

IRIXview™
User's Guide

Document Number 007-2824-002

CONTRIBUTORS

Written by Susan Thomas and Bill Tuthill

Production by Kirsten Pekarek

Engineering contributions by Bruce Johnson, Joe CaraDonna, Laurie Engle,
Jeff Heller, and Ralph Humphries.

St. Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower
image courtesy of Xavier Berenguer, Animatica.

© 1998, Silicon Graphics, Inc.— All Rights Reserved

The contents of this document may not be copied or duplicated in any form, in whole
or in part, without the prior written permission of Silicon Graphics, Inc.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure of the technical data contained in this document by
the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the
Rights in Technical Data and Computer Software clause at DFARS 52.227-7013
and/or in similar or successor clauses in the FAR, or in the DOD or NASA FAR
Supplement. Unpublished rights reserved under the Copyright Laws of the United
States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd.,
Mountain View, CA 94043-1389.

Silicon Graphics, REACT/Pro, IRIS, and IRIX are registered trademarks, and the
Silicon Graphics logo is a trademark, of Silicon Graphics, Inc.

WindView is a trademark of Wind River Systems, Inc.

Contents

	List of Figures	vii
	List of Tables	xi
	About This Guide	xiii
	Other Useful Books	xiv
	Style Conventions	xiv
	Product Support	xv
1.	About IRIXview	1
	What Is IRIXview?	1
	Using IRIXview: A Quick Look	2
	IRIXview Architecture	3
	High-Resolution Timestamp	3
	Host-Side Activities	4
	Starting IRIXview	4
	Invoking the irixview Command	4
	IRIXview Main Window	5
	Using Help	6
	Exiting IRIXview	6
	Other Sources of Information	6
2.	Collecting Event Data	7
	Using IRIXview to Collect Event Data	7
	Prerequisites for Collection	7
	Using the Target Window	8
	Troubleshooting Data Collection	10
	Using rtmon-client or par to Collect Event Data	11
	Using rtmon-client for Collection	12
	Using par for Collection	12
	Adding Timestamps to Your Program	13

- 3. Displaying Event Data 15**
 - Exploring the Graph Window 15
 - Buttons at the Top of the Graph 16
 - State Stipples and Event Icons 18
 - Timeline and Scrollbars 20
 - Opening Previously Collected Event Data 21
 - Using the Context View Graph 22
 - Context Legend Window 23
 - Summary of Mouse Clicks 24
 - Using the CPU View Graph 24
 - CPU Legend Window 25
 - Selecting Event Data 26
 - Showing a Time Interval 26
 - Selecting a Time Instant 27
 - Selecting a Time Interval 28
 - Canceling a Time Instant or a Time Interval 29
 - Selecting an Event 29
 - Examining Event Data 30
 - Using the Event Inspector 31
 - Specifying Time in the Event Inspector 32
 - Dumping and Inspecting Events 32
- 4. Inspecting and Analyzing Data 33**
 - IRIXview Events 33
 - Context Switch Events 35
 - Process State Transition Events 35
 - User-Generated Events 36
 - Analyzing Data 37
 - Example 1—How A Process Starts Executing 37
 - Example 2—Executing on a Multiprocessor System 40
 - Example 3—Beginning of an FRS Frame 41
 - Example 4—FRS Overrun Detected 43

- 5. **Event Dictionary** 45
 - Using the Event Dictionary 45
 - Event Dictionary 47
 - Interrupt Service Routine (ISR) 48
 - Interrupt Entry—Entry to ISR 48
 - Interrupt Exit—Exit From ISR 49
 - Interrupt Types 49
 - Signal 51
 - Signal Receive—Entry to Signal Handler 51
 - Signal Send—Send Signal to a Process 52
 - Processes 53
 - Fork Process—Spawn a Process 53
 - Exit Process—Delete a Process 53
 - Process Events 54
 - Unknown 55
 - Unknown Event—Unknown Event 55
 - User Event 55
 - DefaultUser—Display User-specified Event 55
- 6. **User Interface Reference** 59
 - About IRIXview Window 61
 - Context Graph Options 61
 - Data Format Options 62
 - Display Events/States Window 62
 - Event Inspector Window 65
 - Legend Window Icon 66
 - New Context Graph Menu 68
 - New CPU Graph Window 69
 - Open Event File Window 70
 - Pan Left/Pan Right Icons 72
 - Push/Pop/Exchange Icons 72
 - Quit Menu Choice 73

Save Event File Window	73
Scheduler Summary Window	74
Search Accelerator Icons	76
Search Window Icon	76
System Call Summary Window	78
Target Window	80
Time Units Menu Icon	81
View Options Window	81
Zoom In/Zoom Out Icons	83
Index	95

List of Figures

Figure 1-1	IRIXview Architecture	3
Figure 1-2	IRIXview Main Window	5
Figure 2-1	Target Window	8
Figure 2-2	User Event Symbol	13
Figure 3-1	Buttons at the Top of the Graph Window	16
Figure 3-2	Time Units Menu Icon	16
Figure 3-3	View Options Icon	16
Figure 3-4	Display Events Icon	16
Figure 3-5	Zoom In/Zoom Out Icons	17
Figure 3-6	Pan Left/Pan Right Icons	17
Figure 3-7	Push/Pop Icons	17
Figure 3-8	Exchange Icon	17
Figure 3-9	Search Window Icon	17
Figure 3-10	Search Accelerator Icons	17
Figure 3-11	Legend Window Icon	18
Figure 3-12	State Stipples and Event Icons in Middle of Context Graph	18
Figure 3-13	State Stipple	19
Figure 3-14	Event Icon	19
Figure 3-15	Interrupt Label	19
Figure 3-16	Process Label	19
Figure 3-17	Idle Thread Label	19
Figure 3-18	Timeline and Information at Bottom of Graph Window	20
Figure 3-19	Open Event File Window	21
Figure 3-20	Legend Window for Context View Graph	23
Figure 3-21	Legend Window for CPU View Graph	25
Figure 3-22	View Options Window	26
Figure 3-23	Time Instant and Interval	28

Figure 3-24	Selected Event With Timestamp	29
Figure 3-25	Search Window	30
Figure 3-26	Event Inspector Window	31
Figure 4-1	IntEnt Event Icon	33
Figure 4-2	Event Inspector	34
Figure 4-3	Current Content Line	35
Figure 4-4	User Event Icon	36
Figure 4-5	Single Processor Trace	37
Figure 4-6	Single Processor—Sub-Time Interval	38
Figure 4-7	Multiprocessor Trace	40
Figure 4-8	Beginning of an FRS Minor Frame	41
Figure 4-9	FRS Overrun Detected	43
Figure 5-1	Signal Receive Icon	45
Figure 5-2	Sample Event Dictionary Page	46
Figure 5-3	Interrupt Entry Icon	48
Figure 5-4	Interrupt Exit Icon	49
Figure 5-5	Signal Receive Icon	51
Figure 5-6	Signal Send Icon	52
Figure 5-7	Fork Process Icon	53
Figure 5-8	Exit Process Icon	53
Figure 5-9	Unknown Event Icon	55
Figure 5-10	Default User Icon	55
Figure 6-1	IRIXview Main Window	60
Figure 6-2	Vie Graph Icon Bar	60
Figure 6-3	About IRIXview Window	61
Figure 6-4	Context Graph Options Window	61
Figure 6-5	Display Events/States Icon	62
Figure 6-6	Display Events/States Window	63
Figure 6-7	Event Inspector Window	65
Figure 6-8	Legend Window Icon	66
Figure 6-9	Legend Windows (Context and CPU)	67
Figure 6-10	Context View Graph Window	68
Figure 6-11	CPU View Graph Window	69

Figure 6-12	Open Event File Window	70
Figure 6-13	Pan Left/Pan Right Icons	72
Figure 6-14	Push/Pop/Exchange Icons	72
Figure 6-15	Save Event File Window	73
Figure 6-16	Scheduler Summary Window	75
Figure 6-17	Search Accelerator Icons	76
Figure 6-18	Search Window Icon	76
Figure 6-19	Search Window	77
Figure 6-20	System Call Summary Window	79
Figure 6-21	Target Window	80
Figure 6-22	Time Units Menu Icon	81
Figure 6-23	View Control Window Icon	81
Figure 6-24	View Control Window	82
Figure 6-25	Zoom In/Zoom Out Icons	83

List of Tables

Table 2-1	Event Classes in Target Window	9
Table 3-1	Descriptions of Mouse Clicks	24
Table 4-1	Process State Representation Lines	36
Table 5-1	Information Collected for Interrupt Entry Events	48
Table 5-2	Information Collected for Interrupt Exit Events	49
Table 5-3	Interrupts Defined for IRIXview	49
Table 5-4	Information Collected for Signal Receive Events	51
Table 5-5	Information Collected for kill Events	52
Table 5-6	Information Collected for processCreate Events	53
Table 5-7	Information Collected for processDelete Events	53
Table 5-8	Process Event Icons	54
Table 5-9	Information Collected for Unknown Events	55
Table 5-10	Information Collected for defaultUser Events	55
Table 5-11	REACT/Pro Events and Event Numbers	56

About This Guide

This guide describes the IRIXview graphical analysis tool for IRIX. IRIXview (formerly WindView) allows developers to observe the instantaneous timing of the IRIX kernel and its interactions with applications. The following chapters are provided:

Chapter 1, "About IRIXview," provides an overview of the IRIXview program.

Chapter 2, "Collecting Event Data," presents simple and advanced procedures for collecting data for analysis.

Chapter 3, "Displaying Event Data," describes how to navigate the Context View and CPU View graph windows.

Chapter 4, "Inspecting and Analyzing Data," discusses how to examine and analyze event data.

Chapter 5, "Event Dictionary," documents the event symbols that you will see and their relation to system events.

Chapter 6, "User Interface Reference," provides an alphabetical reference to all IRIXview commands and icons.

Other Useful Books

The following books contain information useful to IRIX programmers.

- For a description of the support IRIX provides for real-time programs, see the *REACT/Pro Programmer's Guide*, part number 007-2499-xxx.
- For details of the architecture of the CPU, processor cache, processor bus, and virtual memory, see *MIPS R4000 Microprocessor User's Manual* by Joseph Heinrich, Prentice-Hall, 1993 (ISBN 0-13-105925-4); and more recently, the *MIPS R10000 Microprocessor User's Manual*, part number 007-2490-xxx.
- For details of many IRIX system facilities not covered in this book, see *Topics in IRIX Programming*, part number 007-2478-xxx, and the *MIPSpro Compiling and Performance Tuning Guide*, 007-2360-xxx.

Style Conventions

This guide follows these conventions:

- Variables are in *italics*. Replace variables with the appropriate string or value.
- Filenames, IRIX command names, and new or emphasized terms are in *italics*.
- Subroutine and function names are shown in **bold** font.
- System messages and displays are shown in `typewriter font`.
- User input is in **bold typewriter font**. For example, to start IRIXview, enter:

```
# irixview
```

This guide uses the standard convention for referring to entries in IRIX documentation: the entry name is followed by a section number in parentheses. For example, `rtmond(1)` refers to the online reference page for the *rtmond* command.

Note: The screen captures in this manual show the Motif window manager. Windows might look different with a different window manager.

Product Support

Silicon Graphics, Inc. provides a comprehensive product support and maintenance program for hardware and software products. For further information, please contact your service organization.

About IRIXview

This chapter covers the following topics:

- “What Is IRIXview?” defines concepts important for this product.
- “Starting IRIXview” on page 4 describes how to start running the product.
- “Other Sources of Information” on page 6 gives pointers to related documentation.
- “IRIXview Architecture” on page 3 explains how pieces of IRIXview fit together.

What Is IRIXview?

The IRIXview product is a logic analyzer for a software system. A software system (consisting of the IRIX operating system and your applications) involves complex interactions among processes, system objects, and interrupts. Interactions must occur within certain time constraints, often with resolutions of microseconds or finer.

Traditional tools for debugging and benchmarking software systems have included source code debuggers and run profilers. These types of tools can provide much useful information about a system. However, they provide only a static picture of very dynamic situations. What developers need is a facility for understanding these highly dynamic interactions in a visual way—a way to look “under the hood” of their software system. IRIXview can provide this facility.

IRIXview enables users to visualize the complex activities of their software system, making it possible to:

- understand system behavior
- detect race conditions, deadlocks, processor starvation, and other problems relating to process interaction
- determine application responsiveness and performance
- recognize cyclic patterns in the application

The interaction of the processes, objects, signals, and interrupts in a software system can result in context switches. The term *context switch* refers to an operation performed by the IRIX scheduler, which switches one thread of execution for another. IRIXview permits the user to collect, display, and inspect information on the state of each process in the system and the events that lead to a context switch.

To run IRIXview, the *irixview.sw.irixview* product image must be installed, and a FlexLM license must be available for the product. See <http://www.sgi.com/Support/Licensing/> for information about licensing.

Using IRIXview: A Quick Look

IRIXview lets you collect and analyze event data for target IRIX systems. Examples of events include process or thread creation and termination, system calls, and system interrupts, including scheduler clock ticks. IRIXview displays events by context or by CPU. A *context* is any thread that can execute on a processor, including user programs, system threads, interrupts, and the kernel's idle thread.

A typical work flow for using IRIXview includes the following steps:

1. Start IRIXview by entering the *irixview* command. See "Starting IRIXview" on page 4 for startup instructions and a description of the IRIXview main window.
2. Load event data. This is accomplished by collecting event data from a specified target system using IRIXview, or by opening a file with saved event data (such a file is called an event log) previously collected by IRIXview, *rtmon-client*, or *par*. See Chapter 2, "Collecting Event Data."
3. Open a view graph to display the event data. The graph shows you the event and state data of each context or CPU over a given time range. See Chapter 3, "Displaying Event Data."
4. Analyze the data. Zoom in on specific time periods to see the fine timing changes of events and states. Open the Event Inspector to examine text lists of events for a given time range, or drag and drop events into the Event Inspector window to look at detailed information. Using other windows, view summaries of system calls or scheduler activities. See Chapter 4, "Inspecting and Analyzing Data."

IRIXview Architecture

IRIXview consists of two components: one resides on the target (the system where events are being collected), and the other on the monitor host (the system running IRIXview). Although Figure 1-1 shows two systems, target and monitor are often the same system.

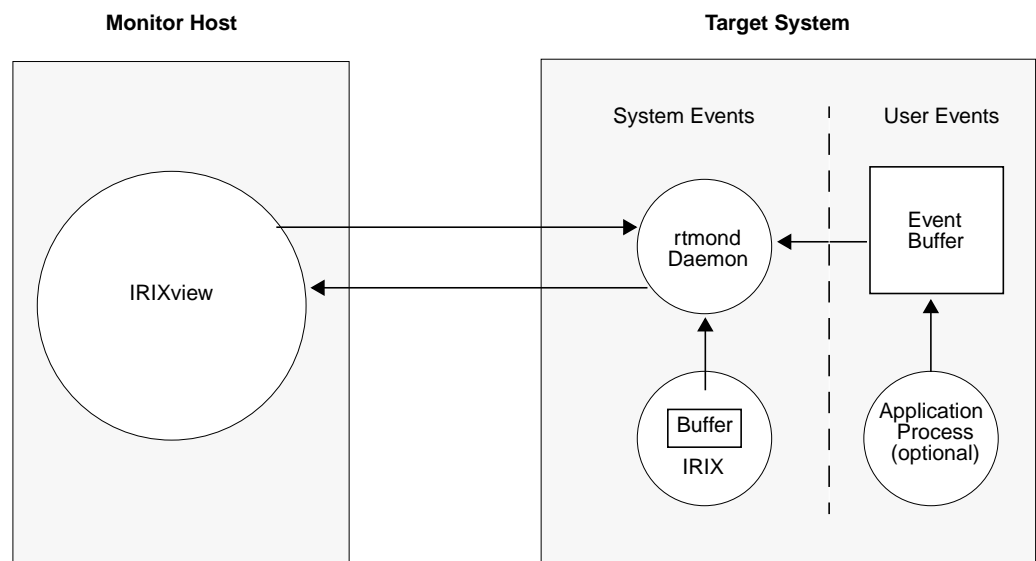


Figure 1-1 IRIXview Architecture

Events are logged to a buffer on the target system. When this buffer starts to fill up, the contents of the buffer are passed to the host by the *rtmond* daemon. The IRIX kernel's instrumentation is highly optimized and operates with minimal intrusion on the system.

