` and the `as` macro
- `/usr/include/vm/seg.h` and the `seg` macro





## Core Dump Analysis

- What instruction failed
- What thread was running
- What process was running
- What arguments were passed to the failing process

### *Core Dump Analysis*

#### *Sample Problem*

During the development of the RAM disk driver, the system crashes with a data fault when running `newfs`. The `strings`, `adb` and `crash` utilities, are used to determine

- What instruction failed
- What thread was running at the time of the panic
- What process was running at the time of the panic
- What arguments were passed to a failing process

## *Core Dump Analysis*

### *Additional Information*

Sun Educational Services offers ST-370: *Core Dump Analysis*, a course that provides three days of instruction in this area. In addition to the ISCDA document, SunSolve provides good examples of knowledgeable users' analyses of system core dumps, which can be a helpful learning tool.

## Core Dump Analysis Using adb

Use the adb debugger and perform the following steps:

```
# adb -k unix.0 vmcore.0
physmem 1e6e

$<msgbuf
BAD TRAP: type=9 rp=f05246f4 addr=30 mmu_fsr=326 rw=1
BAD TRAP: occurred in module "ramd" due to an illegal access
to a user address
mkfs: Data fault
kernel read fault at addr=0x30, pme=0x0
MMU fsr=326: Invalid Address on supv data fetch at level 3
pid=465, pc=0xfc479dbc, sp=0xf0524740, psr=0x40000c2,
context=0
g1-g7: ffffffff98, 0, ffffffff00, 0, f05249e0, 1, fc2dec00
Begin traceback... sp = f0524740
Called from f00df9b4, fp=f05247a8, args=1a40000 f0524808
fc38fc80 f0154664 0 fc479d90
Called from f0070258, fp=f05248b8, args=200 f0524920 2 0 4
fc2d5b04
Called from f0041aa0, fp=f0524938, args=f0160cf8 f0524eb4 0
f0524e90 ffffffff fc ffffffff
Called from 15cc0, fp=efffae8, args=4 32400 200 0 0 3fe00

End traceback...
panic: Data fault
```

## Core Dump Analysis Using adb

The `$c` macro displays the stack. On 32-bit systems, the second argument to `trap()` is a pointer to the registers associated with the thread at the time of panic. On a 64-bit system, use the second argument to `die()` or the `rp` pointer from the strings output with the `regs` macro.

```
$c
complete_panic(0xf026b428,0xfbfbfab98c,0xf0048ec8,0x6a,0xfb
fab818,0xf0279800) + 108
do_panic(?) + 1c
vcmn_err(0xf0266600,0xfbfbfab98c,0xfbfbfab98c,0x7,0xffeec000,
0x3)
cmn_err(0x3,0xf0266600,0x1,0x21,0x21,0xf025c000) + 1c
die(0x9,0xf05246f4,0x30,0x326,0x1,0xf0266600) + bc
trap(0xf028a1d8,0xf05246f4,0x0,0x326,0x1,0x0) + 4f8
fault(?) + 84
Syssize(via
getminor)(0x0,0x3ffff,0x20,0x7fffffff,0xf5c4b4bc,0x315854
86)
ram_write(0xdc0000,0xfbfbabbd8,0xf5a8ed38,0xdc,0xf5970d48,
0xf5c54d90) + 1c
write(0x5) + 190
```

The second argument to `trap` from the stack trace is passed to the `regs` macro. Note the `pc` field. The output is truncated.

### **f05246f4**`$<regs`

|             |          |                 |          |          |
|-------------|----------|-----------------|----------|----------|
| 0xfc020ac4: | psr      | <b>pc</b>       | npc      |          |
|             | 110000c4 | <b>fc479dbc</b> | f5c98dc0 |          |
| 0xfc020ad0: | y        | g1              | g2       | g3       |
|             | 50000000 | ffffff98        | 0        | f00bb080 |
| 0xfc020ae0: | g4       | g5              | g6       | g7       |
|             | 40       | f5ca0648        | 1        | fc2dec00 |
| 0xfc020af0: | o0       | o1              | o2       | o3       |

## Core Dump Analysis Using adb

The next step displays the instruction contained in the program counter. To view more of the routine, type `ram_write/40ai`.

```
fc479dbc/ai
ram_write+0x2c: ld      [%l1 + 0x30], %l2
```

Now use the variable `panic_thread` to display the thread that caused the panic.

**panic\_thread/X** (On a 64-bit system, use `/K`)

```
panic_thread:
panic_thread:  fc2dec00
```

**fc2dec00\$<thread**

```

                                link      stk
                                0         fbfabc08
0xf5c6648c:
                                bound     affcnt      bind_cpu
                                f026b494   0            -1
0xf5c66494:
                                flag      procflag    schedflag   state
                                0         0           11          4
0xf5c664a0:
                                pri       epri       pc          sp
                                14        0         f0048ec8   fbfab818
0xf5c664ac:
                                wchan0   wchan     cid        clfuncs
                                0         0         2          f59a0378
0xf5c664d0:
                                nofault  swap      lock       cpu
                                0         fbfaa000 ff         f026b494
.....
0xf5c66524:
                                lwp     procp    next      prev
                                f5c11828 f5c0fcc8 f5c665a0  f5c66d80
```

## Core Dump Analysis Using adb

Use the `procp` address to display the user structure, and locate the `psargs` field. (The `procp` address can be passed to the `proc` macro to examine the `proc` structure, in which the `pidp` field points to the process ID.) The output again is truncated.

**f5c0fcc8\$<proc2u**

```

execid          execsz          tsize
32581           12e             0
0xf5c0fe8c:
                dsize          start          ticks          cv
0              31585486      405a4         0
0xf5c0fe9c:
                exdata
0xf5c0fe9c:
                vp            tsize          dsize          bsize
0              0              0              0
0xf5c0feac:
                lsize          nshlibs        mach           mag           toffset
0              0              0              0             10b          0
0xf5c0febc:
                doffset        loffset        txtorg         datorg
0              0              0              0              0
0xf5c0fecc: entloc ef7d43a8
0xf5c0fed0:  aux vector
              7d8             effffffe6      3              10034
0xf5c0ff60:  psargs mkfs /devices/pseudo/ramd@0:0,raw 512 8 1
8192 1024 16 10 60 204 8 t 0 -1 8 -1^@^@^@

```

**mkfs^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@**



Sun Educational Services

## Core Dump Analysis – Summary for Using adb

1. Obtain a stack trace  
`$c`
2. Obtain a register dump with the pc address  
`second_arg_to_trap$<regs`  
for a 32-bit machine, or  
`second_arg_to_die$<regs`  
for a 64-bit machine
3. Display the faulting instruction  
`pc_address/ai`

## Core Dump Analysis – Summary for Using adb

Enter the adb debugger and perform the following steps:

1. Obtain a stack trace

`$c`

Note the second argument to `trap()` on a 32-bit machine, or the second argument to `die()` on a 64-bit machine.

2. Obtain a register dump and note the pc address. Use

`second_arg_to_trap$<regs` (or use the `rp` pointer)

for a 32-bit machine, or use the following for a 64-bit machine:

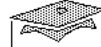
`second_arg_to_die$<regs`

3. Use the pc address to display the faulting instruction.

`pc_address/ai`

To examine more of the routine that contains the faulting instruction, use the syntax

`routine_name/20ai`



## Core Dump Analysis – Summary for Using adb

4. Dump the `panic_thread` symbol

```
panic_thread/X or panic_thread/K
```

5. Run the thread macro

```
panic_thread_address$<thread|
```

6. Display the process structure

```
procp_address$<proc2u
```

## Core Dump Analysis – Summary for Using adb

4. Dump the `panic_thread` symbol. The address matches the thread value for one of the CPUs.

For 32-bit systems, type

```
panic_thread/X
```

For 64-bit systems, type

```
panic_thread/K
```

5. In the thread macro output, note the `procp` value.

```
panic_thread_address$<thread
```

6. Display the process structure, and note the `psargs` field. That is your failing process.

```
procp_address$<proc2u
```



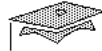
## Information Gathering Using strings

The steps included in this analysis example were obtained on a sun4m system, but can be performed on other architectures, such as sun4u, except where noted. The appearance of the addresses and some details of the display change if working on a 64-bit kernel.