High-Resolution Timestamp

When you start the event collection process (either using the Start button in the Target window, or with the *rtmon_client* command), the instrumented IRIX kernel tags events with a high-resolution *timestamp*. The events are then displayed in the Context View graph along a timeline showing when they occurred, based on these timestamps. You can see the exact timestamp for any event; for details, see "Selecting an Event" on page 29.

The timestamp driver's resolution is hardware dependent, but is always better than one microsecond resolution.

Host-Side Activities

The IRIXview host component (that is, the GUI) is an X application that runs as a normal user process under IRIX. It receives event data from the target system, processes that data, and displays pieces of it in a graphical format. The default target-to-host communication mechanism is TCP/IP.

Once event data has been collected, the information may be navigated and analyzed using the IRIXview Context View or CPU View graph windows. For details on using the Context Graph, see “Using the Context View Graph” on page 22. For details on using the CPU Graph, see “Using the CPU View Graph” on page 24.

In some situations you may find it more convenient and less intrusive to collect the event data in a file on the target host, for later analysis and display. For example, you may wish to collect data at a remote site for later analysis at a computer laboratory. This may be accomplished with the *rtmon-client* or *par* commands, described in “Using *rtmon-client* or *par* to Collect Event Data” on page 11.

Starting IRIXview

This section shows how to start *irixview*, describes the main window menus, including help facilities, and tells how to exit the application.

Invoking the *irixview* Command

To start IRIXview, enter the *irixview* command from a shell window such as a *winterm*; see *winterm(1)*. This brings up the IRIXview main window, as shown in Figure 1-2.

```
% irixview
```



Figure 1-2 IRIXview Main Window

IRIXview Main Window

The IRIXview main window contains the following pulldown menus:

File	This menu contains the commands for opening, analyzing, and saving event logs. It also contains the Quit command, which exits IRIXview.
Windows	This menu brings up windows for selecting a target host and event data, and for examining event data using the Context Graph and CPU Graph windows. You can also request system call and scheduler summaries.
Options	This menu lets you set data format options, and brings up a Context Graph Options window. Options are dynamic based on the loaded file, and apply to all currently open Context View graphs.
Help	This menu brings up a Help contents window, or a window showing the IRIXview version and copyright.

Leave the main window open whenever you are running IRIXview—the message area provides feedback on most operations, and commands in the menu bar are useful for displaying and examining event data.

To use pulldown menus, you do not necessarily need to click the mouse. Keyboard shortcuts are available: press the Alt key and a letter underlined in the main menu bar; with the menu displayed, pick a command from it by typing the underlined letter. For example, to exit IRIXview, press Alt+F and with the File menu displayed, press **Q**.

Using Help

The Help Contents window offers a set of writeups on using IRIXview, including an introductory screen describing how to get started.

Aside from the introductory online help, the Legend windows provide useful reference for the many symbols displayed by IRIXview. See either “Context Legend Window” on page 23 or “CPU Legend Window” on page 25 for more information.

Exiting IRIXview

To exit IRIXview at any time, choose File > Quit. The IRIXview main window and all other IRIXview windows are removed from the screen.

Note: When you exit, IRIXview does not prompt you to save event data. To save before exiting, follow instructions in the section “Save Event File Window” on page 73.

Other Sources of Information

IRIXview offers a graphical interface to the data output from the *rtmond* server process, which collects system and user events. You can think of IRIXview as a way to visualize detailed system activity. IRIXview accomplishes some of the same things as *rtmon-client*, except IRIXview has many added features. For example, graphs allow you to display, select, and analyze event data. For more information, see the reference pages for the related programs *rtmond* and *rtmon-client*, *rtmond(1)* and *rtmon-client(1)*.

The *sar* command, a standard System V utility, offers overall system activity reporting. See *sar(1)* for more information. The *par* command, a custom IRIX utility, provides specific information about a set of specific processes. See *par(1)* for more information.

Collecting Event Data

An *event* is any action by a thread, process, or hardware component that could affect the state of the system. IRIXview lets you collect event data on a selected IRIX target system or open previously saved files containing event data. These saved files are called *event logs*. This chapter covers the following topics:

- “Using IRIXview to Collect Event Data”
- “Using rtmon-client or par to Collect Event Data” on page 11
- “Adding Timestamps to Your Program” on page 13

Using IRIXview to Collect Event Data

With IRIXview, you may collect event data using the Target window. This section tells you how, and contains the following topics:

- “Prerequisites for Collection”
- “Using the Target Window” on page 8
- “Troubleshooting Data Collection” on page 10

Prerequisites for Collection

To collect event data, the target system must be running the *rtmond* daemon; see *rtmond(1)*. This daemon is default-installed as part of the *oe.sw.perf* subsystem, but might need to be turned on with *chkconfig*; see *chkconfig(1M)*.

To collect event data from a remote host, you must edit the */etc/config/rtmond.options* file on that remote host and delete the **-a localhost** access security flag.

Using the Target Window

To open the Target window, choose Windows > Target from the IRIXview main menu. Figure 2-1 shows the Target window.

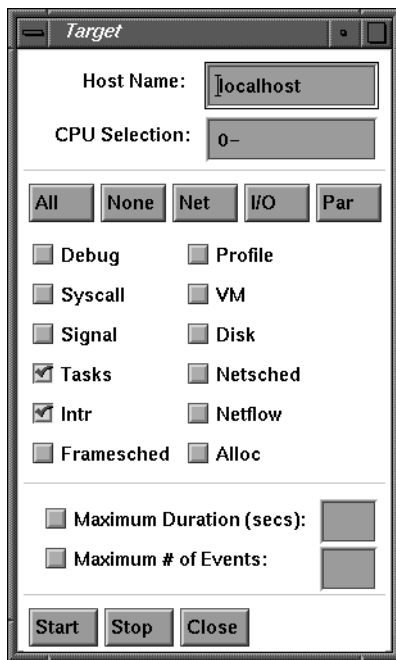


Figure 2-1 Target Window

Table 2-1 explains the event classes that can be checkmarked in the Target window.

Table 2-1 Event Classes in Target Window

Event Class	Explanation
Debug	Kernel debug events (not available on production systems).
Syscall	System calls, documented in section 2 of the reference pages.
Signal	Signal delivery and reception; see the signal(5) reference page.
Tasks	User process and thread scheduling.
Intr	Hardware interrupts; see the intr(D2) reference page.
Framesched	Frame Scheduler (FRS) operations; see the FRS(3) reference page.
Profile	Kernel profiling events (not available on production systems).
VM	Virtual memory operation.
Disk	I/O activity to and from disk drives.
Netsched	Network I/O scheduling
Netflow	Network I/O flow
Alloc	Memory allocation events; see the brk(2) reference page.

Follow these steps to collect event data using the Target window:

1. Type the target system's name in the Host Name field, or accept the *localhost* default. The default hostname is *localhost*, which automatically selects the system on which you started *irixview*.

Note: To determine a system's host name, on the target system enter either **hostname** or **uname -n**.

2. Enter the CPU for which you want to collect data, or use the default of **0-** to indicate all CPUs on the system specified in the Host Name field. You can separate multiple CPU numbers with commas; for example: **0, 3**. You can also enter a consecutive series of processors separated with a dash; for example: **1-3**.

3. Select which events you want to collect, or accept the default selections. The *All* button selects all of the event check boxes, and *None* removes all selections. Click *Signal*, *VM*, or *Disk* to automatically select all events related to such categories. The check boxes add events to the current selection.
Note: You must be superuser to collect Syscall events; otherwise, a warning message appears, the program deselects Syscall, and event logging continues.
4. Optionally, enter the maximum amount of time (in seconds) or number of events for which you want to collect data, then click Start. Data collection continues for the amount of time or number of events you specified. If you do not specify an amount of time or number of events, data collection continues until you click Stop.
5. It is a good idea to save the data collected during this event logging session. To do this, choose File > Save. You lose event data from the current session if you fail to save before you open an event file, collect new data, or exit IRIXview.
6. Now you can display the event data; see Chapter 3, "Displaying Event Data."

Troubleshooting Data Collection

Some problems you might encounter with data collection include:

- Host connection could not be established.
- Context View graph does not update.
- "Lost Event" messages appear in the IRIXview main window.

To collect event data, the *rtmond* daemon must be running on the target system; see *rtmond(1)*. Normally, this daemon starts automatically, but if you are having problems collecting data, make sure the daemon has actually started by running the *ps* command on the target system; see *ps(1)*.

When you are trying to collect data from a remote host, you must have been given permission to obtain event data. Ensure that the */etc/config/rtmond.options* file on the remote target system has been stripped of its **-a localhost** flag.

If you are accustomed to seeing the Context View graph update during data collection, please note that this does not occur in IRIXview, as it did in previous releases.

If you click the Start and Stop buttons in the Target window, but event data never appears in the graph window and the main window lists “Lost Event” error messages, the problem could be that the event rate exceeds the bandwidth of the connection, so the event logging mechanism shuts itself off. Try one or more of the following strategies to solve the problem:

- Close all graphs while you are collecting data.
- If you believe that network traffic may be the cause of the problem, isolate the host and target on a subnetwork or a standalone network.
- Use the *rtmon-client* tool to collect the event data; see “Using *rtmon-client* or *par* to Collect Event Data” on page 11.

Using *rtmon-client* or *par* to Collect Event Data

The *rtmon-client* tool offers a command-line alternative to IRIXview for event collection. The *rtmon-client* tool allows you to collect an event log on the host without loading it into IRIXview until some later time. For example:

- Your system generates a large amount of event data, and *rtmon-client* imposes less processor overhead than IRIXview. Because IRIXview must process event data before it can display the data, you might encounter target event buffer overflow conditions if your network is not fast enough to allow IRIXview to process events in real time (see “Troubleshooting Data Collection” on page 10).
- You want to collect event data at a remote site for later analysis at a lab. You can use *rtmon-client* to collect an event log as the remote system runs, and then import and view the event data with the IRIXview GUI at the lab.
- You want to save event logs to multiple files, each generated from a different operating condition. The *rtmon-client* tool provides an option that saves event logs into multiple files; each time event logging is turned on, a new file is generated.

System prerequisites are the same for *rtmon-client* and *par* as they are for the Target window. See “Prerequisites for Collection” on page 7 for details.

Using `rtmon-client` for Collection

To use `rtmon-client`, enter the following on the target system:

```
rtmon-client
```

By default, the collected event log is saved in the file `default`. You can change this name with the `-f` option. To stop `rtmon-client` event data collection, press `Ctrl+C`. Additional options specify the number of seconds to run (`-t sec`), a remote host to monitor (`-h name`), a processor list (`-p cpuList`), and whether to enable debugging (`-d 1`). For a description of `rtmon-client` and its options, see the `rtmon-client(1)` reference page.

As an example, to create 10-second traces for processor 1 through 3, and save them to the file `mp_test.irv`, enter the following:

```
rtmon_client -f mp_test -p 1-3 -t 10
```

This command creates three files, `mp_test.1.wvr`, `mp_test.2.wvr`, and `mp_test.3.wvr` for the three processors.

Using `par` for Collection

The `par` command is useful for collecting event data limited to a number of individual processes, rather than for the system as a whole. Here is an example `par` command:

```
par -s -SS -O /usr/tmp/test.irv -o /dev/null -p ProcessID -p 2ndProcessID
```

In this example, `-s` means collect system call and signal data, `-SS` means print both the summary of system call activity and a trace of each system call and signal action, `-O` outputs events to file `/usr/tmp/test.irv`, `-o` disposes of other output to `/dev/null`, and the two `-p` options indicate process IDs to trace.

The `par` command exits when all its given processes terminate, or you can press `Ctrl+C` to terminate `par` manually. For more information see the `par(1)` reference page.

Adding Timestamps to Your Program

Use the user timestamp logging function `rtmon_log_user_tstamp()` to insert timestamps into application code. This function logs events and passes them to the *rtmond* daemon. Such user events can be viewed in a Context View graph, shown by the symbol in Figure 2-2, and dragged and dropped into the Event Inspector window. For more information about implementing the timestamp logging function, see the reference page for `rtmon_log_user_tstamp(3)`.



Figure 2-2 User Event Symbol

User events are merged into the standard system event data stream (it is really the same stream), so user events appear in chronological order along with system events.

Displaying Event Data

This chapter covers the following topics:

- “Exploring the Graph Window”
- “Opening Previously Collected Event Data” on page 21
- “Using the Context View Graph” on page 22
- “Using the CPU View Graph” on page 24
- “Selecting Event Data” on page 26

Exploring the Graph Window

IRIXview provides two views for displaying an event log: a Context View graph and a CPU View graph. The Context View graph displays the state of all system contexts across time, while the CPU View graph shows the current run state (idle, user program, system thread, or interrupt) for each processor across time.

To display a graph, choose Windows > New Context Graph or > New CPU Graph from the IRIXview main menu.

This section details the elements common to both the Context Graph and the CPU Graph, including:

- “Buttons at the Top of the Graph” on page 16
- “State Stipples and Event Icons” on page 18
- “Timeline and Scrollbars” on page 20

For specific information on using either graph, see “Using the Context View Graph” on page 22 or “Using the CPU View Graph” on page 24.

Buttons at the Top of the Graph

The top of a Context or CPU View graph is shown in Figure 3-1. This section explains the function of icon buttons in the top row.

Figure 3-1 Buttons at the Top of the Graph Window



Figure 3-2
Time Units Menu Icon

Time Units Menu Icon

If you click and hold the left mouse button over this menu icon, you can choose the unit of time displayed in the Timeline and the Detailed Time Information field. The choices are:

- sec seconds (the default)
- msec milliseconds
- usec microseconds
- nsec nanoseconds



Figure 3-3
View Options Icon

View Options Window Icon

Clicking this icon displays the View Control window, which provides control over time intervals and zoom factors.

The resulting window is shown and described in the section “View Options Window” on page 81.



Figure 3-4
Display Events Icon

Display Events/States Window Icon

Clicking this icon displays the Display Events/States window, which provides control over which events or states to display in a View graph, and how event interrelations are shown.

The resulting window is shown and described in the section “Display Events/States Window” on page 62.



Figure 3-5
Zoom In/Zoom Out Icons

Zoom In/Zoom Out Icons

The Zoom In (capital Z) icon lets you focus on details; the Zoom Out (small z) icon lets you focus on the bigger picture.

Ordinarily Zoom In halves the time interval displayed, preserving the screen’s midpoint. If you have selected a time interval, Zoom In focuses on this time interval. For information on selecting a time interval, see the section “Selecting Event Data” on page 26.

Zoom Out doubles the time interval, retaining the midpoint if possible.



Figure 3-6
Pan Left/Pan Right Icons

Pan Left/Pan Right Icons

Clicking these icons moves the time interval one page to the left or right, where a page is defined as the width of the current time interval. Note that widening a View Graph window stretches but does not change the time interval.



Figure 3-7
Push/Pop Icons

Push/Pop Icons

The Push icon saves the current time interval. You can later move back to this time interval with the Pop or Exchange icon (see below). You can push up to 16 time intervals; if you push more than that, the oldest time intervals are discarded in FIFO order.

The Pop icon causes the most recently pushed time interval to be displayed (after clicking the Push icon).



Figure 3-8
Exchange Icon

Exchange Icon

This icon swaps the currently displayed time interval with the most recently pushed time interval. For example, find an interval that is of interest to you and save it with the Push icon. Move to another time interval of interest. Then click the Exchange icon repeatedly to move between that interval and the current interval. For a description of this icon, see “Push/Pop/Exchange Icons” on page 72.



Figure 3-9
Search Window Icon

Search Window Icon

Clicking the Search Window icon (uppercase “S”) displays a Search window, where you can search for events. The Search window is fully described in the section “Search Window Icon” on page 76.



Figure 3-10
Search Accelerator Icons

Search Accelerator Icons

Clicking on one of these icons finds the next or previous occurrence of the currently selected event. An event may be selected with the Search window, or by clicking on it with the middle mouse button.

The underlined arrows find the previous (or next) occurrence of the selected event in the same context, that is, in the same interrupt level, process, or idle thread context. The arrows without underlines search for the previous (or next) occurrence of the selected event, regardless of context. See “Search Accelerator Icons” on page 76 for details.



Figure 3-11
Legend Window Icon

Legend Window Icon

Clicking this icon displays the Legend window, which shows what each event icon and process state stipple indicates. The Legend window is different for the Context Graph and the CPU Graph. Details for each are explained in the sections “Using the Context View Graph” on page 22 and “Using the CPU View Graph” on page 24.

State Stipples and Event Icons

The Context View graph displays stipples and event icons, as shown in Figure 3-12.

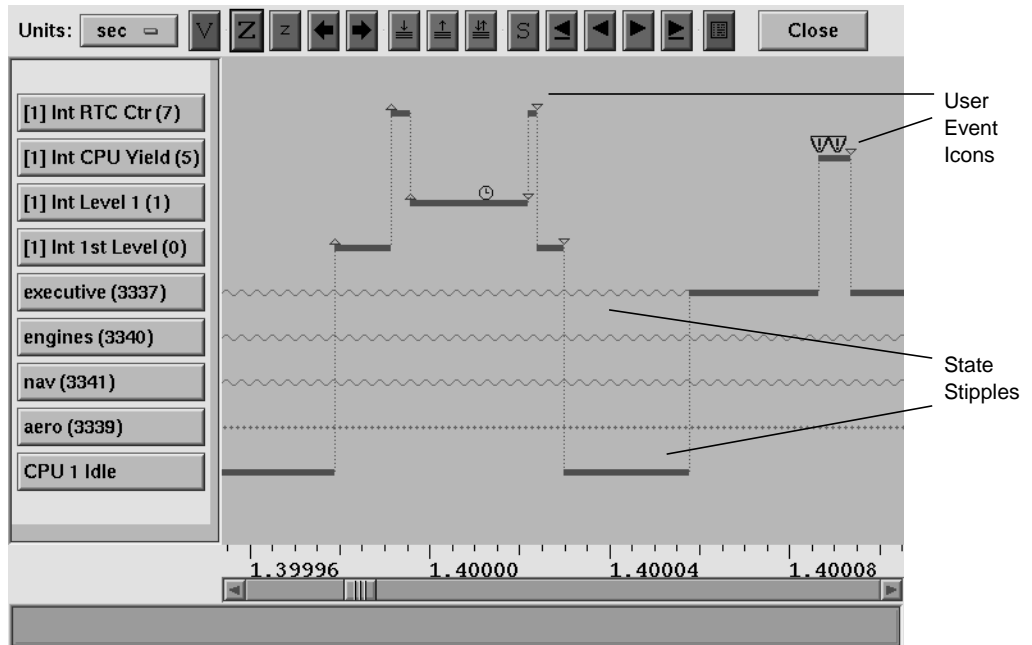


Figure 3-12 State Stipples and Event Icons in Middle of Context Graph

.....
Figure 3-13
 State Stipple



Figure 3-14
 Event Icon

State stipples are horizontal lines, while event icons are small symbols.

State Stipples These are horizontal lines that show the state of each process. The state stipple shown in Figure 3-13 is the Suspended stipple. For information about what each state stipple represents, see the Legend window, whose icon is shown in Figure 3-11.

Event Icons Depending on the event logging mode and the events you have filtered (see “Examining Event Data” on page 30), various icons are displayed that correspond to events. Figure 3-14 shows a Signal Receive event icon. For information on what event icons represent, see the Legend window. To learn specific information about the occurrence of an event icon, see “Examining Event Data” on page 30. For a complete list of event icons and their meanings, see Chapter 5, “Event Dictionary.”

Different types of state stipples and event icons are available in the Context View graph and in the CPU View graph. For a discussion of each, see “Using the Context View Graph” on page 22 and “Using the CPU View Graph” on page 24.

Notice that the sidebar of each graph contains labelled buttons. The sidebar of the CPU View graph shows each processor by number. The sidebar of the Context View graph contains the following items:



Figure 3-15
 Interrupt Label

Interrupts At the top of the Context Graph, interrupts occurring in this event log are listed. If you want more space above the interrupts to view icons that appear there, place the cursor over a labelled button and click the right mouse button. To remove the extra space, place the cursor over any button, press Shift and click the right mouse button again.

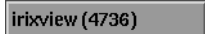


Figure 3-16
 Process Label

Processes After the interrupts, processes occurring in the event data are listed in order of priority. The first several items that look like processes are actually kernel interrupt threads.



Figure 3-17
 Idle Thread Label

Idle Thread After processes come the kernel’s CPU idle threads, listed by processor.

Timeline and Scrollbars

The bottom of the graph displays the timeline and a scrollbar to traverse the timeline. Time information also appears at the bottom of the window, as shown in Figure 3-18.

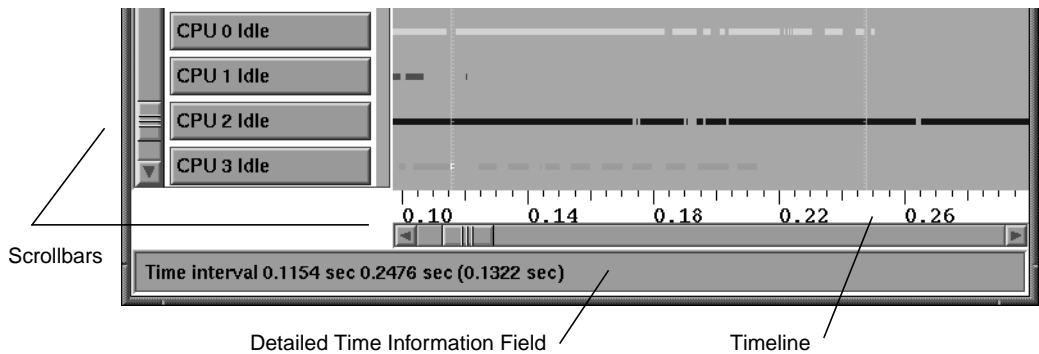


Figure 3-18 Timeline and Information at Bottom of Graph Window

Timeline The timeline displays the time in seconds since event logging began. You can change the units that are displayed using the Time Units Menu.

Scrollbars Use either scrollbar by dragging the scrollbar's *thumb* (the rectangle in the scrollbar), clicking in the *gutters* (the area on either side of the thumb), or clicking the arrows.

Detailed Time Information Field
Detailed time information about the current time instant, event, or sub-interval is displayed in this field. For example, click on an event icon with the middle mouse button and the timestamp of that event is displayed in this field. For more details, see "Using the Context View Graph" on page 22. You can change the units that are displayed by the Detailed Time Information field with the Time Units Menu.

Opening Previously Collected Event Data

If you have previously collected event data using *IRIXview*, *rtmon-client*, or *par*, you can open the *.irv* event data files by choosing File > Open from the IRIXview main window. The Open Event File window appears, as shown in Figure 3-19.

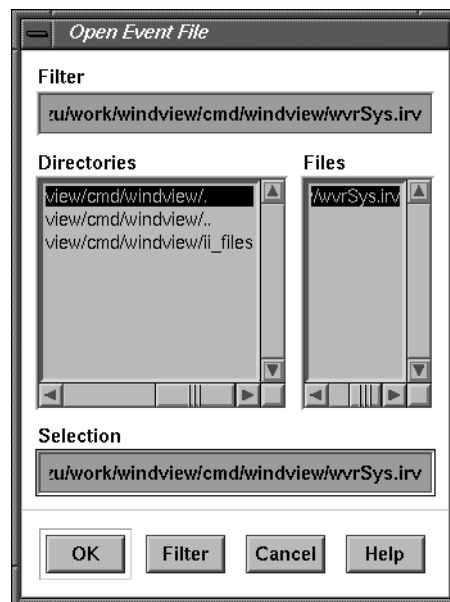


Figure 3-19 Open Event File Window

Type an appropriate collection directory name into the Filter field, then click Filter or press Enter to open that directory. If too many files are displayed in the Files subwindow, limit them by entering **.irv* into the Filter field. If the file that you wish to open is listed in the Files subwindow, double-click its name, or select it and click OK to open it.

The IRIXview main window indicates that the event log is open by listing how many events from the log have been read.

For information on lost events, see “Troubleshooting Data Collection” on page 10.

Using the Context View Graph

Use a Context View graph to examine the status and interaction of system events such as user timestamps, interrupts, active processes or threads, and processor idle threads. All system events, or just events you select for examination, are visible over the time period of data collected.

To use the Context View window, follow these general steps:

1. You must first load event data into IRIXview; see Chapter 2, "Collecting Event Data."
2. To open a Context View graph, choose Windows > New Context Graph in the main menu. You can have multiple Context View graphs open. They are numbered in the order opened.
3. If you want to set options for the view graph, choose Options > Context Graph in the main menu. See "Context Graph Options" on page 61 for more information.
4. Using mouse clicks, select a time interval for which you want detailed information; see "Selecting Event Data" on page 26.
5. In the main menu, choose Windows > Event Inspector to open the Event Inspector window. To get more details about events, click the Dump Context Events button. Events for the selected time interval appear in the Event Dump subwindow. See "Using the Event Inspector" on page 31 for more information.
6. Analyze the data. See "Analyzing Data" on page 37.

Context Legend Window

The Context View graph is full of graphic symbols. To understand what they mean, click the brown book icon to the left of the Close button inside a Context View window. This brings up the Legend window, as shown in Figure 3-20.

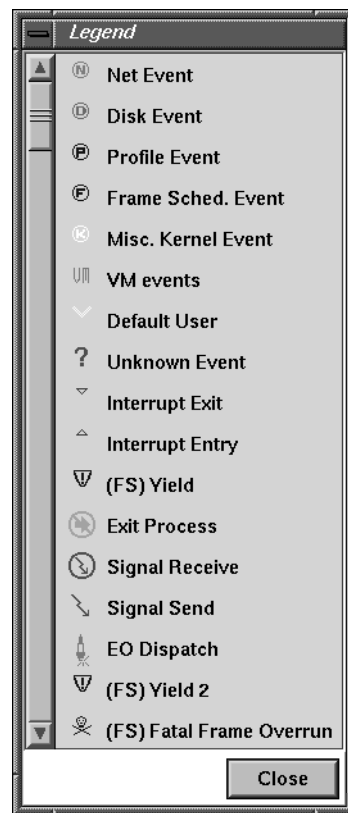


Figure 3-20 Legend Window for Context View Graph

Summary of Mouse Clicks

The various mouse clicks described in this chapter are summarized in Table 3-1.

Table 3-1 Descriptions of Mouse Clicks

Mouse Click	Description
Right button	Refresh screen
Left button	Select a time instant
Left button, move, left button	Select a sub-time interval
Shift+left button	Cancel a time or event selection
Middle button	Select an event
Ctrl+left button	Select a nearby event

Using the CPU View Graph

Use a CPU View graph to examine processor activity. This could be especially useful on multiprocessor systems. All CPU states are visible over the time period of data collected.

To use the CPU View window, follow these general steps:

1. You must first load event data into IRIXview; see Chapter 2, "Collecting Event Data."
2. To open a CPU View graph, choose Windows > New CPU Graph in the main menu. You can open multiple CPU View graphs. They are numbered in the order opened.
3. Using mouse clicks, select a time interval for which you want detailed information; see "Selecting Event Data" on page 26.
4. In the main menu, choose Windows > Event Inspector to open the Event Inspector window. To get more details about events, click the Dump Events for CPU button. Events for the selected time interval appear in the Event Dump subwindow. See "Using the Event Inspector" on page 31 for more information.
5. Analyze the data. See "Analyzing Data" on page 37.

CPU Legend Window

The CPU View graph is full of graphic symbols. To understand what they mean, click the brown book icon to the left of the Close button inside a CPU View window. This brings up the Legend window, as shown in Figure 3-21.

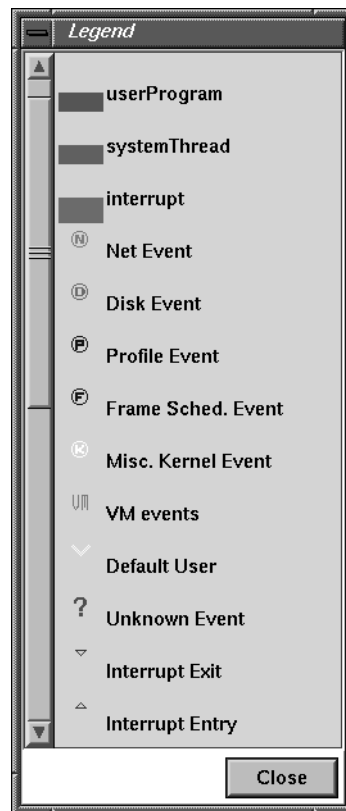


Figure 3-21 Legend Window for CPU View Graph

Selecting Event Data

Various types of event data may be selected, including time time instants, time intervals, and events. This section describes how to select event data, and contains the following sections:

- “Showing a Time Interval” on page 26
- “Selecting a Time Instant” on page 27
- “Selecting a Time Interval” on page 28
- “Canceling a Time Instant or a Time Interval” on page 29
- “Selecting an Event” on page 29

Showing a Time Interval

A time interval is that portion of the event log that is currently displayed in the graph. You can set a particular time interval using the View Options window. Click the “V” icon (see Figure 3-3) in a View graph window to display the View Options window. The CPU and Context View Options windows are similar; the latter is shown in Figure 3-22.

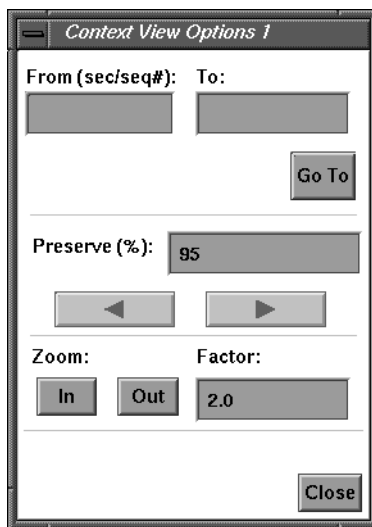


Figure 3-22 View Options Window

To use this window, follow these steps:

1. In the From and To fields, you can specify which time interval, in units of seconds or event sequence numbers, you would like to examine.

For example, type 1.0 in the From field and 2.0 in the To field, and then click the Go To button to view the interval from second 1.0 to second 2.0. Use integers to specify the range of event sequence numbers you want displayed; for example, from 1 to 10, or from 1500 to 2000.

2. The left and right arrow buttons act like the Pan Left and Pan Right icons on the graph window (see Figure 3-6), but are constrained by the Preserve (%) field.

For example, if Preserve is set to 50, the arrows move the view forward or back one-half page at a time (where a page is the width of the current time interval). However, if Preserve is set to 90, they move forward and back just 10% of the current time interval at a time. If Preserve is set to 0, they act the same as the icons on the graph window; if Preserve is 100, these arrows are disabled.

3. The Zoom In and Zoom Out buttons act like the zoom icons on the graph window (see Figure 3-5), but are constrained by the Factor field.

For example, if Factor is set to 10, Zoom In displays 1/10 of the current time interval and Zoom Out displays 10 times the current time interval. If Factor is set to 2, they act the same as the zoom icons on the graph window—Zoom In displays 1/2 the current time interval and Zoom Out displays 2 times the current interval. However, if Factor is set to less than 1, the actions of these zoom buttons are reversed.

4. Clicking the Display Events button brings up the Display Events/States window; for details see “Display Events/States Window” on page 62.

Tip: Before selecting time intervals and events, open the Event Inspector window, which displays useful information about the events and time intervals you select. See “Using the Event Inspector” on page 31 for more information.

Selecting a Time Instant

Select a time instant by clicking the left mouse button over a time of interest in the graph. The cursor must be in the graph window; it cannot be over the timeline, for example. A vertical line appears in the event log, and that time appears at the bottom of the window in the Detailed Time Information field (see Figure 3-20). The Event Inspector window also updates, if it is open.

Selecting a Time Interval

Select a sub-time interval by first selecting a time instant, then moving the cursor to another time of interest and clicking the left mouse button a second time. Two vertical lines appear in the event graph, and the times of each instant and the difference between them are displayed in the Detailed Time Information field. For example, Figure 3-23 shows time instants at 1.0484 and 1.7056 seconds with an interval of 0.6573 seconds. Selecting a time interval is one way to determine the amount of time that has occurred between two events.