Some of the following output has been truncated. You can use the `msgbuf` macro in `adb` as an alternative to `strings`. This command is illustrated on the following page.

```
# strings vmcore.0 | more
BAD TRAP: type=9 rp=f05246f4 addr=30 mmu_fsr=326 rw=1
BAD TRAP: occurred in module "ramd" due to an illegal access
to a user address
mkfs: Data fault
kernel read fault at addr=0x30, pme=0x0
MMU fsr=326: Invalid Address on supv data fetch at level 3
pid=465, pc=0xfc479dbc, sp=0xf0524740, psr=0x40000c2,
context=0
g1-g7: ffffffff98, 0, ffffffff00, 0, f05249e0, 1, fc2dec00
Begin traceback... sp = f0524740
Called from f00df9b4, fp=f05247a8, args=1a40000 f0524808
fc38fc80 f0154664 0 fc479d90
Called from f0070258, fp=f05248b8, args=200 f0524920 2 0 4
fc2d5b04
Called from f0041aa0, fp=f0524938, args=f0160cf8 f0524eb4 0
f0524e90 ffffffff fc ffffffff
Called from 15cc0, fp=effffae8, args=4 32400 200 0 0 3fe00

End traceback...
panic: Data fault
```



## Information Gathering – Summary for Using strings

```
# strings vmcore.0 | more
```

- The panic message
- The program counter address (*pc*)
- The register pointer address (*rp*)
- The address of *g7*
- The *cpu id* number
- The stack pointer (*sp*) address

### *Information Gathering – Summary for Using strings*

The following summarizes the steps used with the example. There is not a strict sequence to the use of the steps. Usually, a combination of the commands `strings`, `adb`, and `crash` provides the most comprehensive approach.

The `strings` command can be used on the `vmcore` file. The following information can also be obtained using the `msgbuf` macro in `adb`.

```
# strings vmcore.0 | more
```

Search the file for the following information:

- The panic message
- The program counter address (*pc*)

---

## *Information Gathering – Summary for Using strings*

- The register pointer address (`rp`) which can be used with the `regs` macro.
- The address of `g7`, the register which contains the current thread.
- The CPU ID number
- The stack pointer (`sp`) address



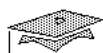
## Commonly Used crash Commands

- user
- stat
- proc
- cpu
- stack

### *Commonly Used crash Commands*

The crash debugger can also be used to view kernel information and core dump files. The most comprehensive way to approach core dump analysis is to combine features of both `adb` and `crash`. The most useful crash commands are

- |                                             |                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>user</code> (alias: <code>u</code> )  | Print the user structure for the designated process.                                                                                                                                                                                                                            |
| <code>stat</code>                           | Prints system statistics, including date and time of the crash, and register information.                                                                                                                                                                                       |
| <code>proc</code> (alias: <code>p</code> )  | Print the process table, similar to the listing obtained with the <code>ps</code> command. ( <code>adb</code> does not have a macro to do this.)                                                                                                                                |
| <code>cpu</code>                            | Display the CPU structure, which includes the current thread.                                                                                                                                                                                                                   |
| <code>stack</code> (alias: <code>s</code> ) | Dump the stack. The <code>-u</code> option prints the user stack. The <code>-k</code> option prints the kernel stack. If no arguments are entered, the kernel stack for the current thread is printed. Otherwise, the kernel stack for the currently running thread is printed. |



Sun Educational Services

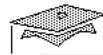
## Commonly Used crash Commands

- defthread
- defproc
- file
- inode
- kmastat
- help *command*

### *Commonly Used crash Commands*

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| defthread           | List the current thread.                                                      |
| defproc             | List the process slot of the currently active process.                        |
| file                | List the system open file table.                                              |
| inode               | List the system inode table.                                                  |
| kmastat             | Display how much memory is allocated to various kernel tables and structures. |
| ?                   | Invoke help, displaying a list of all available crash commands.               |
| help <i>command</i> | Display help for the specified command.                                       |

For more information about crash commands, refer to the man pages.



## Core Dump Analysis – Summary for Using crash

1. Get a status listing with the `stat` command.
2. List the current process with the `u(ser)` command.
3. Get a process table listing with the `proc` command.
4. Use `defthread` and `defproc` to obtain the addresses of the current thread and processes.
5. Enter a question mark (?) any time for help.

## *Core Dump Analysis – Summary for Using crash*

Use the crash debugger to perform the following steps:

1. Get a status listing with the `stat` command. This provides the program counter, the stack pointer, and general statistics.
2. List the current process with the `u(ser)` command. This provides the process slot number and information for the active process.
3. Get a process table listing with the `proc` command, and match the slot ID with the process slot listed with the `user` command.
4. Use `defthread` and `defproc` to obtain the addresses of the current thread and processes.
5. Use a question mark (?) for help.

## Core Dump Analysis Using crash

```
# crash vmcore.0 unix.0
dumpfile = vmcore.0, namelist = unix.0, outfile = stdout

> stat
system name:      SunOS
release:          5.7
node name:        mustang
version:          Generic
machine name:     sun4m
time of crash:    Fri Nov 29 09:04:19 1998
age of system:    18 min.
panicstr:         Data fault
panic registers:
    pc: f004c13c      sp: f0922808

> u
PER PROCESS USER AREA FOR PROCESS 34
PROCESS MISC:
    command: mkfs, psargs: mkfs /devices/pseudo/ramd@0:0,raw
512 8 1 8192 1024 16 10 60 2048 t 0 -1 8 -1
    start: Fri Nov 29 09:04:19 1998
    mem: 8a, type: exec
    vnode of current directory: f0bd8688
OPEN FILES, POFILE FLAGS, AND THREAD REFCNT:
    [0]: F 0xf06d3db8, 0, 0 [1]: F 0xf06d3db8, 0, 0
    [2]: F 0xf06d3db8, 0, 0 [3]: F 0xf06d3938, 0, 0
    [4]: F 0xf06d3ae8, 0, 0 [5]: F 0xf06d32a8, 0, 0
    [6]: F 0xf06d38d8, 0, 0 [9]: F 0xf06d3878, 0, 0
    [10]: F 0xf06d3848, 0, 0
    cmask: 0022
RESOURCE LIMITS:
    cpu time: unlimited/unlimited
    file size: unlimited/unlimited
    swap size: 2147479552/2147479552
    stack size: 8388608/2147479552
    coredump size: unlimited/unlimited
    file descriptors: 64/1024
    address space: unlimited/unlimited
SIGNAL DISPOSITION:
    1: default  2: default  3: default  4: default
    5: default  6: default  7: default  8: default
    9: default 10: default 11: default 12: default
    13: default 14: default 15: default 16: default
    17: default 18: default 19: default 20: default
```

## Core Dump Analysis Using crash

```

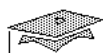
> proc
PROC TABLE SIZE = 4058
SLOT ST  PID  PPID  PGID  SID  UID PRI  NAME  FLAGS
  0 t    0    0    0    0    0  96 sched  load sys lock
  1 s    1    0    0    0    0  58 init   load
  2 s    2    0    0    0    0  98 pageout load sys lock nowait
  3 s    3    0    0    0    0  60 fsflush load sys lock nowait
  4 s   253    1  253  253    0  58 sac    load jctl
  5 s   147    1  147  147    0  48 inetd  load
  6 s   257   253  253  253    0  58 ttymon load jctl
  7 s   130    1  130  130    0  58 rpcbind load
  8 s   138    1  138  138    0  12 kerbd  load
  9 s   191    1  191  191    0  22 nscd    load
 10 s   150    1  150  150    0   0 statd  load
 11 s   132    1  132  132    0  12 keyser load
 12 s   122    1  122  122    0  58 in.routed load
 13 s   152    1  152  152    0  12 lockd  load
 14 s   254    1  254  254    0  48 sh     load
 15 s   171    1  171  171    0   4 automountd load
 16 s   175    1  175  175    0  58 syslogd load nowait
 17 s   185    1  185  185    0  56 cron   load
 18 s   209   201  201  201    0  44 lpNet  load nowait jctl
 19 s   201    1  201  201    0  33 lpsched load nowait
 20 s   229    1  229  229    0  58 vold   load jctl
 21 s   210    1  210  210    0   0 sendmail load jctl
 22 s   220    1  220  220    0  58 utmpd  load
 23 s   269   254  254  254    0  43 openwin load
 24 s   273   269  254  254    0  38 xinit  load
 25 s   274   273  274  254    0  59 Xsun   load
 26 s   275   273  275  254    0  55 sh     load
 27 s   280    1  275  254    0  59 fbconsole load
 28 s   361   318  318  318    0  44 newfs  load jctl
 29 s   286    1  275  254    0  59 vkbd   load
 30 s   291   147  147  147    0   0 rpc.ttdbserver load jctl
 31 s   289    1  275  254    0  59 ttsession load jctl
 32 s   293   275  275  254    0  59 olwm   load
 33 s   294   293  275  254    0  10 olwmslave load
 34 p   363   362  318  318    0   0 mkfs    load
 35 s   298    1  298  298    0  59 cmdtool load jctl

> defproc
Procslot=34

> defthread
Current Thread =