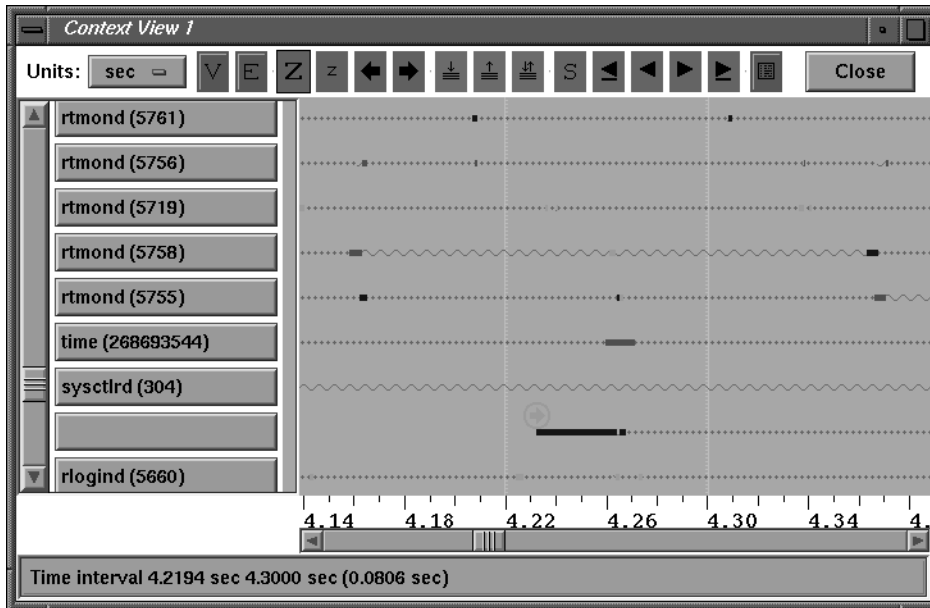


Figure 3-23 Time Instant and Interval

If you have the Event Inspector window open, the time fields automatically fill in. Once you have selected a time interval, you can zoom in on it by using the Zoom In icon (see Figure 3-5). Note that this deselects the time interval. You can return to the previous scale by using the Zoom Out icon, although without the time interval selection.

Canceling a Time Instant or a Time Interval

To cancel a time instant or time interval, with the mouse pointer anywhere in the graph subwindow, press the Shift key and click the left mouse button. Shift+left also cancels an event selection; see “Selecting an Event” on page 29.

Selecting an Event

To select an event, click the middle mouse button on an event icon. Its name, timestamp, and explanation appear in the Detailed Time Information field. For example, Figure 3-24 shows a selected interrupt event icon, with a timestamp of 1.2674 seconds.

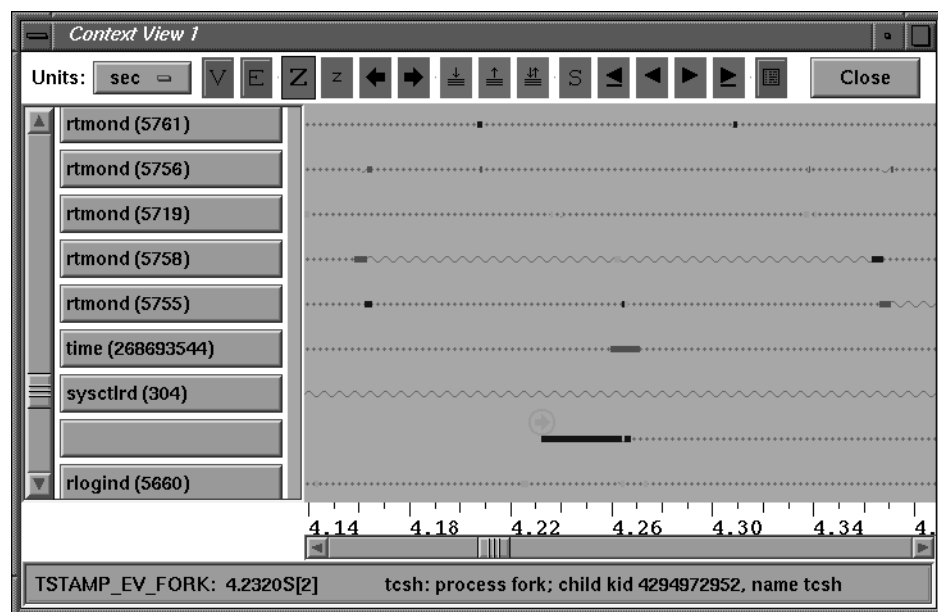


Figure 3-24 Selected Event With Timestamp

Another way to select an event is to search for it. Click the “S” icon (see Figure 3-9) in a View graph window. The Search window appears, as shown in Figure 3-25.

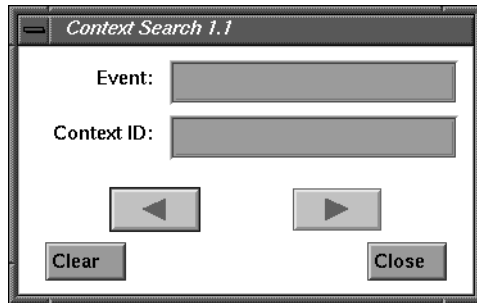


Figure 3-25 Search Window

To learn how to use the Search window, see section “Search Window Icon” on page 76.

Examining Event Data

An obvious way to examine the event data is to simply look at it. Use the scrollbars to scan the data as a whole, zooming in and out as appropriate with the Zoom icons. Even with this simple method you can gain useful information about your software system; for example, you can answer the following questions:

- Are deadlines being met?
- Are all application processes getting a chance to execute?
- Is my application blocking on a system call?
- Is too much time being spent in interrupts or in the idle thread?

Another way to examine event data is to use the methods described in “Selecting Event Data” on page 26 to select time intervals or events of interest, zoom in or out on them, and examine their timestamps and sub-intervals.

For a more detailed analysis of your software system, you can examine specific events using the Event Inspector. See “Using the Event Inspector” on page 31. See Chapter 5, “Event Dictionary” for a complete list of possible event icons and their meanings.

Using the Event Inspector

Use the Event Inspector to examine specific events. Choose **Windows > Event Inspector** from the main window. The Event Inspector window appears, as shown in Figure 3-26.

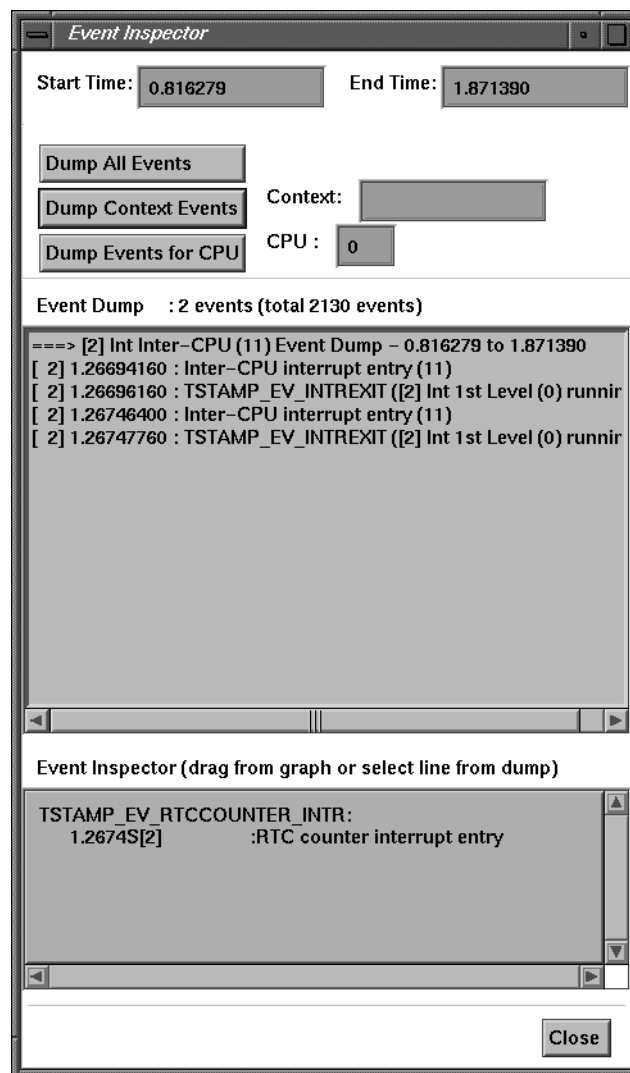


Figure 3-26 Event Inspector Window

To use the Event Inspector, follow these general steps:

1. Specify a start and end time for the data you want to view. Indicate a segment of time from the entire interval appearing in your View graph. See “Specifying Time in the Event Inspector” for details.
2. Dump events for inspection or drag and drop selected events from a View graph into the Event Inspector subwindow (at the bottom of the Event Inspector window). See “Dumping and Inspecting Events” for details.

Specifying Time in the Event Inspector

To select a series of events for a specific time interval, you can type the Start Time and End Time into the Event Inspector window. However, it is easier to select a time instant and interval using a View graph window. See “Selecting a Time Instant” and “Selecting a Time Interval” on page 28 for details. Clicking the mouse causes the start and end times to update in the Event Inspector window.

Dumping and Inspecting Events

To dump events using the Event Inspector window:

1. Set a start and end time. See “Specifying Time in the Event Inspector” on page 32.
2. To see information about events in the selected time interval, click Dump All Events. To see just context events, click Dump Context Events. To see just CPU events, click Dump Events for CPU. The results appear in the Event Dump panel.

To limit context events to a single context, type that context name into the Context field. To limit CPU events to a single processor, type its number into the CPU field.

3. If you select an event within the Event Dump panel, full event information appears in the Event Inspector panel below.

There are two ways to get details on a selected event:

- By clicking the mouse, select an item in the Event Dump panel.
- From a View graph window, select an event icon (see Figure 3-14) with the middle button and drag it to the Event Inspector panel (below the Event Dump panel).

See Chapter 5, “Event Dictionary” for a complete list of event icons and the information collected about an event.

Inspecting and Analyzing Data

This chapter includes the following sections:

- “IRIXview Events” provides an introduction to various types of events
- “Analyzing Data” on page 37 provides several analysis scenarios

IRIXview Events

In IRIXview, an *event* is any action by a thread, process, or hardware component that could affect the state of the software system. Examples of events are process creations and deletions, system clock ticks, and interrupts. IRIX has been instrumented to log this event information.

In the IRIXview Context View graph windows, time is represented on the horizontal axis, while the current system’s contexts are represented on the vertical axis:

- At the top of the list, system interrupts are listed, followed by interrupt threads.
- Below interrupts, all processes are listed in order of priority.
- The last contexts shown are idle threads for each processor, with the highest-level processor listed first.

When an event occurs, an *event icon* appears in the Context View graph. It is placed on the vertical axis according to the context in which it occurred, and on the horizontal axis according to the time (or sequence number) when it occurred.

You can drag an event icon into an Event Inspector window to see information associated with the event icon. For example, when a hardware interrupt occurred for which there is an associated ISR, an *IntEnt* event icon (see Figure 4-1) is displayed. By dragging the icon into the Event Inspector, you see information on that event: its timestamp, its context, the event name.



Figure 4-1
IntEnt Event Icon

Figure 4-2 shows a timestamp for the real-time clock (RTC) counter appearing in the Event Inspector window. For help using the Event Inspector, see Chapter 3, “Displaying Event Data.” For details about individual events, see Chapter 5, “Event Dictionary.”

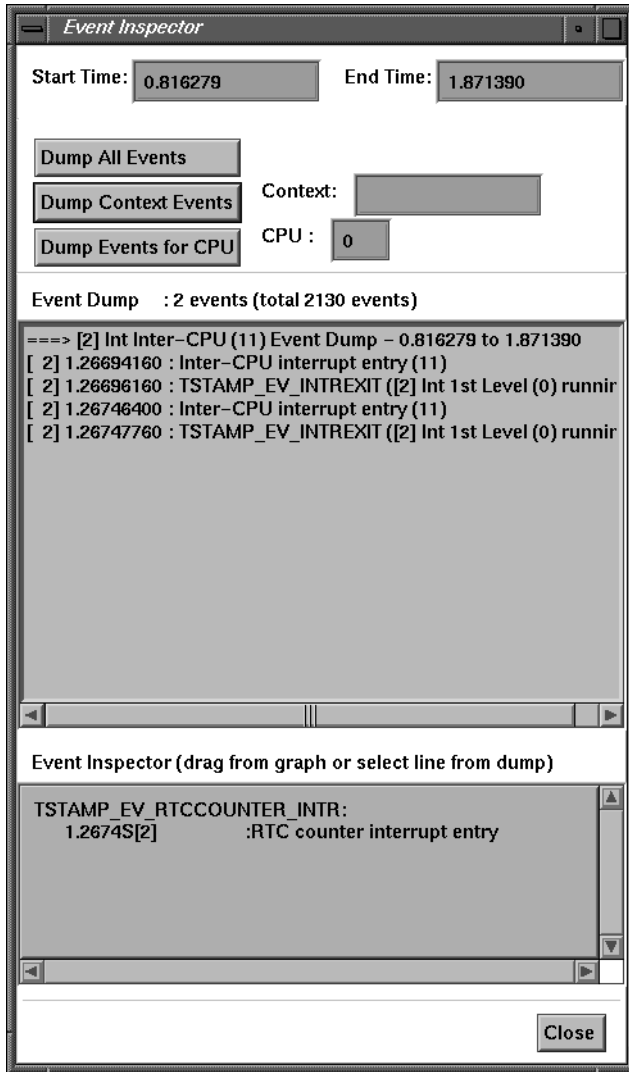


Figure 4-2 Event Inspector

The following sections describe features of event logging. These descriptions include information on how the GUI looks by default. You may customize many GUI attributes to meet your own needs; see Chapter 6, “User Interface Reference.” In addition, you can choose which events and states to display; see “Examining Event Data” on page 30.

Note: In this manual, a kernel mode switch, such as the switch the processor makes from idle mode to resume execution of a user process, is referred to as a context switch.

Context Switch Events

The term *current context* usually refers to the current process and the information needed to restore the process' condition, such as the state of the processor registers, operating system control information, and the stack. For IRIXview, the meaning of current context has been extended to include any thread of execution: a process, an ISR, or the kernel's idle thread. A *context switch* or *mode switch* refers to a change in the current context, which can occur when one process preempts another, when a process delays itself or waits on a resource, or when a process is interrupted by an ISR.

Figure 4-3

Current Content Line

When Context Switch events are logged, IRIXview shows the current context and where it is switched. The current context is shown as a solid, horizontal line (see Figure 4-3), with a different color used for each processor (on multiprocessor systems). When a context or mode switch occurs, and the Transition Lines button has been toggled on in the Display Event/States window, a dotted vertical line connects the previous context's line to the current context's line.




Process State Transition Events

The term *process state transition* refers to a process exiting from one state and entering into another; for example, from the pending state to the executing state. A process state transition may or may not result in a context switch, depending on the states of other processes in the system when the process in question makes a transition between states.

When Process State Transition events are logged, IRIXview shows the process state transitions and possibly the events that cause them. A process state is shown by the type of horizontal line (known as *state stipples*) used to display it. See Table 4-1 for a listing of the state stipples. State stipples are further differentiated by color. In addition, the event that caused the process state transition is shown as an icon.

As an optimization, events in Process State Transition are not separately timestamped. However, such events receive the timestamp of the next exit from the IRIX kernel. This exit is typically only a few microseconds after the event that originally caused the process state transition, and marks the moment at which the process state transition truly takes effect.

Table 4-1 Process State Representation Lines

Type of Line	Process State	Description
	Executing	The process, interrupt service routine (ISR), or idle thread has control of the processor.
	Suspended	The process attempted to gain access to a resource or event and the resource or event was not available (also referred to as a “blocked” process). On multiprocessor systems, if not all processors are being traced, a process can show as Suspended even after it has run, because process migration might occur.
	Ready	The process is not waiting for any resource other than the processor. That is, it is ready to execute, but has not yet been executed by the scheduler. On multiprocessor systems, if not all processors are traced, a process may display as Ready even after it has run, because process migration can occur.

Note: In earlier releases there was a diagonally striped stipple to indicate nondegrading real-time priority. You can still obtain priority levels from a Context View graph window by check marking the Priority button in the Context Graph Options window.

User-Generated Events



Figure 4-4
User Event Icon

When event logging has been started, IRIXview shows application-specific events. By default, these events are displayed by the User Event icon (see Figure 4-4) on the Context View and CPU View graphs. Event numbers are taken from a parameter provided with the `rtmon_log_user_tstamp()` function call; see `rtmon_log_user_tstamp(3)` for details. User-generated events are described in detail in the section “DefaultUser—Display User-specified Event” on page 55.

Analyzing Data

This section contains several examples of Context View windows showing how to analyze a progression of events as they are displayed in real time.

The first example shows events as IRIXview traces a single processor. The second example shows events as IRIXview traces a multiprocessor system. The third and fourth examples show IRIXview tracing a processor running the REACT/Pro Frame Scheduler. Callout numbers outside the figures point out important events that are described in nearby text (step 1 in the text describes what happens in the figure at callout 1).

Note: If scheduling actions are performed on an untraced processor, they do not display in the View graph. To completely follow the progress of the process, you must trace every processor on which the process can run.

Example 1—How A Process Starts Executing

Figure 4-5 shows a trace where a process starts executing on a single processor.

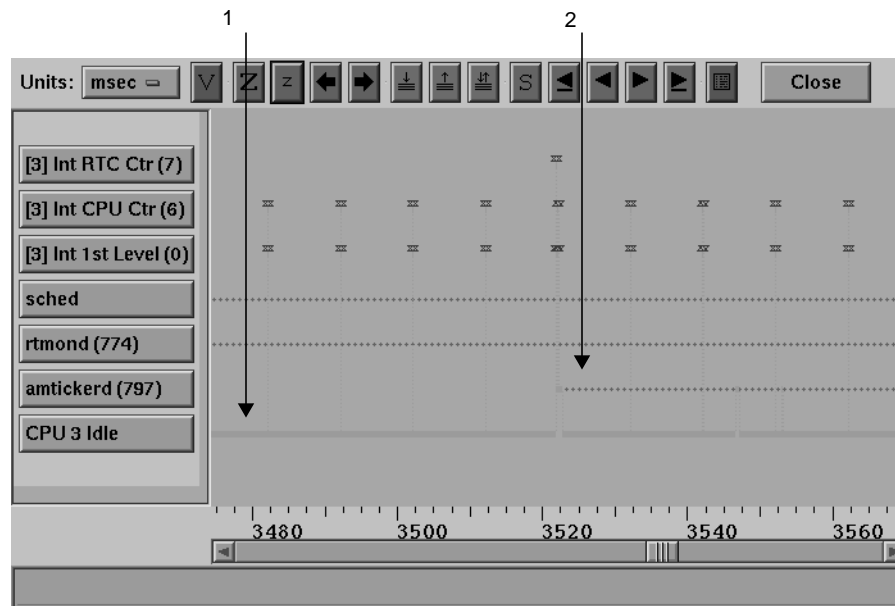


Figure 4-5 Single Processor Trace

Callouts are as follows:

1. At the beginning of this trace (at approximately 3475 microseconds into the trace) the CPU is running the idle thread (CPU 3 Idle), processes *sched* and *rtmond* are blocked, and process *amtickerd* has yet to begin execution on this CPU. The top three execution states on the graph, Int RTC Ctr(7), Int CPU Ctr(6), and Int 1st Level(0), are for identifying interrupts when they occur.

Notice that Int CPU Ctr(6), which is the processor scheduling clock interrupt, is occurring every 10 milliseconds, and that Int RTC Ctr(7), the real-time clock (RTC) interrupt, occurs just once in this image.

2. At approximately 3520 microseconds into the trace an event occurs that causes the *amtickerd* process to begin execution on CPU 3.

At this point, we want to examine in more detail the events that occurred which caused the *amtickerd* process to run. We zoom in on the graph around the 3520 microsecond time to see what is occurring in more detail, as shown in Figure 4-6.

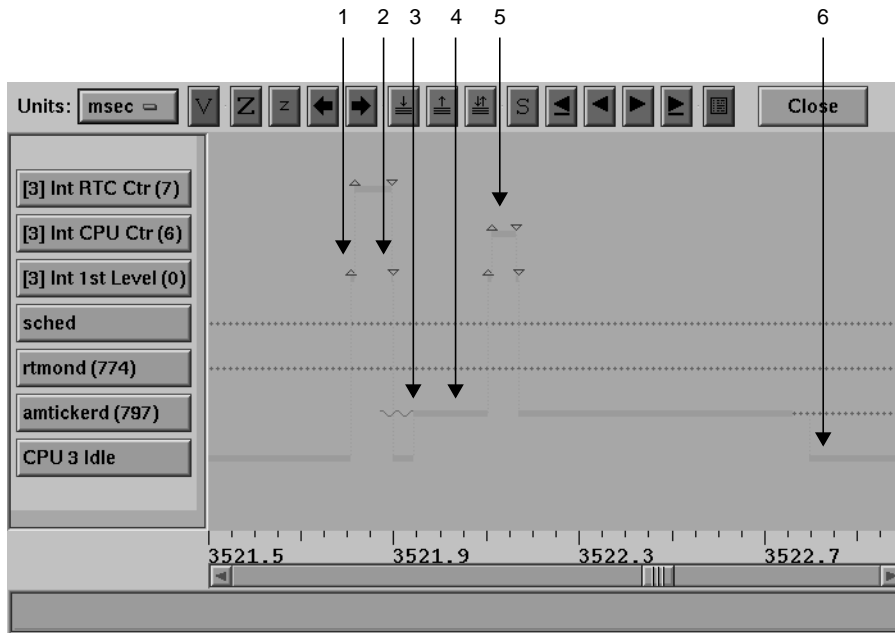


Figure 4-6 Single Processor—Sub-Time Interval

Callouts are as follows:

1. At approximately 3521.8 microseconds into the trace an interrupt occurs. The Context View graph shows the interrupt first being received and qualified through the Int 1st Level context—all interrupts pass through this context and are further qualified at this level.

Note: The processor is interrupted only once; the other interrupts that follow in this figure are simply further qualifications of this same processor interrupt.

2. The interrupt is qualified as a RTC interrupt, indicating that an event timeout has occurred and interrupt processing continues.
3. During processing of the RTC interrupt, the Interrupt Service Routine (ISR) initiates an event that permits the *amtickerd* process to become ready to run (the wavy line).
4. Upon exit from the RTC ISR, the IRIX scheduler determines which runnable process has the highest priority, and performs a context switch to that process. Since *amtickerd* is the highest priority runnable process, it transitions from ready-to-run into executing on CPU 3.
5. Less than a millisecond later the CPU processes a scheduling clock interrupt that apparently leads to no rescheduling, so the *amtickerd* process continues to execute.
6. Almost a millisecond later, the *amtickerd* process blocks and CPU 3 returns to the idle state.

Example 2—Executing on a Multiprocessor System

Figure 4-7 shows a multiprocessor system trace, showing how processes migrate between two processors as they execute.

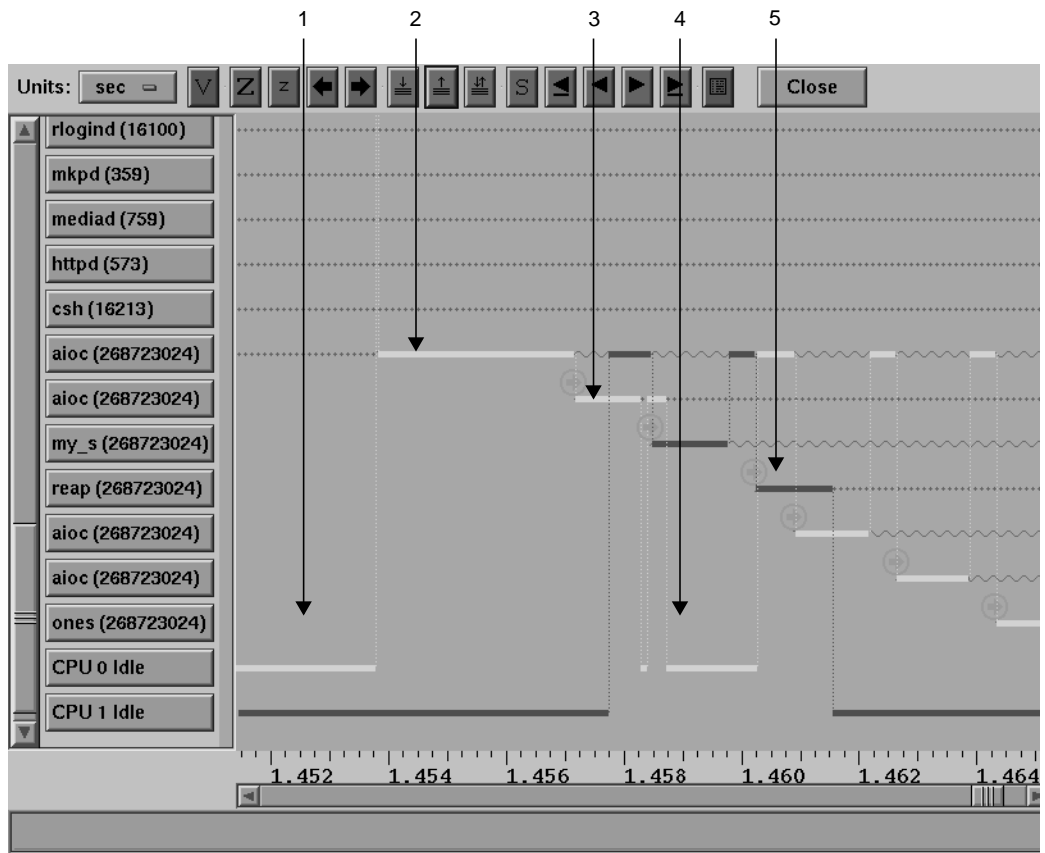


Figure 4-7 Multiprocessor Trace

Callouts are as follows:

1. At the start of the trace, both processors are executing the idle thread.
2. A few milliseconds later, the *aioc* process begins executing on CPU 0. As time progresses, the graph shows how this process migrates to and from CPU 1 as this processor becomes available. It then migrates back to CPU 0 later in the timeline.
3. A second invocation of *aioc* (a **pthread** application) initiates and begins execution on CPU 0. The arrow-in-circle icon indicates the startup of a new process.
4. CPU 0 is idle when there are no other processes to run. Due to processor affinity and other scheduling considerations, it is possible for there to be runnable processes and idle CPUs at the same time.
5. Other processes are spawned and execute on the two available processors.

Example 3—Beginning of an FRS Frame

Figure 4-8 shows the start of a Frame Scheduler (FRS) minor frame.

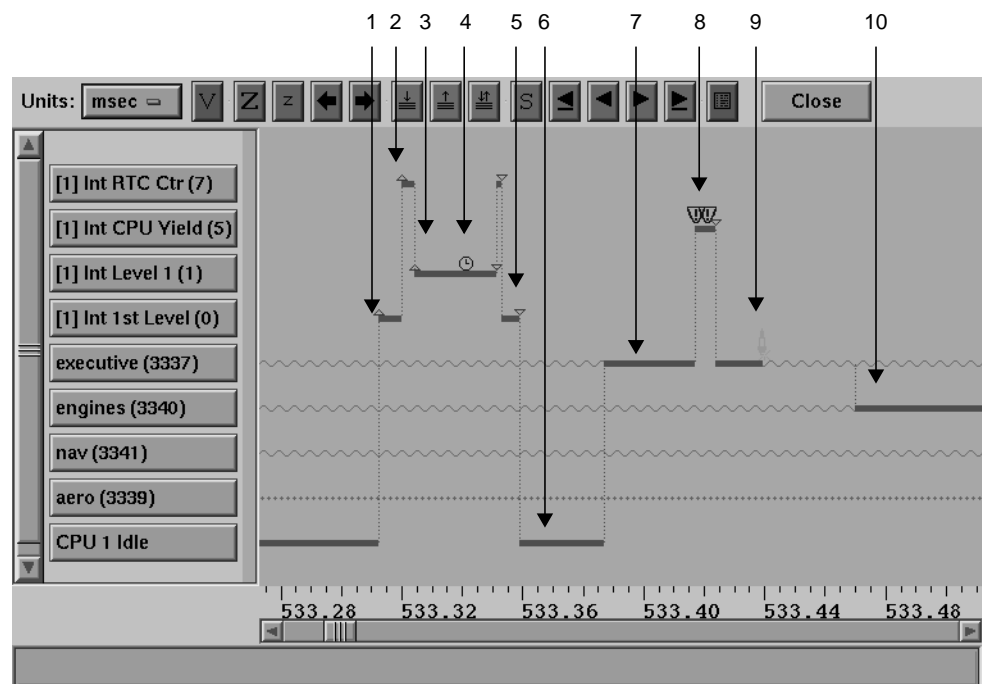


Figure 4-8 Beginning of an FRS Minor Frame

Callouts are as follows:

1. CPU 1 receives an interrupt and transitions from an idle state to the first level interrupt, Int 1st Level(0).
2. The interrupt is further qualified as an RTC interrupt, Int RTC Ctr.
Note: The processor is interrupted only once; the other interrupts that follow in this figure are simply further qualifications of this same processor interrupt.
3. The interrupt is further qualified as a Frame Scheduler (FRS) interrupt Int Level 1, signifying that the FRS has interrupted the CPU.
4. The small clock icon indicates that this FRS interrupt designates the start of a minor frame.
5. The interrupt exits the interrupt handler.
6. The idle thread immediately schedules the first frame-scheduled activity thread (*executive*) for that minor frame, and context switches to it.
7. The first FRS scheduled process (*executive*) begins execution.
8. The executive process completes execution for this minor frame and issues an **frs_yield** (shown in the yield-exclamation point icon).
9. The sparkplug icon indicates that another process is dispatched to run upon completion of the executive process.
10. A context switch occurs to the next process enqueued in this minor frame, *engines*.

Example 4—FRS Overrun Detected

This is an example of Frame Scheduler (FRS) overrun detection. Figure 4-9 show a minor frame overrun being detected.

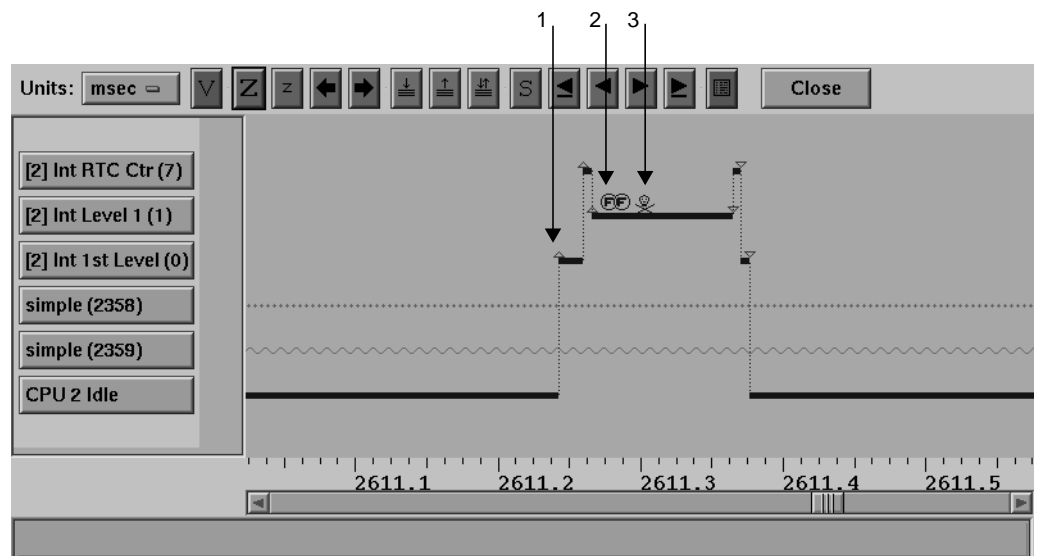


Figure 4-9 FRS Overrun Detected

Callouts are as follows:

1. A start-of-minor-frame interrupt occurs as was shown in the previous example.
2. In this case two additional frame scheduler events are shown indicating an anomaly has occurred (further information about these events could be obtained by using the Event Inspector window).
3. The skull and crossbones icon appears, indicating that the FRS has detected an unrecoverable frame overrun condition. In this case, the overrun is occurring because process *simple* did not complete its execution (reach its FRS yield) before the start of the next minor frame. Note that process *simple* (pid 2358) is blocked and has therefore not been able to continue processing during this frame.

Event Dictionary

This chapter provides the following sections:

- “Using the Event Dictionary” explains how to make sense of the event dictionary.
- “Event Dictionary” on page 47 shows various events that IRIXview collects.

Using the Event Dictionary

This section provides tips for understanding entries in the event dictionary. The event dictionary is a catalog of events collected by IRIXview, listing each event by object type, providing the following information for each one:

- the event’s icon
- possible causes of the event
- the possible process state effects that can result from the event
- information collected about the event (that is, information displayed when you drag the event’s icon into the Event Inspector window; see “Using the Event Inspector” on page 31 for details.)