```





Sun Educational Services

## The ISCDA Document

The script `iscda` can be run to automatically provide some useful information after a system crash.

Sample usage

```
# cd /var/crash/mymachine
# iscda unix.0 vmcore.0 > /tmp/iscda.output
```

### *The ISCDA Document*

#### *Initial System Crash Dump Analysis*

The `iscda` script can be run to automatically provide some useful information after a system crash. As long as a core dump has been generated, this script will perform some data gathering that can be used to determine the cause of the crash. You may be able to do this at your site or you can forward the information to a Sun Solution Center support engineer for analysis. Using this script could shorten the resolution time for your problem by providing basic information before you place a call to Sun.

#### *Obtain the Script*

The `iscda` script is included on the SunSolve CD-ROM under the top-level directory `ISCDA` (or you can download it). Copy this script into a directory which contains various administrative commands. Be sure to change the permissions to allow execute permission and prevent unauthorized modifications.

## *The ISCDA Document*

### *Run the Script*

If your system panics or hangs, you can run the script once the system has rebooted and the core file is stored on disk. Redirect the output to a file. This output may be fairly long, especially if you have a large system that was manually aborted.

### *Sample Usage*

```
# cd /var/crash/mymachine  
# iscda unix.0 vmcore.0 > /tmp/iscda.output
```

Change to the directory you selected for crash dump storage (usually in `/var/crash`). Use the appropriate number for your particular core file instance, and redirect the output to a file of your choice. The output will consist of the results from a sequence of `adb` and `crash` commands. If needed, you can send this file to the Sun solution center support engineer handling your call via email.

## *Exercise: Performing Kernel Core Dump Analysis*



**Exercise objective** – The objective of this exercise is to perform an initial crash analysis and analyze a hung system. It familiarizes you with kernel structures that commonly need to be examined when analyzing a crash or a hung system.

### *Preparation*

The files needed for the sun4m labs are located in the `/opt/st350files` directory. Consult the instructor for access to these files. Refer to the examples in this module, the online header files, and the reference material at the end of this guide as needed while completing the labs.

For Part 1 of this exercise, select the appropriate version according to the architecture of your classroom system. No additional files are needed for the sun4u core dump analysis exercise in Part 1.

## Exercise: Performing Kernel Core Dump Analysis (sun4m) , Part 1

### Tasks

This part of the exercise duplicates the classroom example. Take your time, the exercise is self-paced.

1. Consult your instructor regarding driver installation. If needed, run the script called `ramdsetup`.
2. In a Shell Tool, change directories to `/devices/pseudo`.

```
# cd /devices/pseudo
# ls
```

If the RAM disk has been installed correctly, two entries are in this directory: `ramd@0:0`, and `ramd@0:0,raw`.

3. Run the `sync` command to make sure that the file systems are synchronized with the disk. Then run `newfs` to invoke the driver.

```
# sync; sync; newfs /devices/pseudo/ramd@0:0,raw
```

The system panics, saves the core dump, and reboots.

4. Change to `/var/crash/system_name`. Use `strings`, `adb`, and `crash` to analyze the problem. Follow the example beginning on page 7-29.
  - a. Find the failing instruction.
  - b. Find the failing process (or command).
  - c. Find the failing argument to the process or command.

## Exercise: Performing Kernel Core Dump Analysis (sun4m), Part 1

In these next steps, use adb commands to modify a live kernel.

1. Change directories to `/usr/kernel/drv`.

```
# cd /usr/kernel/drv
```

2. Ensure `test1` is executable as a script. Invoke `test1`. (This program reads and writes to the backseat pseudo device.)

```
# test1
```

3. Invoke `adb`.

```
# adb -kw /dev/ksyms /dev/mem  
physmem xxx <= (adb prints this out and returns a non-prompt.)
```

4. Type the following command to display a portion of the machine code that is used within the backseat program:

```
backseat_write,10/X
```

5. Type the following command to display the machine code in assembler syntax. You are going to replace the top instruction with `FFFFFFFF` (an illegal instruction format) or `00000000` (a real instruction in the wrong place).

```
backseat_write,10/i
```

6. Insert an error instruction of your choice in the live kernel code used by the backseat program. `/W` opens the location for writing `FFFFFFFF` or `00000000`.

```
backseat_write+20/W FFFFFFFF or 00000000
```

7. Press Control-d to exit `adb`.

8. Use the `sync` command several times, then invoke `test1` again. The system panics.

9. Analyze the core dump to locate the failing instruction, the faulting process, and arguments that were passed to the process.

## *Exercise: Performing Kernel Core Dump Analysis (sun4u), Part 1*

### *Tasks*

Complete these steps:

1. Invoke adb on your running kernel.

```
adb -kw /dev/ksyms /dev/mem
```

2. Inject a problem into the ksyms driver.

```
ksyms_open/20i  
ksyms_open+14/W 0  
ksyms_open/20i  
$q
```

3. Invoke the driver using the nm command, which causes a system panic.

```
/usr/ccs/bin/nm /dev/ksyms > symbol_file
```

4. After the system reboots, log in and change to your crash directory. Run strings on the vmcore file and redirect the output to a file. Use grep to locate the following fields, and note the address associated with each:

```
pc _____  
g7 _____  
rp _____
```

Search for the string TRAP, and notice the text message.

Search for string cpu, and note the CPU ID number. \_\_\_\_\_

---

**Note** – Depending on what the system captures, step 4 may not work. It may be necessary to run msgbuf in adb.

---

5. Invoke adb on your crash dump files.

```
# adb -k unix.x vmcore.x
```

## Exercise: Performing Kernel Core Dump Analysis (sun4u), Part 1

6. Perform a stack trace by typing  
**\$c**
7. Note the second argument to the `die()` routine which lists on your stack trace. This should be the same address as the `rp` value obtained in your `strings` output.

Second argument to `die()` \_\_\_\_\_

8. Use the address from step 7 to obtain a register dump, and note the address which lists under the `pc` header.

`second_arg_to_die`**\$<regs**

`pc` address \_\_\_\_\_

9. Dump out the faulting instruction. This line should include the string `ksyms_open+0x14: illtrap 0x0`.

`pc_address/ai`  
**ksyms\_open/40i**

10. Examine the thread that caused the panic.

**panic\_thread/K**

11. Use the thread address from the previous step and run the `thread` macro. Display `thread.h` in another window for a definition of all the fields of the display. Note the `procp` address.

`panic_thread_address`**\$<thread**

`procp` address \_\_\_\_\_

---

## *Exercise: Performing Kernel Core Dump Analysis (sun4u), Part 1*

12. Use the `procp` address to examine the `proc` structure, and note the `psargs` address. In another window, display `proc.h`.

```
procp_address$<proc2u
```

Failing process \_\_\_\_\_

13. Exit `adb`, and invoke the crash debugger.

```
crash vmcore.x unix.x
```

14. Enter the `u`(ser) command to view the faulting process. Then enter the `defproc` command and note the process slot number.

```
> u  
> defproc
```

Process slot number \_\_\_\_\_

15. Obtain a process table listing, and note the name of the process associated with the process slot number from step 13.

```
> p
```

Process name \_\_\_\_\_

16. Obtain the address of the thread that caused the panic. This should match the `g7` address used in step 10.

```
> defthread
```

Thread address \_\_\_\_\_

17. Enter the command `stat` and note general statistics.

```
> stat
```

18. Use the question mark (?) to obtain a help listing.



---

## *Exercise: Performing Kernel Core Dump Analysis, Part 2*

### *Analyzing a Hung System*

Refer to “Fault Worksheet #34 – Script Hangs the System” in Appendix D, which involves analysis of a hung system. You can complete this workshop individually or as a small group.