Figure 5-1
Signal Receive Icon

Suppose you are viewing an event log with the IRIXview main window and you see the *Signal Receive* event icon, as shown in Figure 5-1.

1. Look in the Legend window (see “Context Legend Window” on page 23) to determine that this is the *Signal Receive* event icon.
2. Using the event dictionary in this chapter, look up the information on what can cause a *Signal Receive* event, what effect on process state it may have, and what information is collected for a *Signal Receive* event.

Figure 5-2 shows the structure of a typical entry in the event dictionary.

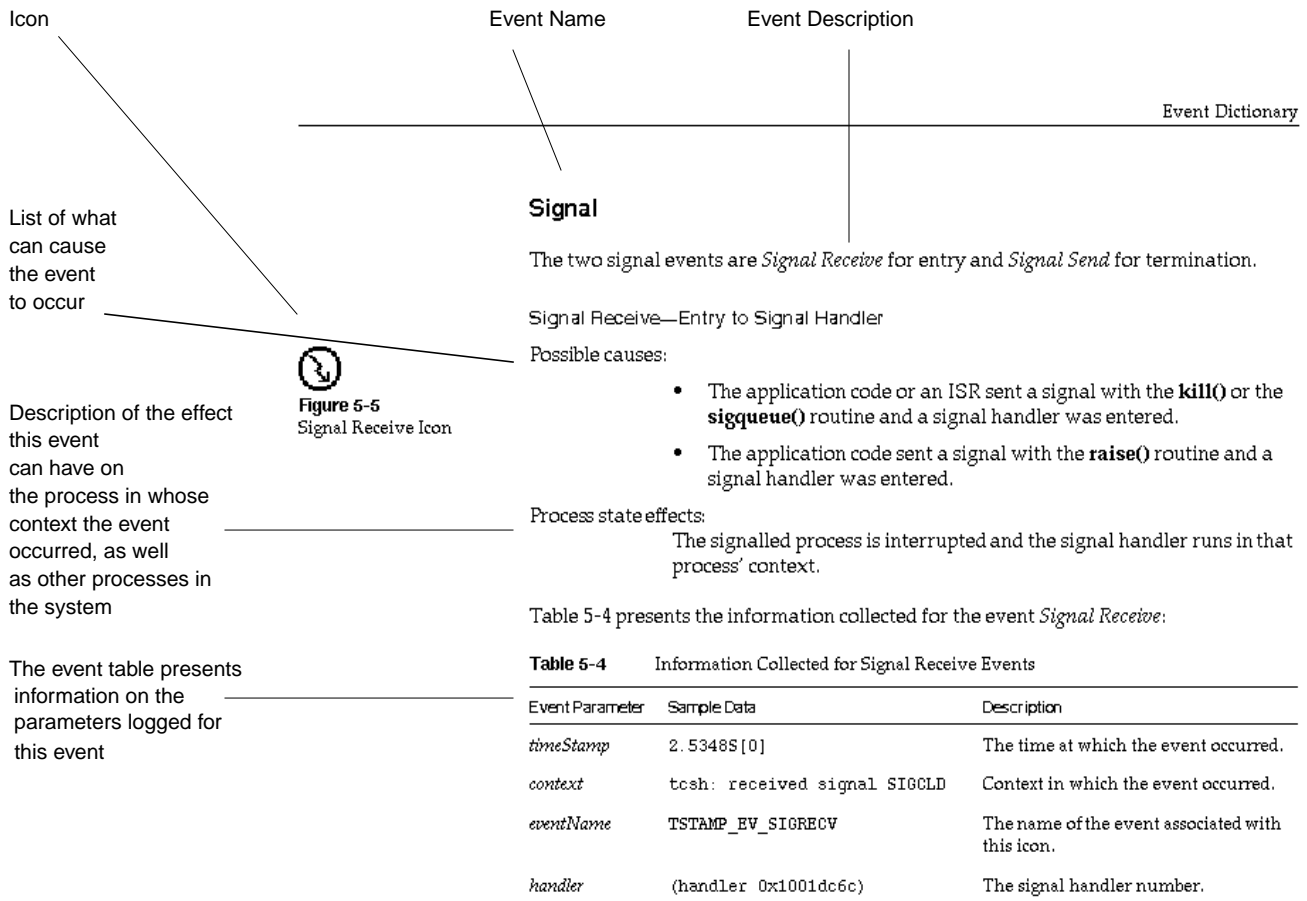


Figure 5-2 Sample Event Dictionary Page

Most of the elements that are called out on the sample page are self-explanatory, with the possible exception of the event table, which presents a definition of the information collected about the event.

The event table describes the information that is logged for a particular event.

Looking at the sample *Signal Receive* event dictionary page above, you see the *context* (the process, ISR, or idle thread in which the event occurred) and the *eventName*. You also see that the *timeStamp* is logged. For information on starting event logging, see Chapter 2, “Collecting Event Data.”

For example, if you drag the *Signal Receive* icon into the Event Inspector window, you see something like this:

```
TSTAMP_EV_SIGRECV:
    0.2947S[1]          tcsh:received signal SIGCLD
    (handler 0x1001dc6c)
```

This provides the following information:

- A *Signal Receive* event occurred (internal event type `TSTAMP_EV_SIGRECV`).
- The second line indicates the time (in seconds) since tracing began (0.2947).
- `tcsh:` indicates the process name that received signal `SIGCLD`
- the final line specifies the signal handler number (0x1001dc6c).¹

Note: If an invalid parameter is passed to a routine, the event icon may not appear, depending on whether the error was detected before or after event logging occurred. In particular, if an invalid object ID is passed to a routine, the event icon will not appear.

Event Dictionary

The event dictionary is organized into the following sections:

- “Interrupt Service Routine (ISR)” on page 48 describes interrupt events.
- “Signal” on page 51 describes software signal and termination events.
- “Processes” on page 53 describes process deletion events.
- “Unknown” on page 55 describes events that cannot be classified.
- “User Event” on page 55 describes non-kernel process events.

¹ A unique event ID is assigned to each event in the log. This is to differentiate between a process of a particular ID that is deleted and a new process spawned with the deleted process’s ID.

Interrupt Service Routine (ISR)

The two interrupt events are *Interrupt Entry* for start and *Interrupt Exit* for transition.

Interrupt Entry—Entry to ISR

Possible causes:

A hardware interrupt occurred for which there is an associated ISR.

Process state effects:

If the interrupt occurs in the context of an executing process, the process is displayed as making a transition to the ready state when the ISR starts executing.

In IRIX, there is a single global interrupt entry point for each processor. With *irixview*, this global entry point appears in a Context View graph as interrupt 0 ([x] Int 1st Level 0). All interrupts received by a processor pass through this global entry point. Each interrupt may be further qualified by other kernel events that appear in the Context View graph as additional interrupts. A description of interrupts is provided in Table 5-3.

Note: This icon is displayed by default. To suppress this icon, toggle the Interrupts button in the Display Events/States window.

Table 5-1 presents the information collected for the event *Interrupt Entry*.

Table 5-1 Information Collected for Interrupt Entry Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	0.0532S[0]	The time at which the event occurred.
<i>context</i>	CPU counter interrupt entry	Context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_CPUCOUNTER_INTR	The name of the event associated with this icon.

 **Figure 5-3**
Interrupt Entry Icon



Figure 5-4
Interrupt Exit Icon

Interrupt Exit—Exit From ISR

Possible causes:

The ISR finished executing.

Process state effects:

When this ISR finishes executing, control returns to the interrupted context.

Note: The display of this icon is suppressed by default. To display this icon, toggle the Interrupts button in the Display Events/States window.

Table 5-2 presents the information collected for the event *Interrupt Exit*.

Table 5-2 Information Collected for Interrupt Exit Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	0.0532S[0]	The time at which the event occurred.
<i>context</i>	CPU counter interrupt exit	Context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_INTREXIT	The name of the event associated with this icon.

Interrupt Types

The following interrupts are identified by IRIXview. Each interrupt is delineated by an interrupt entry and interrupt exit timestamp.

Table 5-3 Interrupts Defined for IRIXview

Interrupt Level	Description	Probable Cause
Int 1st Level (0)	Interrupt qualifier	All interrupts that occur under IRIX “pass through” this level. Some interrupts are not further qualified by IRIXview (for example, vsync and VME interrupts) and appear at this level only.
Int Level 1 (1)	Frame Scheduler Interrupt	This is a further qualified version of the CC counter interrupt (see Int 7) indicating that the Frame Scheduler interrupted the processor.

Table 5-3 Interrupts Defined for IRIXview

Interrupt Level	Description	Probable Cause
Int User Level (4)	User-Level Interrupt	By default IRIX uses no user-level interrupts. This interrupt appears only on systems where users have provided ULIs; see uli(3).
Int CPU Yield (5)	Frame Scheduler yield	Not actually an IRIX interrupt, this level indicates that a user process has called either the frs_yield() function or the schedctl() (MPTS_FRS_YIELD) system call.
Int CPU Ctr (6)	CPU tick interrupt	The IRIX scheduler interrupts each processor every 10 milliseconds. IRIX uses this interrupt to perform scheduling processes and other housekeeping functions.
Int RTC Ctr (7)	CC counter interrupt	An event timeout interrupt, either from expiration of a kernel-initiated timer or a user-initiated timer (ITIMER).
Int Prof Ctr (8)	Profiler interrupt	An interrupt initiated from either the profiler (see prof(1)) or the built-in audio hardware. ^a
Int Group (9)	Group interrupt	Designates interrupt occurring on IRIX multiprocessor systems. Group interrupts are seen when multiple Frame Schedulers are synchronized across multiple processors.
Int Inter-CPU (11)	Inter-processor interrupts	One processor has interrupted another based on some action initiated by the first processor (for example, TLB flush, TLB fault, and so forth).
Int Network (12)	Network interrupts	Some network drivers, such as the Challenge ethernet driver, provide this interrupt event.

a. Note that on systems with built-in audio (such as the O2, Octane, and Onyx2) a profiler interrupt occurs every millisecond, even when no audio is being played or recorded.

Signal

The two signal events are *Signal Receive* for entry and *Signal Send* for termination.

Signal Receive—Entry to Signal Handler

Possible causes:

- The application code or an ISR sent a signal with the **kill()** or the **sigqueue()** routine and a signal handler was entered.
- The application code sent a signal with the **raise()** routine and a signal handler was entered.

Process state effects:

The signalled process is interrupted and the signal handler runs in that process' context.

Table 5-4 presents the information collected for the event *Signal Receive*:

Table 5-4 Information Collected for Signal Receive Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	2.5348S[0]	The time at which the event occurred.
<i>context</i>	tcsh: received signal SIGCLD	Context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_SIGRECV	The name of the event associated with this icon.
<i>handler</i>	(handler 0x1001dc6c)	The signal handler number.



Figure 5-5
Signal Receive Icon



Figure 5-6
Signal Send Icon

Signal Send—Send Signal to a Process

Possible causes:

- Application code or an ISR sent a signal to the specified process with the **kill()** or **sigqueue()** routine.
- Application code sent a signal to the calling process with the **raise()** routine.

Process state effects:

A process receives a pending signal the next time the process exits from the kernel domain. For most signals, this could occur:

- when the process is dispatched after a wait or preemption
- upon return from some system call
- upon return from the kernel’s usual 10-millisecond tick interrupt
- at the start of a minor frame, under the Frame Scheduler

SIGALRM is delivered as soon as the kernel is ready to return to user processing after the timer interrupt, to preserve timer accuracy. Thus, for a process that is ready to run, in a processor that has not been made nonpreemptive, normal signal latency is at most 10 milliseconds and SIGALRM latency is less. However, when the receiving process is not ready to run, or when there are competing processes with superior priorities, the delivery of a signal is delayed until the next time the receiving process is scheduled.

Table 5-5 presents the information collected for the event *Signal Send*.

Table 5-5 Information Collected for kill Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	2.5344S[1]	The time at which the event occurred.
<i>context</i>	x: was sent signal SIGTSTP	Context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_SIGSEND	The name of the event associated with this icon.
<i>processID</i>	mediad (5807)	PID of the process to receive the signal.

Processes

The process event is *Exit Process* for process termination.

Fork Process—Spawn a Process

Possible causes: The system or application code called the **fork()** routine.

Process state effects:

The parent process usually executes another image by means of the **exec()** routine, and a context switch occurs.

Table 5-6 presents the information collected for the event *Fork Process*.

Table 5-6 Information Collected for processCreate Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	1.6621S[0]	The time at which the event occurred.
<i>context</i>	rlog: process fork	The context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_FORK	Name of the event associated with this icon.

Exit Process—Delete a Process

Possible causes: The system or application code received a signal with the action to terminate the process, or the application code called the **exit()** routine.

Process state effects:

If the routine is successful, the specified process is terminated. If the executing process kills itself, a context switch occurs.

Table 5-7 presents the information collected for the event *Exit Process*.

Table 5-7 Information Collected for processDelete Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	1.6621S[0]	The time at which the event occurred.
<i>context</i>	rlog: process exit	The context in which the event occurred.
<i>eventName</i>	TSTAMP_EV_EXIT	Name of the event associated with this icon.



Figure 5-7
Fork Process Icon



Figure 5-8
Exit Process Icon

Process Events

Table 5-8 shows the various process events icons in IRIXview.

Table 5-8 Process Event Icons









Process Event Type	Event Icon	Numeric Range
Net Event		K400-K499
Disk Event		K300-K399
Profile Event		K18, K19
Miscellaneous Kernel Event		
Virtual Memory Event		K100-K199
System Call Begin Event		K22
System Call End Event		K23
Memory Allocate Event		K28



Figure 5-9
Unknown Event Icon

Unknown

Unknown Event—Unknown Event

Possible causes:

IRIXview has received an event that it does not recognize.

Process state effects:

Indeterminate.

Table 5-9 presents the information collected for the event *unknown*:

Table 5-9 Information Collected for Unknown Events

Event Parameter	Sample Data	Description
<i>UnknownId</i>	TSTAMP_UNKNOWN	The ID of the event.

User Event

DefaultUser—Display User-specified Event

Possible causes:

The application code called the `rtmon_log_user_tstamp()` function.

Process state effects:

None.

Table 5-10 presents the information collected for the event *defaultUser*:

Table 5-10 Information Collected for defaultUser Events

Event Parameter	Sample Data	Description
<i>timeStamp</i>	1.6621S[0	The time at which the event occurred.
<i>context</i>	myapp: entering myfunction()	Context in which the event occurred.
<i>userEventId</i>	TSTAMP_USER_EVENT	The name and number of the user event.
<i>address</i>	0x1a644	The address at which the event-point was set.



Figure 5-10
Default User Icon

Other information may be collected based on the passing of parameters to the **rtmon_log_user_tstamp()** routine. See the `rtmon_log_user_tstamp(3)` man page or “Adding Timestamps to Your Program” on page 13 for more information.

Many IRIX kernel events are shown as “user” events—or, more accurately, user event numbers above 20000 are reserved for kernel events. These events are unique to IRIXview and are described in Table 5-11. In the table, IRIX REACT/Pro Frame Scheduler is designated as FRS for convenience.

Note: For online use, see the Legend window for icon pictures and explanations.

Table 5-11 REACT/Pro Events and Event Numbers













Event Number and Icon	Description	Possible Cause
K0	Undefined system event	An event (timestamp) of an unknown type was generated from the IRIX kernel.
K1	Undefined daemon event	An event (timestamp) of an unknown type was generated from an IRIX daemon.
K2 	FRS dispatch	The FRS chooses a new process to run, or idles the processor if no other FRS processes are ready to run.
K3 	FRS yield	A user process has called either the <code>frs_yield()</code> function or the <code>schedctl (MPTS_FRS_YIELD)</code> system call. This event generally appears with INT 5.
K4 	FRS fatal frame overrun	The FRS has incurred a frame overrun (indicates that a process failed to yield in a minor frame) and there is no recovery mechanism in place.
K8 	FRS frame underrun	The FRS has incurred a frame underrun (indicates that a process should have been dispatched in a minor frame and was not). Qualifier #2 returns the current number of underruns.
K9 	FRS frame overrun	The FRS has incurred a frame overrun (indicates that a process failed to yield in a minor frame). This event is accompanied with either event 20010, 20011, 20012, or 20013. Qualifier #2 returns the current number of overruns.

Table 5-11 (continued) REACT/Pro Events and Event Numbers

Event Number and Icon	Description	Possible Cause
K10 	FRS no recovery	No recovery mechanism is in place upon an overrun condition (this event is always accompanied by event 20009).
K11 	FRS inject frame	An FRS minor frame has been injected as the recovery mechanism for a frame overrun occurrence (this event is always accompanied by event 20009).
K12 	FRS stretch frame	An FRS minor frame has been stretched as the recovery mechanism for a frame overrun occurrence (this event is always accompanied by event 20009).
K13 	FRS steal frame	An FRS minor frame has stolen time from an adjacent frame as the recovery mechanism for a frame overrun occurrence (this event is always accompanied by event 20009).
K14 	FRS maximum errors	The FRS has received the maximum number of allowable errors (overruns or underruns) permitted by the application or program. This event causes the FRS to terminate.
K15	Dropped timestamps	In some extreme circumstances, the <i>rtmond</i> daemon issues this event to indicate that timestamps have been dropped from the event stream (this could occur, for example, when events are being generated faster than they can be saved).
K16 	FRS start of major frame	Designates the start of a Frame Scheduler major frame.
K17 	FRS start of minor frame	Designates the start of a Frame Scheduler minor frame.

User Interface Reference

This chapter provides a reference to the IRIXview GUI menus and icons, presented in alphabetical order by menu or icon name:

- “About IRIXview Window” on page 61
- “Context Graph Options” on page 61
- “Data Format Options” on page 62
- “Display Events/States Window” on page 62
- “Event Inspector Window” on page 65
- “Legend Window Icon” on page 66
- “New Context Graph Menu” on page 68
- “New CPU Graph Window” on page 69
- “Open Event File Window” on page 70
- “Pan Left/Pan Right Icons” on page 72
- “Push/Pop/Exchange Icons” on page 72
- “Quit Menu Choice” on page 73
- “Save Event File Window” on page 73
- “Scheduler Summary Window” on page 74
- “Search Accelerator Icons” on page 76
- “Search Window Icon” on page 76
- “System Call Summary Window” on page 78
- “Target Window” on page 80
- “Time Units Menu Icon” on page 81
- “View Options Window” on page 81
- “Zoom In/Zoom Out Icons” on page 83

IRIXview commands are located in two places:

- In one of the IRIXview main window menus: File, Windows, Options, or Help. The main window, shown in Figure 6-1, displays when you enter the *irixview* command; see “Starting IRIXview” on page 4.

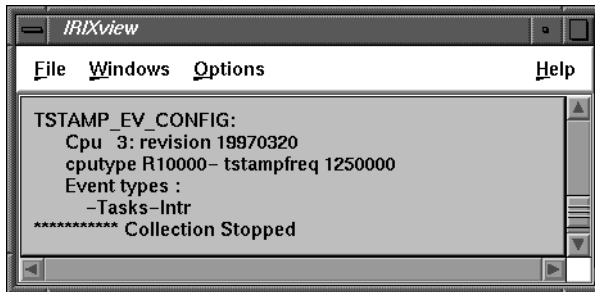


Figure 6-1 IRIXview Main Window

- In the icon bar across the top of a Context View or CPU View Graph, as shown in Figure 6-2. Choose Windows > New Context Graph to display a Context View graph, or Windows > New CPU Graph to display a CPU View graph.

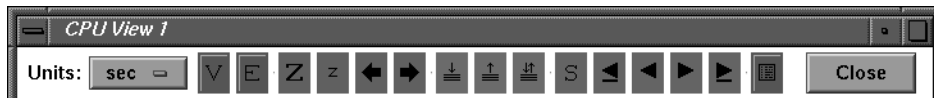


Figure 6-2 Vie Graph Icon Bar

The icon bar in a Context View window looks the same as in a CPU View window. For general information about using IRIXview, see Chapter 2, “Collecting Event Data.”

About IRIXview Window

When you choose Help > About IRIXView, the program displays the window shown in Figure 6-3, which contains version and copyright information. To dismiss this window, click the OK button or double-click the control menu bar in the upper left hand corner.

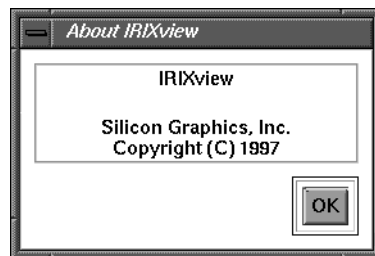


Figure 6-3 About IRIXview Window

Context Graph Options

When you choose Options > Context Graph, the program displays the window shown in Figure 6-4, which provides useful control over all active Context View graphs.

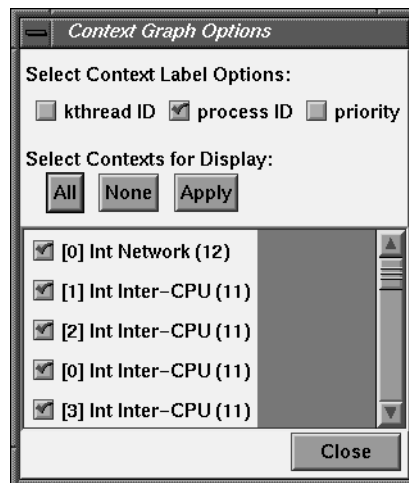


Figure 6-4 Context Graph Options Window

To display the kernel thread ID number in the interrupt and process buttons of all active Context View graph windows, check mark the Kthread ID box. To display process or thread priorities, check mark the Priority box. To remove the default process ID number, uncheck mark the Process ID box. The effect is immediate.

The scrolling list of this window contains a list of all interrupts and processes shown in the Context View window. By default, all items in the list are check marked. You can uncheck them all by clicking the None button, or uncheck mark individual items in the scrolling list. The All button check marks all items again. To change the display of items in the Context View graph, click the Apply button.

Data Format Options

For events that print `qual[0-3]` data, especially user-defined events, choosing the Options > Data Format menu item allows you to change the output data format from decimal (default) to octal, hexadecimal, or ASCII.

Display Events/States Window

This icon is located in the icon bar of a Context or CPU View graph, which are displayed with the “New Graph” choices in the Windows menu.



Figure 6-5 Display Events/States Icon

When you click this icon, IRIXview displays the Display Events/States window, as shown in Figure 6-6. This window contains commands that let you change which events are displayed within the graph subwindow, and how they are treated.

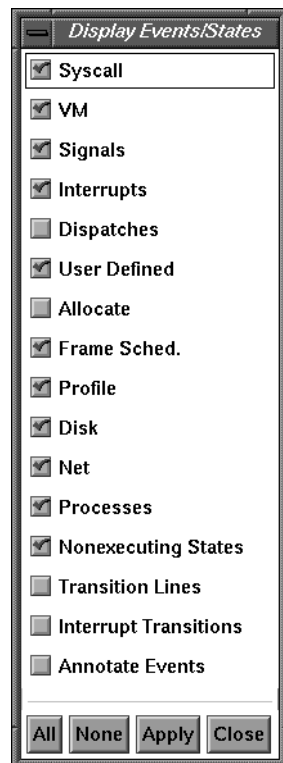


Figure 6-6 Display Events/States Window

To control which elements are displayed in the Context View or CPU View graph, toggle these event or state types on (or off) and click Apply. When you bring up a Display Events/States window from a CPU View graph, some of the items are grayed out. From a Context View graph, some event or state types are not displayed by default.

Note: This is different from the Target window, which controls event logging, rather than event display; see "Target Window" on page 80.

The following list describes the event or state types you can display:

- Syscall If toggled on, system call icons are displayed.
- VM If toggled on, virtual memory event icons are displayed.
- Signals If toggled on, signal event icons are displayed.
- Interrupts If toggled on, interrupt event icons (*intEnt* and *intExit*) are displayed.
- Dispatches If toggled on, dispatch event icons are displayed
- User Defined If toggled on, user-defined event icons are displayed.
- Allocate If toggled on, memory allocation event icons are displayed.
- Frame Sched. If toggled on, frame scheduler (FRS) event icons are displayed?
- Profile If toggled on, profiler event icons are displayed?
- Disk If toggled on, disk I/O event icons are displayed?
- Net If toggled on, network event icons are displayed?
- Processes If toggled on, process event icons are displayed.

Note: Not all listed process events can be displayed (for example, the *processSpawn* event cannot be displayed).

Nonexecuting States

If toggled on, process states besides executing state (for example, pended state and ready state) are displayed.

Transition Lines

If toggled on, vertical lines connecting a previous context to the current context are displayed.

Interrupt Transitions

If toggled on, vertical lines to or from an interrupt context are displayed.

Annotate Events

If toggled on, number are placed alongside event icons.

Event Inspector Window

This choice is located in the Windows menu, and displays the Event Inspector window, as shown in Figure 6-7.

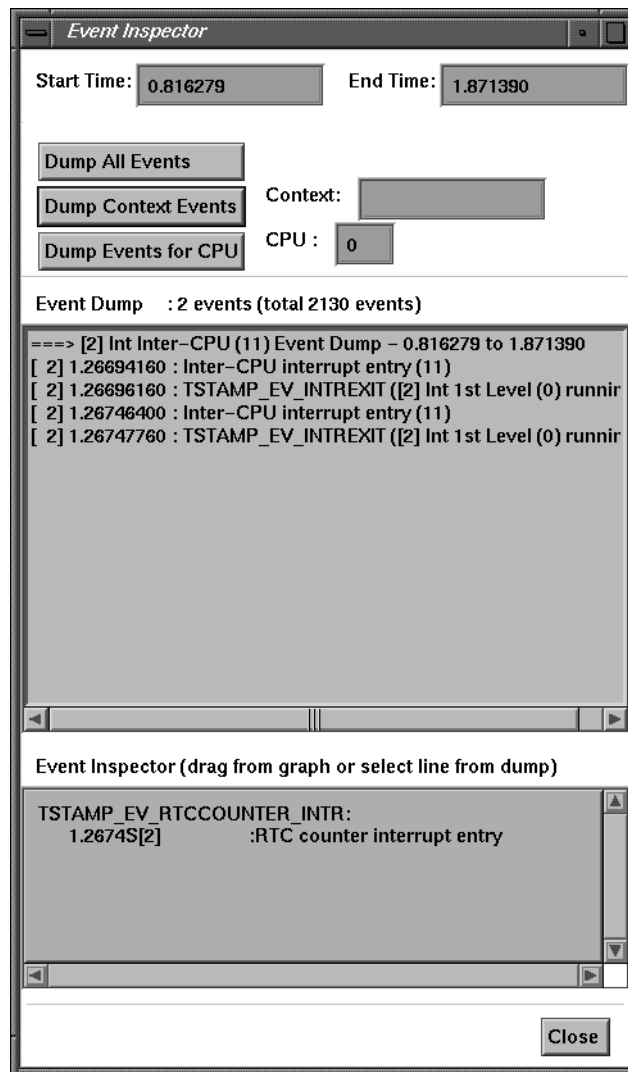


Figure 6-7 Event Inspector Window

To use this window:

1. Click an event icon in an event log with the middle mouse button.
2. Drag the event icon to the Event Inspector window. The cursor changes to the shape of the event icon, letting you see what is being dragged.
3. “Drop” the icon into the Event Inspector. Information about that event appears.
(For details on what type of information is logged for an event at each mode, see Chapter 5, “Event Dictionary.”)
4. When you drag a new icon into the window, it overwrites the previous event’s information. You can also move the cursor into the window and press the C key to clear the Event Inspector.

Legend Window Icon

The Legend Window icon is located in the icon bar of the Context View or CPU View windows, which may be displayed by choosing Windows > New...Graph.



Figure 6-8 Legend Window Icon

Click this icon to display a scrollable Legend window, showing what each event icon and stipple means. Legend windows for the Context View and CPU View are different; both are shown in Figure 6-9.

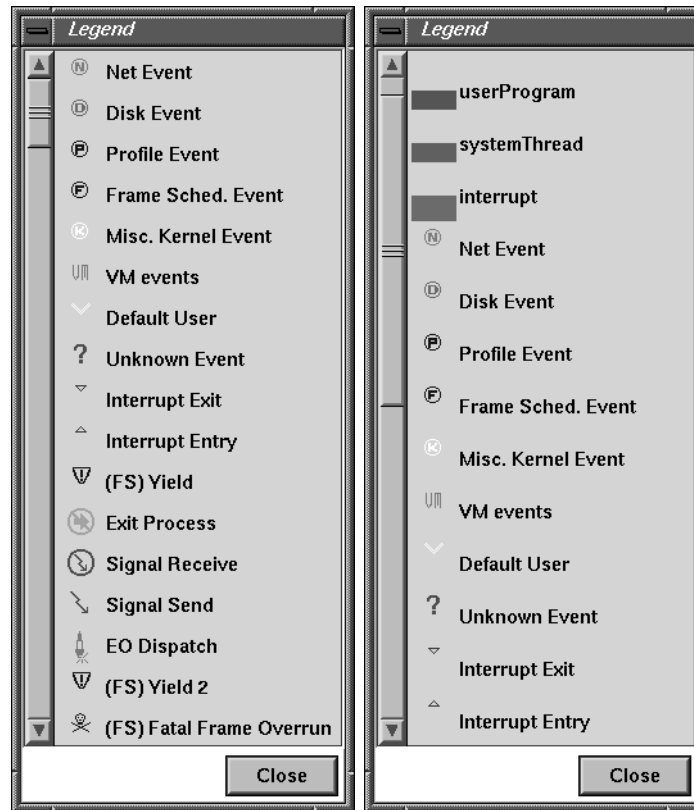


Figure 6-9 Legend Windows (Context and CPU)

New Context Graph Menu

This choice is located in the Windows menu and displays a Context View graph window where you can examine event data, as shown in Figure 6-10.

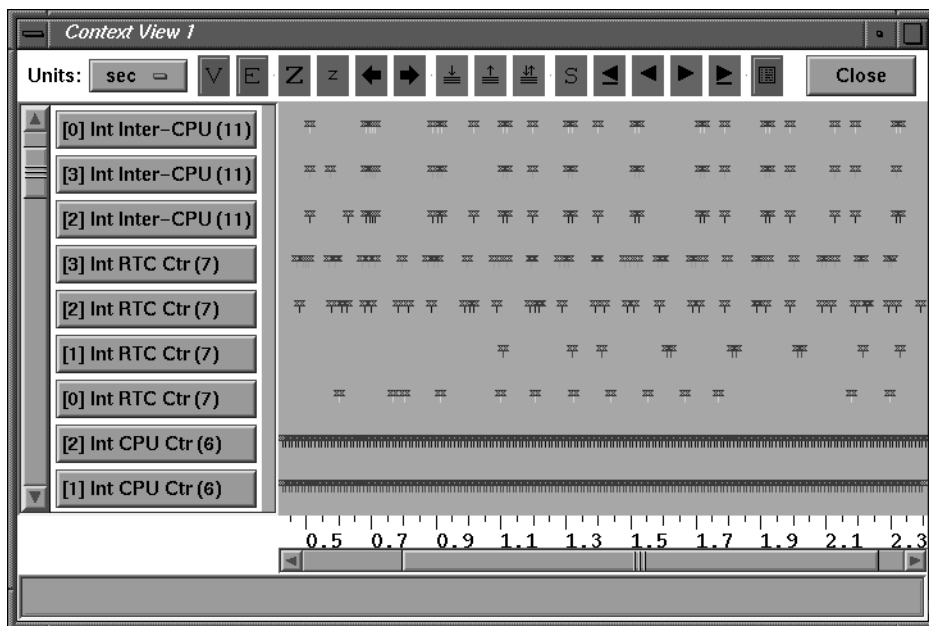


Figure 6-10 Context View Graph Window

The Context View is a window into the event data; in most cases, it does not show the entire event log. Instead, what is shown is a *time interval*. Use your window manager to resize the Context View window, if needed. You can refresh the Context View graph at any time by moving the cursor into the window and clicking the right mouse button.

The first Context View window that you display is labeled View 1. You can display up to 16 Context View windows at a time, which can be useful for looking at different portions of the same event log. Each one is numbered in the order that it is displayed; the second graph would be labeled Context View 2, and so on. When you display auxiliary windows (for example, by clicking the V icon to display the View Options window; see “View Options Window” on page 81), auxiliary windows are numbered to match the Context View from which they were invoked. For example, if you click the V icon from View 3, the resulting window is labeled View Options 3.