## *Exercise: Performing Kernel Core Dump Analysis, Part 3*

### *Modifying Kernel Parameters*

Complete Workshop #49 for this part of the exercise. You can work individually or in a group.

---

## *Exercise: Performing Kernel Core Dump Analysis, Part 4*

### *(Optional) Core Dump Analysis With kadb*

Complete these steps:

1. (Optional) Shut your system down and boot the system with the kadb flag.
2. Repeat part 1 of this exercise in the kadb environment. (See Appendix C for kadb information, if necessary.)

---

**Note** – In this mode, when the panic occurs, you will immediately enter the debugger. The keyboard interrupt routine usually displays the stack trace resulting from the use of Stop-a.

---

## *Exercise: Performing Kernel Core Dump Analysis*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

---

## *Check Your Progress*

Before continuing on, check that you are able to accomplish or answer the following:


- Describe how a process creates a system dump
- Configure a system to collect and store core files
- Differentiate a system panic condition from a system hang
- List the steps to perform an initial core dump analysis
- Use the adb and crash debuggers to analyze crash dumps

## *Think Beyond*

Now that you have some exposure to analyzing kernel core dumps and using the various debuggers, how do you correct the problems you have identified?

# *Fault Tracker Progress Chart*

---

A 

## *Team Members*

---

---

---





---

## *Fault Tracker Progress Chart*

| <b>Fault Number</b> | <b>Hardware/Software</b> | <b>Time</b> |
|---------------------|--------------------------|-------------|
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |
|                     |                          |             |

## *Fault Analysis Worksheet Template*

### *Analysis Phase*

#### *Initial Customer Description*

#### *Problem Statement*

#### *Resources*

#### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |
|                       |                                |                         |                          |
|                       |                                |                         |                          |
|                       |                                |                         |                          |

---

## *Fault Analysis Worksheet Template*

### Diagnostic Phase

#### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result</b> | <b>Verification</b> |
|------------------------|----------------|---------------|---------------------|
|                        |                |               |                     |
|                        |                |               |                     |
|                        |                |               |                     |
|                        |                |               |                     |
|                        |                |               |                     |

#### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |
|                     |                      |                      |



## Objectives

Upon completion of this appendix, you should be able to

- Troubleshoot a watchdog reset condition
- Use OBP commands to display register contents
- Locate virtual memory information using a debugger and OBP commands

## References



**Additional resources** – The following references can provide additional details on the topics discussed in this appendix:

- Drake, Chris and Kimberly Brown. *Panic!* SunSoft Press
- SunSolve documents
- *The SPARC Architecture Manual, V8 and V9*

## Troubleshooting Watchdog Resets

### *OBP Environment*

Watchdog resets bring the system immediately to the ok prompt in PROM. It is important to examine the system immediately before rebooting so as not to overwrite valuable information. The PROM variable `watchdog-reset?` by default is set to `false` for this reason.

The `obpsym` module should be loaded to maximize the amount of symbolic information available in the PROM environment. Without this driver module, information is displayed in an absolute fashion, almost strictly by addresses without including symbolic textual information.

To check if the `obpsym` module is loaded, use the `modinfo` command.

```
# modinfo | grep obpsym
```

To load the module from the command line, use `modload`.

```
# modload -p misc/obpsym
```

To load with each boot, enter the following line in `/etc/system`:

```
forceload: misc/obpsym
```

---

## Troubleshooting Watchdog Resets

### *OBP Register Commands*

The OBP commands that are common to the sun4m and sun4u architectures are:

- `.locals` – Displays the local CPU registers
- `.registers` – Dumps the registers of the current window, those in use at the time of the crash
- `ctrace` – Displays a stack trace, listing routines that were active when the system went down (obpsym module should be loaded)

### *OBP Register Commands – sun4u*

OBP register commands for sun4u systems include:

- `.pstate` – Formatted display of the process state register
- `.ver` – Formatted display of the version register
- `.ccr` – Formatted display of the *ccr* or cache control register
- `.trap-registers` – Display of trap-related registers

### *OBP Register Commands – sun4m*

OBP register commands for sun4m systems include

- `.psr` – Formatted display of the process status register
- `.fregisters` – Display of floating point registers

## *Procedure for Troubleshooting Watchdog Resets*

The following steps represent a general approach to apply when watchdogs occur:

1. If your system was booted with `kadb`, use the debugger commands discussed in Module 7 to perform an analysis. Then use `$q` to enter PROM, and perform OBP diagnostics.
2. If the system was not booted with `kadb`, perform OBP diagnostics immediately.
3. Record key values from OBP before proceeding (see the Module 4 lab exercise for which values to record.)
4. Attempt to force a core dump using `sync`. (This often fails, and there is no guaranty of integrity of the information the core dump creates.) If successful, analyze the core dump.
5. Boot into single-user mode with the `kadb` flag enabled. Compare the values from the OBP diagnostic session with the values on your running system.
6. If needed, submit the problem to Sun Service. Refer to SunSolve document number 14230, *System Crashes and How to Prepare for Analysis by Sun Service*.



---

## *Virtual Memory Overview*

Virtual memory on the Solaris operating system is composed of physical memory (all of the dynamic random access memory [DRAM] chips), anonymous memory (swap), and mappable file system space.

It represents a design scheme which is able to virtually address more memory than what is physically available. Benefits to this design include larger processes can be run, and usually a greater number of processes can simultaneously execute on the same system. In fact, it is possible to run a process whose address space exceeds what is strictly available through physical memory.

### *Page Faults*

Major side effects of virtual memory design is demand-based paging, page faulting, and virtual to physical address translation operations.

A subset of the entire address space for any given program is loaded at startup. Eventually, an instruction for which the page is not memory resident is reached; this is called a *page fault*.

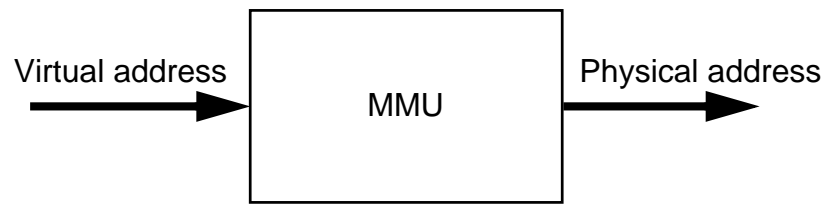
There are two categories of page faults:

- *Minor* – The page is loaded somewhere in memory and is reclaimed by a process.
- *Major* – The page is not memory resident and disk I/O must be performed to locate and load the page.

The architecture-independent areas of the operating system use the page structure as the unit of organization for virtual memory, which is described in `/usr/include/sys/page.h`.

## *The Memory Management Unit*

Another side effect of virtual memory design is the requirement for virtual to physical address translation. A hardware component called the memory management unit (MMU) is dedicated to this task. Its purpose is to translate the virtual address, generated by the executing code, to a physical address.



**Figure 2-1** Memory Management Unit

Architecture-dependent areas of the operating system, called the hardware address translation (hat) layer, include the code that controls the MMU. The page table entry structure (pte) maintains the information needed by the MMU to do its work:

- The physical address
- Reference, modify, and cache bits
- Access (protection bits)

While page faults are routinely handled by the kernel and normally require no intervention or analysis, it is possible for system faults to occur due to illegal address references, page protections, or page contents.

Another possible error condition can result if a page fault occurs in kernel space; this can cause a panic. Since most of the kernel is locked down and not subject to paging, page faults should not occur in this area.

## *Exercise: Analyzing Watchdog Resets and Virtual Memory*



**Exercise objective** – The objective of this exercise is to analyze a watchdog reset condition. Part 2 of the exercise provides the skills needed to locate virtual memory information using the debugger.

### *Preparation*

The workshop provides an approach to gather as much information as possible about a watchdog reset. You will install the bug that creates the watchdog reset.

You may work individually, with a partner, or in your workshop group for Part 1 and Part 2 of this exercise. Select the appropriate version of the labs depending on your classroom architecture.

## *Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1*

### *Tasks*

To analyze a watchdog reset for sun4u systems, complete these steps,

1. Ensure that the `obpsym` module is loaded. Enter the following line in `/etc/system`:

```
forceload: misc/obpsym
```

2. Shut down your system, and ensure that `NVRAM watchdog-reboot?` is `false`.
3. Boot the system to the Solaris operating environment.
4. Log in as `root`.
5. Invoke `adb` with the `-kw` qualifiers on your running kernel.

```
adb -kw /dev/ksyms /dev/mem
```

6. Use the following instruction to write an invalid value into the `sys_trap` routine:

```
sys_trap/W 0.
```

7. Wait for the watchdog error (usually the watchdog reset occurs immediately).

---

**Note** – If a watchdog error does not occur, ask for assistance.

---

---

## *Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1*

### *Tasks (Continued)*

8. At the OBP ok prompt, perform the following:
  - a. Note the number next to the ok prompt, which is the number of the CPU that hit the watchdog reset (MP only).
  - b. Note the information in the following fields:
    - `.registers` – Valid addresses associated with the window registers on the display. These are the registers that were current when the system went down.
    - `.locals` – Valid addresses associated with the registers on this display, which are the registers visible to the running process when the system went down.
    - `ctrace` – pc address and the names of any routines listed on the stack trace and the first couple of arguments listed next to routines.
    - `.ver` – The implementation (IMPL) and (MANUF) manufacturer numbers
    - `.trap-registers` – The trap type (TT), the state (TSTATE), and the processor state (PSTATE).
    - `.pstate` – The RED value, which is similar to the ET (enable trap) bit on SPARC Version 8.

---

## Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1

### Tasks (Continued)

9. Boot the system to the Solaris operating environment, and save the output of the following Solaris commands and files:
  - ▼ `showrev -p`
  - ▼ `prtconf -v`
  - ▼ `pkginfo`
  - ▼ `/usr/ccs/bin/nm /dev/ksyms > symbol_file`
  - ▼ `/usr/platform/sun4u/sbin/prtdiag -v > prtdiag_file`
  - ▼ `/etc/system`
  - ▼ `/var/adm/messages*`
10. Search the SunSolve database for the string *watchdog reset*. See if patch and bug reports exist for your system type. Also check the databases Symptoms and Resolutions, Frequently Asked Questions, Info Docs, and White Papers for related information.

Related document numbers in the SunSolve database include

- ▼ 1360 – “Troubleshooting Watchdog Resets”
- ▼ 14133 – “Is System Crash due to Hardware or Software?”
- ▼ 14230 – “System Crashes and How to Prepare for Analysis by Sun Service”

---

**Note** – You can set up a `tip` line to a system which has incurred a watchdog reset to facilitate saving the OBP command outputs to a file.

---

---

## Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1

### Tasks

To analyze a watchdog reset on a sun4m system, complete the following steps:

1. Check that the `obpsym` module is loaded. If not, enter the following line in `/etc/system`:

```
forceload: misc/obpsym
```

2. Shut your system down, and ensure that `NVRAM watchdog-reboot?` is false.
3. Boot the system to the Solaris operating environment.
4. Log in as `root` and invoke `adb` with the `-kw` qualifiers on your running kernel.

```
adb -kw /dev/ksyms /dev/mem
```

5. Use the following instruction to write an invalid value into the `sys_trap` routine:

```
sys_trap/W 0.
```

6. Wait for the watchdog error (usually the watchdog reset occurs immediately).

---

**Note** – If a watchdog error does not occur, ask for assistance.

---

---

## *Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1*

7. At the OBP ok prompt, note the values for the following fields:
  - ▼ `.registers` – Valid addresses associated with the window registers on the display, the number next to the ok prompt (which is the number of the CPU that hit the watchdog reset) and the address associated with `%g7`, the address of the current thread.
  - ▼ `.locals` – Valid addresses associated with the registers on this display, which are the registers visible to the running process when the system went down.
  - ▼ `ctrace` – The `pc` address and the names of any routines listed on the stack trace. Also note the first couple of arguments listed next to routines.
  - ▼ `.psr` – The processor interrupt level (PIL) and setting of the enable trap (ET) bit.
8. Boot the system to the Solaris operating environment, and save the output of the following Solaris commands:
  - ▼ `showrev -p`
  - ▼ `prtconf -v`
  - ▼ `pkginfo`
  - ▼ `/usr/ccs/bin/nm /dev/ksyms`



---

## Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 1

9. Copy the following files:

- ▼ /etc/system
- ▼ /var/adm/messages\*

10. Search in the SunSolve database for the string *watchdog reset*. Check if there are patch and/or bug reports related to your problem and your system type. Other databases to check include Symptoms and Resolution, Frequently Asked Questions, Info Docs and White Papers.

Related document numbers in the SunSolve database include:

- ▼ 1360 – “Troubleshooting Watchdog Resets”
- ▼ 14133 – “Is System Crash due to Hardware or Software?”
- ▼ 14230 – “System Crashes and How to Prepare for Analysis by Sun Service”

---

**Note** – You can set up a `tip` line to a system which has incurred a watchdog reset to facilitate saving the OBP command outputs to a file.

---

## *Exercise: Analyzing Watchdog Resets and Virtual Memory, Part 2*

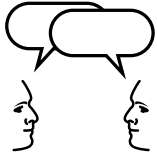
### *Locating Virtual Memory Information*

Complete Workshop #31 in Appendix D, which requires tracing virtual memory information using `adb` and `crash` commands. Select the appropriate architecture-dependent version of this workshop.

---

## *Exercise: Analyzing Watchdog Resets and Virtual Memory*

### *Exercise Summary*



**Discussion** – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
  
- Interpretations
  
- Conclusions
  
- Applications

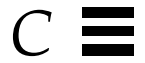
## *Check Your Progress*

Before continuing, check that you are able to accomplish or answer the following:

- Troubleshoot a watchdog reset condition
- Use OBP commands to display register contents
- Locate virtual memory information using a debugger and OBP commands

## *kadb and nmCommands*

---



This appendix contains brief descriptions of the `kadb` and `nm` commands. Various kernel structures are also discussed.

## *Invoking the kadb Debugger*

kadb uses a command interface which is identical to adb in most cases. It is invoked through the boot command, and allows the kernel to be run under the control of the debugger.

```
ok boot kadb
```

This command boots kadb, which loads the kernel. When running the kernel under the control of the debugger, the Stop-a sequence invokes the debugger, and the kadb[0]: prompt (not the ok prompt) is displayed the screen.

The operating system is suspended while in kadb. This environment allows examination of kernel structures in a static state and the setting of breakpoints in kernel routines.

To resume the kernel after working in kadb, enter the command :c at the kadb prompt. (It may be necessary to refresh the screen after returning from the kadb debugger.)

---

## *Introduction to Kernel Structures*

The exercise in this appendix enables you to become familiar with limited and carefully selected Solaris data structures related to crash analysis using the `kadb` utility.

Use the man pages and the header files to gain insight into the Solaris operating system and to increase your fault analysis skills with advanced concepts.

### *Using kadb*

`kadb` is an interactive debugger with a user interface similar to that of `adb`. `kadb` must be loaded prior to loading the kernel. It runs in the same address space as the kernel, but is not able to use the facilities available to the system (such as the mouse, and access to file systems) because the system is suspended when `kadb` is running.

Because the operating system is suspended when `kadb` is active, it is possible to examine the system in a static state. The keyboard facilities (Control-c, Control-s, and Control-q) are needed to control the screen display.

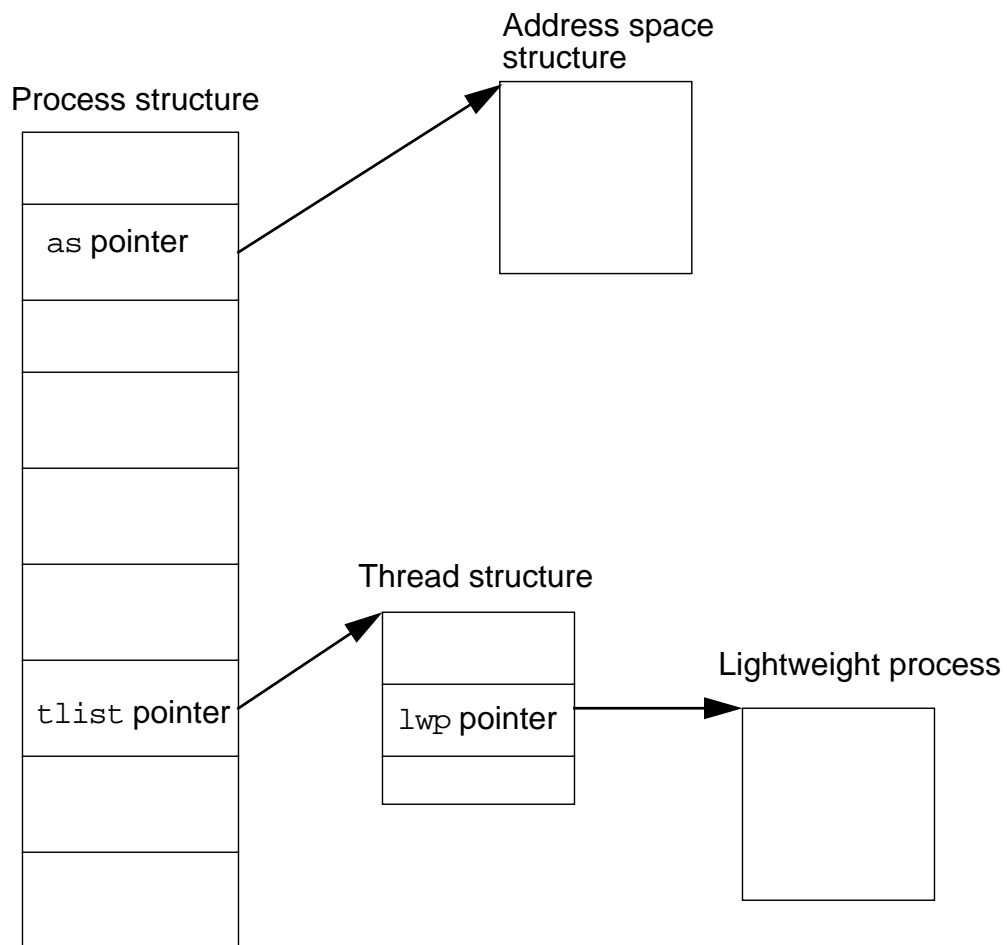
The keyboard abort sequence (Stop-a [L1-a]) suspends kernel operations and breaks into the debugger. If booted with `kadb`, the system enters `kadb` when it panics, allowing you to do an immediate analysis as to why the system went down.

The `kadb` debugger is useful when it is not possible to save a core dump, as is sometimes the case with a system hang, or if your dump device (swap device) is too small to save the crash dump file. It provides the only mechanism for setting breakpoints on a live kernel.

## Kernel Core Dump Analysis With kadb

The process address space is comprised of a number of structures. Those structures that are important in crash analysis are discussed here.

The address of the process structure is displayed in the ADDR field with `ps -el` output. This is your starting point for the lab exercise. Refer to the header files in `/usr/include/sys` as needed.



**Figure C-1** Process Address Space



## Exercise: Conducting Kernel Core Dump Analysis With `kadb`



**Exercise objective** – The objective of this exercise is to use `kadb` to analyze a kernel core dump

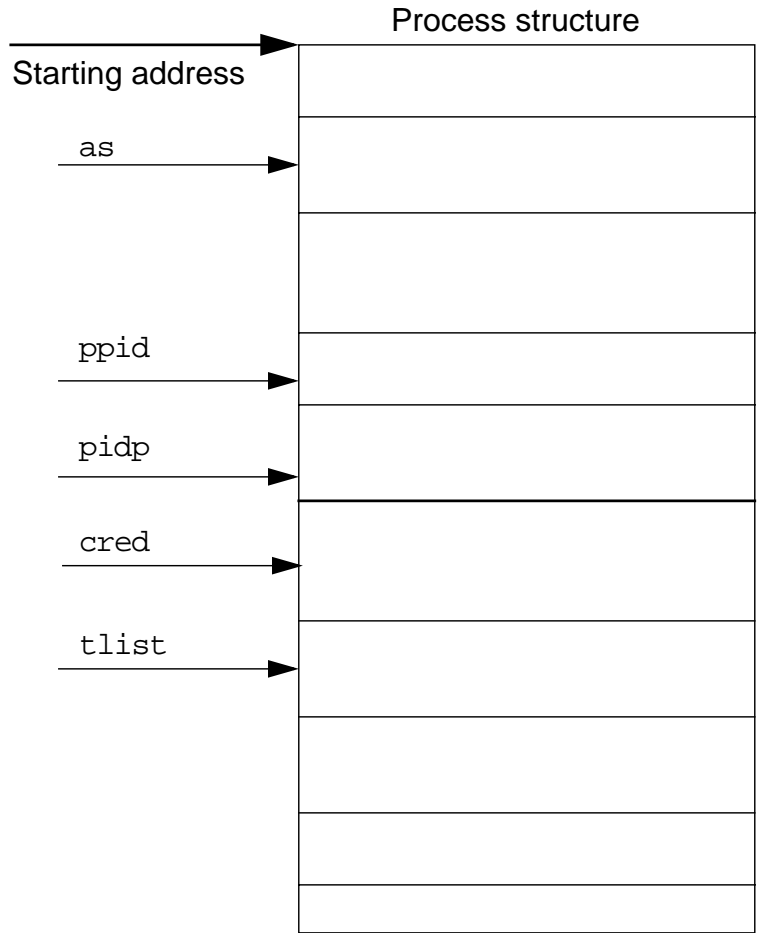
### Tasks

Complete the following steps to use `kadb` in kernel core dump analysis:

1. Boot the system with `kadb`.
2. Use `ps -e1` to obtain the starting address, `ADDR`, of any running process, for example `init`.
3. Enter the `kadb` debugger by pressing `Stop-a`.
4. View the `proc` structure and record the fields in Figure C-2.

```
kadb[0]: ps_ADDR_address $< proc
```

*Exercise: Conducting Kernel Core Dump Analysis With kadb*



**Figure C-2** Process Structure

---

## Exercise: Conducting Kernel Core Dump Analysis With `kadb`

### Tasks (Continued)

- Using the content of the process structure, you can trace other data structures to acquire further information about the process. For example, you can acquire information about the address, segment, thread, and credential structures. Use the addresses from the previous step to fill in the address variable in the following commands.

---

**Note** – The `as` and `seg` structures are defined in header files `as.h` and `seg.h`, which are in the `/usr/include/vm` directory. The `thread` and `cred` structures are defined in the header files `thread.h` and `cred.h` in the `/usr/include/sys` directory.

---

```
kadb[0]:as_address$<as
```

Note the number beneath the `nsegs` field: \_\_\_\_\_

```
kadb[0]:pidp_address$<pid
kadb[0]:tlist_address$<thread
kadb[0]:cred_address$<cred
```

- A process's address space is partly composed of segments. The virtual address space of a process contains different types of segments: text, data, stack, and other memory mapped objects such as regular files and device files.

Determine how many segments there are in your process.

```
kadb[0]: as_address $< as
kadb[0]: segs_address $< seg
kadb[0]: next_address $< seg
kadb[0]: next_address $< seg
```

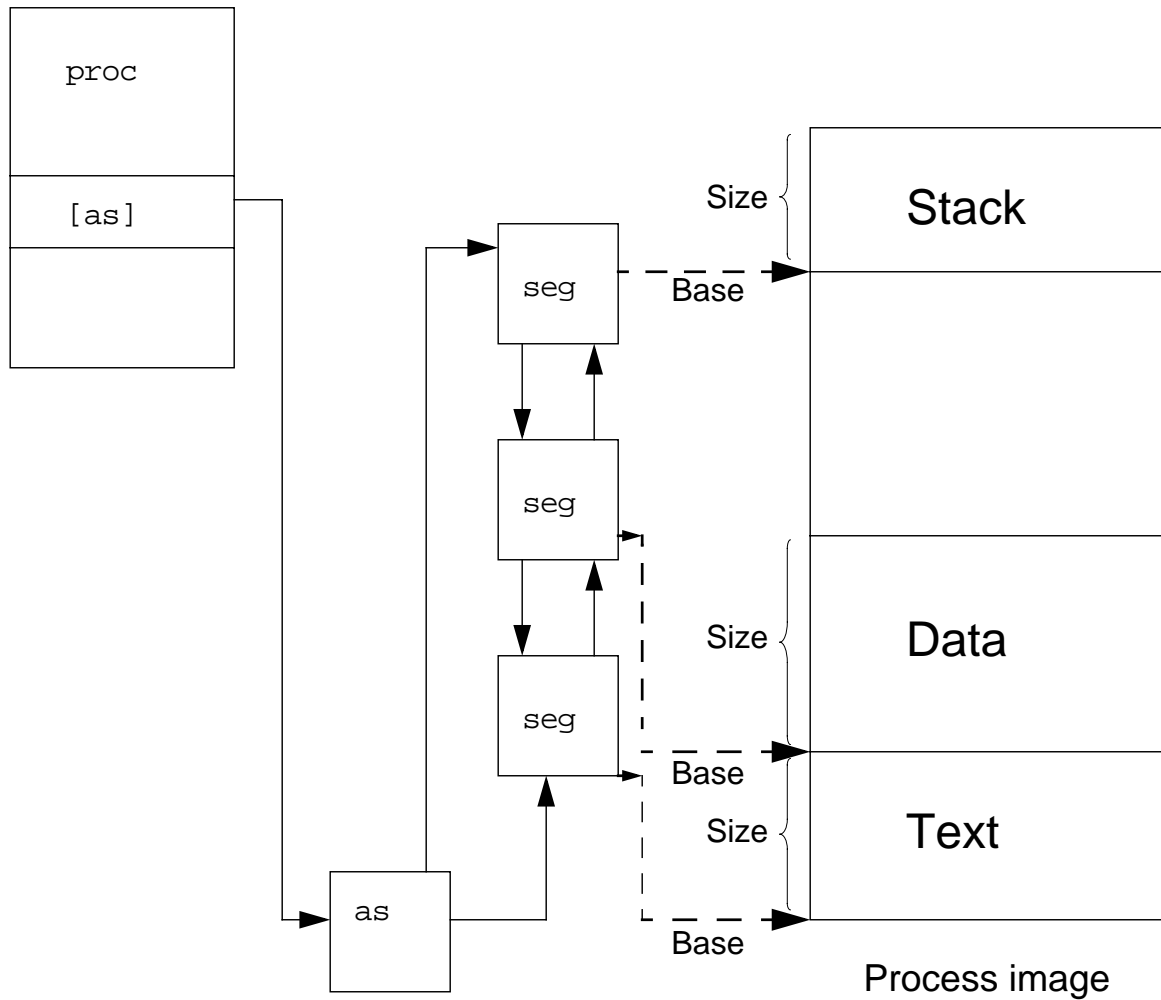
- Repeat the last command until the `next` field is populated with a zero, indicating you are at the end of segment list. Does the number of segments located with the `seg` macro match the number indicated by the `nsegs` field in step 5? \_\_\_\_\_

*Exercise: Conducting Kernel Core Dump Analysis With kadb*

*Tasks (Continued)*

8. Exit kadb, and reboot the machine.

```
kadb[0]: $q
ok boot
```



**Figure C-3** Segment Mapping

---

## Debugging Utilities

### *The nm Utility*

The `nm` utility displays symbol table information for each ELF (executable and linking format) executable file specified on the command line. Symbol table information includes variables, libraries, and header files which are part of the code for an executable.

In order to symbolically debug an executable, the symbol table information needs to be available. If a program has been stripped (see `strip` in section 1 of the man pages), then only limited symbolic debugging can be done.

The kernel is not stripped so that symbolic debugging can be done on a kernel image using the `adb` and `crash` utilities with `/dev/ksyms`, which invokes a driver that reads kernel symbol table information. Applications can also be debugged using the same tools, as long as they have not been stripped.

## Debugging Utilities

### The nm Command Output Display

```
# /usr/ccs/bin/nm /kernel/drv/options
```

```
/kernel/drv/options:
```

| [Index] | Value | Size | Type | Bind | Other | Shndx | Name             |
|---------|-------|------|------|------|-------|-------|------------------|
| [28]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | ddi_get_name     |
| [29]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | mod_driverops    |
| [30]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | mod_info         |
| [25]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | mod_install      |
| [20]    | 44    | 12   | OBJT | LOCL | 0     | 3     | modldrv          |
| [21]    | 56    | 20   | OBJT | LOCL | 0     | 3     | modlinkage       |
| [27]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | nodev            |
| [23]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | nulldev          |
| [1]     | 0     | 0    | FILE | LOCL | 0     | ABS   | options          |
| [15]    | 0     | 0    | FILE | LOCL | 0     | ABS   | options.c        |
| [18]    | 184   | 60   | FUNC | LOCL | 0     | 1     | options_attach   |
| [19]    | 244   | 36   | FUNC | LOCL | 0     | 1     | options_detach   |
| [22]    | 0     | 4    | OBJT | LOCL | 0     | 5     | options_devi     |
| [17]    | 132   | 52   | FUNC | LOCL | 0     | 1     | options_identify |
| [16]    | 52    | 80   | FUNC | LOCL | 0     | 1     | options_info     |
| [31]    | 0     | 44   | OBJT | GLOB | 0     | 3     | options_ops      |
| [32]    | 0     | 0    | NOTY | GLOB | 0     | UNDEF | strcmp           |

---

## Debugging Utilities

### The nm Column Headers

A brief description of each column of information that is provided with the nm command follows. Output can be varied by changing the options with which the program is run.

- The *Index* column displays a numeric value in brackets representing the index number assigned to the item by nm.
- The *Value* column displays a section offset, a virtual address, or alignment constraints which is dependent on the object type.
- The *Size* column displays the size in bytes for the object.
- The *Type* column indicates the symbol type, such as OBJECT, FUNC, SECTION, or FILE.
- The *Bind* column indicates the scope of the symbol, as in LOCAL or GLOBAL.
- The *Other* column is reserved for future use.
- The *Shndx* column usually lists the section header table index related to the symbol. Exceptions to this are ABS (absolute), COMMON, and UNDEF (undefined) symbols.

## Debugging Utilities

### The `nm` Command Output Display

The following example shows how to use `nm` to view kernel symbol table information. The use of `grep` with the command filters the output so that symbol table entries that contain the string “tmpfs” are displayed.

With no filtering, the output of `nm` with `/dev/ksyms` is very lengthy, and include all kernel variable names and tunable parameters, header files, libraries, and many function names.

```
# /usr/ccs/bin/nm /dev/ksyms | grep tmpfs
```

```
/dev/ksyms:
```

|         |            |     |      |      |   |     |                   |
|---------|------------|-----|------|------|---|-----|-------------------|
| [12539] | 4126248960 | 64  | FUNC | GLOB | 0 | ABS | tmpfs_hash_init   |
| [5917]  | 4126249280 | 276 | FUNC | LOCL | 0 | ABS | tmpfs_hash_lookup |
| [5916]  | 4126249164 | 116 | FUNC | LOCL | 0 | ABS | tmpfs_hash_out    |
| [12525] | 4126269732 | 4   | OBJT | GLOB | 0 | ABS | tmpfs_maxkmem     |
| [12532] | 4126269736 | 4   | OBJT | GLOB | 0 | ABS | tmpfs_minfree     |
| [5937]  | 4126303200 | 4   | OBJT | LOCL | 0 | ABS | tmpfs_minor       |
| [5939]  | 4126303208 | 4   | OBJT | LOCL | 0 | ABS | tmpfsfstype       |
| [5928]  | 4126255372 | 256 | FUNC | LOCL | 0 | ABS | tmpfsinit         |



## *Introduction*

Worksheet templates similar to those presented in Module 1 are provided in this appendix. In workshop groups, you will apply the fault analysis method discussed earlier in the course and record key points about your analysis and diagnosis for each problem.

There is no requirement to complete any particular number of the workshops. The *most* important thing is to apply a logical fault analysis method to those workshops you complete.

The worksheet templates are designed to coincide with procedures used in fault analysis. Every box inside each worksheet chart need not always be filled in. The amount of information that you record will vary with each problem.

## *Preliminary Task – Create a Student Account*

If a non-root account does not exist on your system, create one during your first workshop session on each system. The student account is required for some workshops, and is useful for comparison in others.

Try to employ all of the troubleshooting tools possible in your fault analysis workshops, and explore the use of new utilities. Your confidence will grow as you apply the fault analysis method to each problem and successfully bring it to solution.

## *Fault Worksheets – Student Guide*

This appendix contains the following worksheets:

- Fault Worksheet #1 – Blank Monitor
- Fault Worksheet #2 – Device Error During Boot
- Fault Worksheet #3 – File System Errors During Boot
- Fault Worksheet #4 – Incomplete Boot to Solaris Operating System
- Fault Worksheet #5 – Login Problem
- Fault Worksheet #6 – adb Macro Error
- Fault Worksheet #7 – Feckless
- Fault Worksheet #8 – System Hangs During Boot
- Fault Worksheet #9 – Turn the Page
- Fault Worksheet #10 – Login Problem
- Fault Worksheet #11 – Admintool Problem
- Fault Worksheet #12 – Common Desktop Environment Problem
- Fault Worksheet #13 – Shutdown During CDE Startup
- Fault Worksheet #14 – Network Printer Problem
- Fault Worksheet #15 – Boot Failure
- Fault Worksheet #16 – Constant Reboot Problem
- Fault Worksheet #17 – The ps Command Returns Nothing
- Fault Worksheet #18 – NIS or NIS+ Network Problem
- Fault Worksheet #19 – Network Problem
- Fault Worksheet #20 – No CDE Login Screen
- Fault Worksheet #21 – Banner Logo Has Been Changed
- Fault Worksheet #22 – Do Not Tread on Me
- Fault Worksheet #23 – vi Editor Problem

---

## *Fault Worksheets – Student Guide*

- Fault Worksheet #24 – Cracker Intrudes the System
- Fault Worksheet #25 – No Common Desktop Environment
- Fault Worksheet #26 – Login Problem
- Fault Worksheet #27 – ASCII Terminal Goes Blank
- Fault Worksheet #28 – No Network
- Fault Worksheet #29 – Where It Is At
- Fault Worksheet #30 – Faulty CD-ROM
- Fault Worksheet #31 – See It Now, SPARC 5 Example
- Fault Worksheet #31 – See It Now, Ultra Example
- Fault Worksheet #32 – Cannot Identify Root
- Fault Worksheet #33 – No Network or Interface
- Fault Worksheet #34 – Script Hangs the System
- Fault Worksheet #35 – No shcat
- Fault Worksheet #36 – Login Problem
- Fault Worksheet #37 – Client-Server ftp Problem
- Fault Worksheet #38 – Network Problem
- Fault Worksheet #39 – Slow and Fast Perceptions
- Fault Worksheet #40 – User Application Problems
- Fault Worksheet #41 – SunSolve Workshop
- Fault Worksheet #42 – Let Me In
- Fault Worksheet #43 – File Transfer Protocol Unavailable
- Fault Worksheet #44 – Slow NFS Server
- Fault Worksheet #45 – System Unavailable to Users

## *Fault Worksheets – Student Guide*

- Fault Worksheet #46 – Cannot Talk to Machine A
- Fault Worksheet #47 – Not on This Network
- Fault Worksheet #48 – Do Not Point At Me
- Fault Worksheet #49 – Resource Temporarily Unavailable
- Fault Worksheet #50 – Student Designed Workshop

---

## *Fault Worksheet #1 – Blank Monitor*

### *Analysis Phase*

#### *Initial Customer Description*

The administrator upgraded PROM on a system and customized some of the settings. Since then, the monitor remains blank when the system is turned on.

#### *Problem Statement*

#### *Resources*

#### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #1 – Blank Monitor*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #2 – Device Error During Boot*

### *Initial Customer Description*

Error messages occur when booting to the Solaris operating system.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #2 – Device Error During Boot*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result</b> | <b>Verification</b> |
|------------------------|----------------|---------------|---------------------|
|                        |                |               |                     |
|                        |                |               |                     |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |



---

## *Fault Worksheet #3 – File System Errors During Boot*

### *Initial Customer Description*

The boot sequence is incomplete due to apparent file system corruption after the last crash.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #3 – File System Errors During Boot*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #4 – Incomplete Boot to Solaris Operating System*

### *Initial Customer Description*

The default boot sequence appears to start correctly and then reports an unknown device. When the customer performs a boot -a and takes all default parameters, the system boots.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #4 – Incomplete Boot to Solaris Operating System*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #5 – Login Problem*

### *Initial Customer Description*

When logging in to `root`, an error message complains of an improper shell and immediately logs out.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #5 – Login Problem*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #6 – adb Macro Error*

### *Initial Customer Description*

Upon logging into root, the v macro (v\$<v) returns no parameters during an adb working session.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

*Fault Worksheet #6 – adb Macro Error*

*Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

*Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |



---

## *Fault Worksheet #7 – Feckless*

### *Initial Customer Description*

You cannot write to the directory /feck. The problem appeared during an attempt to create a directory called test and a file called my.test in the /feck directory.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

*Fault Worksheet #7 – Feckless*

*Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

*Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #8 – System Hangs During Boot*

### *Initial Customer Description*

The system administrator was tuning the system over the weekend and left for another tuning class in Dallas, Texas. You have been asked to restore the system. The problem is that the system hangs during boot sequence.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #8 – System Hangs During Boot*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #9 – Turn the Page*

### *Initial Customer Description*

The pg command does not work. The passwd command does not work. Only users with no password can log in.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #9 – Turn the Page*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #10 – Login Problem*

### *Initial Customer Description*

The root user cannot log in successfully. The login prompt and password (if required) are accepted, and it appears a login is starting, but then the system logs out. The system administrator has just come back from training, worked the weekend, and left on a holiday

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #10 – Login Problem*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |



---

## *Fault Worksheet #11 – Admintool Problem*

### *Initial Customer Description*

When opening Admintool and using the Browser Software option, customer gets error message stating that an incompatible OS release is being used.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #11 – Admintool Problem*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #12 – Common Desktop Environment Problem*

### *Initial Customer Description*

The CDE environment is unavailable. An administrator had been working with drivers on the system, and now only a direct log in to a shell is possible.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #12 – Common Desktop Environment Problem*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #13 – Shutdown During CDE Startup*

### *Initial Customer Description*

When the superuser logs into the CDE environment, the workstation shuts down. The problem seems only to occur in the root account.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #13 – Shutdown During CDE Startup*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |

---

## *Fault Worksheet #14 – Network Printer Problem*

### *Initial Customer Description*

The network printer has stopped printing.

### *Problem Statement*

### *Resources*

### *Problem Description*

| <b>Error Messages</b> | <b>Symptoms and Conditions</b> | <b>Relevant Changes</b> | <b>Comparative Facts</b> |
|-----------------------|--------------------------------|-------------------------|--------------------------|
|                       |                                |                         |                          |
|                       |                                |                         |                          |

## *Fault Worksheet #14 – Network Printer Problem*

### *Test and Verification*

| <b>Likely Cause(s)</b> | <b>Test(s)</b> | <b>Result(s)</b> | <b>Verification(s)</b> |
|------------------------|----------------|------------------|------------------------|
|                        |                |                  |                        |
|                        |                |                  |                        |

### *Corrective Action*

| <b>Final Repair</b> | <b>Communication</b> | <b>Documentation</b> |
|---------------------|----------------------|----------------------|
|                     |                      |                      |