New CPU Graph Window

This choice is located in the Windows menu and displays a CPU View graph window where you can examine processor data, as shown in Figure 6-11.

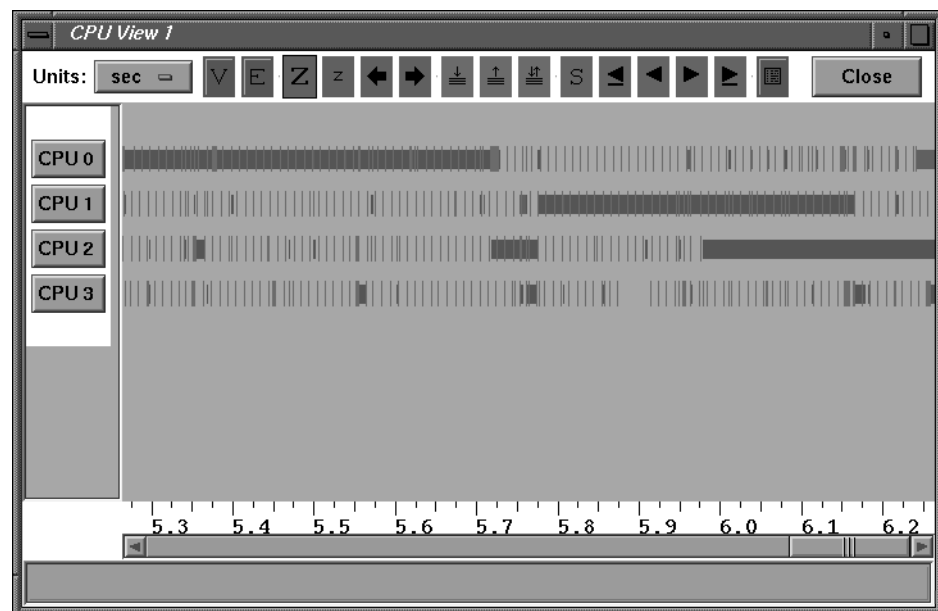


Figure 6-11 CPU View Graph Window

The CPU View is a window into the processor data; in most cases, it does not show the entire event log. Instead, what is shown is a *time interval*. Use your window manager to resize the CPU View window, if needed. You can refresh the CPU View graph at any time by moving the cursor into the window and clicking the right mouse button.

The first CPU View window that you display is labeled View 1. You can display up to 16 CPU View windows at a time, which can be useful for looking at different portions of the same event log. Each one is numbered in the order that it is displayed; the second graph would be labeled CPU View 2, and so on. When you display auxiliary windows (for example, by clicking the V icon to display the View Options window; see “View Options Window” on page 81), auxiliary windows are numbered to match the CPU View from which they were invoked. For example, if you click the V icon from View 3, the resulting window is labeled View Options 3.

Open Event File Window

When you choose File > Open, IRIXview displays the Open Event File window, as shown in Figure 6-12. This window lets you open an event log created with the Save Event File window; see “Save Event File Window” on page 73.

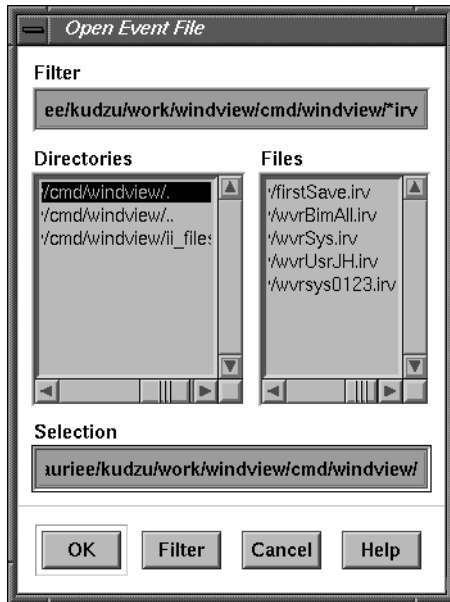


Figure 6-12 Open Event File Window

The current working directory is displayed in the Filter and Selection fields by default. As you move to other directories, the Directories and Files subwindows automatically resize to show as much information as possible.

To use this window, follow these steps:

1. To search for the event log to open, use the Filter field to specify a particular directory, double-clicking names in the Directories subwindow, clicking the *Filter* button, or pressing the **Enter** key, as appropriate.

As you “filter” directories, they are listed in the Selection field, and their subdirectories and files are listed in the Directories and Files subwindows. You can continue filtering directories in this manner until the appropriate directory name is specified in the Selection field.

2. Once you are in the correct directory, use the Selection field to specify the particular event log to open. You can click the name in the Files subwindow or type the name in. Files are named with the syntax *name.processor_number.irv*.
3. Double-click the event log name, click the OK button, or press Enter to analyze the file and remove the Analyze Event Log window from the screen.

If the event log is successfully opened, a message like the following appears in the Main window message area:

```
Connected to host localhost.  
4 CPUs on localhost, 4 selected CPUs  
TSTAMP_EV_CONFIG:  
  Cpu    0: revision 19970320  
  cputype R4400- tstampfreq 47619047  
  Event types :  
    -Tasks-Intr ...
```

Then, if a Context View graph is displayed, the event log is displayed there; see “New Context Graph Menu” on page 68. (Note that you can display the Context Graph either before or after you have opened an event log.)

At any time, you can click the Cancel button to dismiss the Open Event File window from the screen, or click the Help button to display information on this window.

Pan Left/Pan Right Icons

These icons are located in the icon bar of the Context View or CPU View windows, which may be displayed by choosing Windows > New...Graph.



Figure 6-13 Pan Left/Pan Right Icons

Pan Left and Pan Right move the time interval one page to the left or right, where a page is defined by the width of the current time interval.

Push/Pop/Exchange Icons

These icons are located in the icon bar of the Context View or CPU View windows, which may be displayed by choosing Windows > New...Graph.



Figure 6-14 Push/Pop/Exchange Icons

The Push icon saves the current time interval. You can later move back to this time interval with the Pop or Exchange icon. You can push up to 16 time intervals— if you push more than 16, the oldest intervals are discarded in FIFO order.

The Pop icon causes the most recently pushed time interval to be displayed.

The Exchange icon swaps the currently displayed time interval with the most recently pushed time interval. For example, find an interval that is of interest to you and save it with the Push icon. Then click the Exchange icon repeatedly to move between that interval and the current interval.

Quit Menu Choice

To exit IRIXview, choose File > Quit. The IRIXview main window and all its related windows are removed from the screen.

Note: When you exit IRIXview, you are not prompted to save your event data. To save event data before exiting, see “Save Event File Window” on page 73.

Save Event File Window

This choice is located in the File menu, and displays the Save Event File window, as shown in Figure 6-15. This window lets you save event logs that you can later with the Open Event File window; see “Open Event File Window” on page 70.

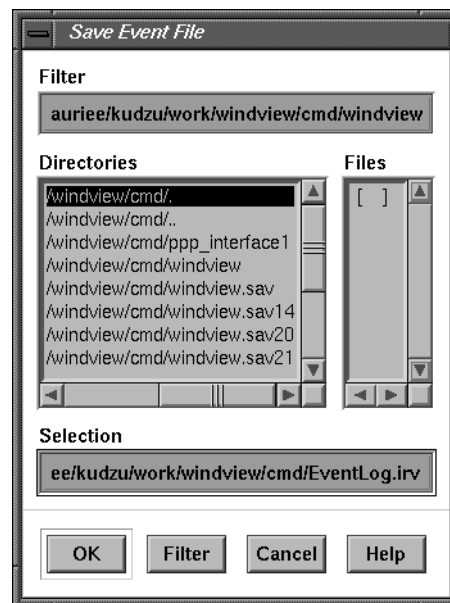


Figure 6-15 Save Event File Window

The current directory is displayed in the Filter and Selection fields. As you move to other directories, the Directories and Files subwindows are automatically resized to show as much information as possible.

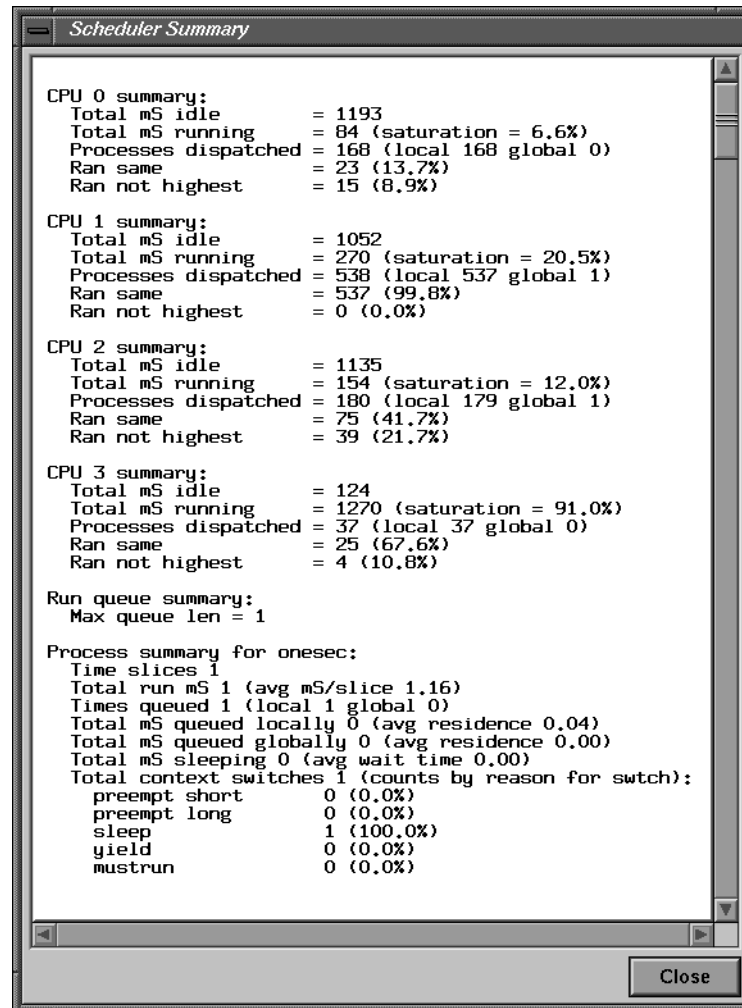
1. To search for a save directory, use the Filter field to specify the directory name, double-clicking names in the Directories subwindow, clicking the *Filter* button, or pressing the **Enter** key, as appropriate. As you filter directories, they are listed in the Selection field, and the files they contain are listed in the subwindows. Continue filtering directories until the one you want appears in the Selection field.
2. Once you are in the correct directory, use the Selection field to specify the particular event log to save. Click a name in the Files subwindow or type the name.
3. Double-click the event log name, click the OK button, or press the **Enter** key to save the file and dismiss the Save Event File window from the screen.

When an event log is saved, its filename usually has an *irv* suffix, for example: *filename.irv*.

At any time, you can click the Cancel button to remove the Save Event File window from the screen, or click the Help button to display information on the window.

Scheduler Summary Window

When you choose Windows > Scheduler Summary from the main menu of IRIXview, the Scheduler Summary window appears, as shown in Figure 6-16. This window shows status information for all processors, the run queue length, and scheduler status.



```
Scheduler Summary

CPU 0 summary:
  Total mS idle      = 1193
  Total mS running  = 84 (saturation = 6.6%)
  Processes dispatched = 168 (local 168 global 0)
  Ran same          = 23 (13.7%)
  Ran not highest   = 15 (8.9%)

CPU 1 summary:
  Total mS idle      = 1052
  Total mS running  = 270 (saturation = 20.5%)
  Processes dispatched = 538 (local 537 global 1)
  Ran same          = 537 (99.8%)
  Ran not highest   = 0 (0.0%)

CPU 2 summary:
  Total mS idle      = 1135
  Total mS running  = 154 (saturation = 12.0%)
  Processes dispatched = 180 (local 179 global 1)
  Ran same          = 75 (41.7%)
  Ran not highest   = 39 (21.7%)

CPU 3 summary:
  Total mS idle      = 124
  Total mS running  = 1270 (saturation = 91.0%)
  Processes dispatched = 37 (local 37 global 0)
  Ran same          = 25 (67.6%)
  Ran not highest   = 4 (10.8%)

Run queue summary:
  Max queue len = 1

Process summary for onesec:
  Time slices 1
  Total run mS 1 (avg mS/slice 1.16)
  Times queued 1 (local 1 global 0)
  Total mS queued locally 0 (avg residence 0.04)
  Total mS queued globally 0 (avg residence 0.00)
  Total mS sleeping 0 (avg wait time 0.00)
  Total context switches 1 (counts by reason for swtch):
    preempt short    0 (0.0%)
    preempt long     0 (0.0%)
    sleep            1 (100.0%)
    yield            0 (0.0%)
    mustrun          0 (0.0%)

Close
```

Figure 6-16 Scheduler Summary Window

Search Accelerator Icons

The Search Accelerator icons are located in the icon bar of the Context View or CPU View windows, which may be displayed by choosing Windows > New...Graph.

These icons find the next or previous occurrence of the currently selected event.



Figure 6-17 Search Accelerator Icons

An event may have been selected because it was found by a previous search request, or you may have selected it with the middle mouse button (see “Selecting Event Data” on page 26).

The underlined arrows find the next (or previous) occurrence of the currently selected event in the same context, that is, in the same process, interrupt level, or idle thread context.

The middle arrows without underlines search for the next or previous occurrence of the currently selected event, regardless of context.

Search Window Icon

The Search Window icon is located in the icon bar of the Context and CPU graphs. To open a graph window, choose Windows > New Context Graph or New CPU Graph.



Figure 6-18 Search Window Icon

When you click the Search Window icon, a Search window displays, as shown in Figure 6-19. (The number 1.1 in the title indicates that this is the first search window for the corresponding first View window.)

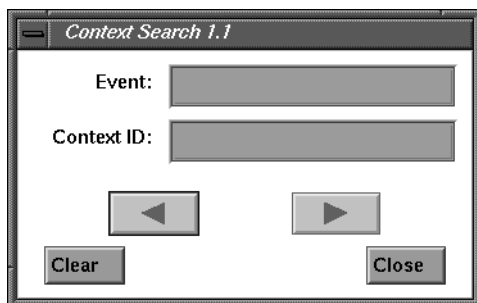


Figure 6-19 Search Window

You can search for a particular event, the next or previous event in a particular context, or the next or previous event of any type in any context. Context entries can be the context name only, the process ID (PID), or the kernel thread ID (KID). Follow these steps to use the Search window:

1. Specify a particular event by entering its name in the Event field, or by dragging and dropping an icon from the Legend window.

If you know its name, you can enter the event by typing its name into the Event field. You can also drag and drop icons from the (Context) Legend window into the Event field (the exceptions are the *defaultUser* and *unknown* icons). See "Selecting Event Data" on page 26 for information on dragging event icons.

Leave this field blank if you are searching for any event in a particular context, or any event in any context.

2. You can further constrain the search for a particular event by specifying its context ID (interrupt level, process, or idle thread).

Enter the context information by typing it or by dragging the icon of interest into the field. You can also enter the information by selecting the context label from the vertical axis with the middle mouse button, then dragging the word "CONTEXT" that appears into the Context ID field.

If there are multiple contexts with the same name and only a name is entered, the program uses the first context it finds with a matching name.

If the Event field is blank, any data in the Object ID field are ignored, and the next or previous event of any type is found in the specified context.

If the Event field and the Context ID field are blank, then the next or previous event of any type in any context is found.

3. After you have specified the search parameters, click the appropriate arrow to perform the search.

When the next (or previous) occurrence of the event is found, the Context View graph displays the time interval in which it occurs, and indicates the occurrence that it found by placing a vertical line through it. Timing information is displayed in the Detailed Information field (see “Using the Context View Graph” on page 22).

The CPU Search window works the same way, but note that event symbols are not normally displayed in the CPU View graph window. You have to enable symbol display by check marking events in the Display Events/States window. For more information, see “Display Events/States Window” on page 62.

To search for another occurrence of the event, you can use the Search window again, or you can use the Search Accelerator icons in the Context Graph. For information on these icons, see “Search Accelerator Icons” on page 76.

Note: You can display multiple Search windows for a graph. Each Search window is labeled sequentially. The first number represents the graph window from which you invoked the Search window, and the second number represents which Search window it is, relative to all Search windows currently displayed for this Context graph. For example, if you have two Context graphs open, the second window you opened is labeled Context View 2. Clicking the S button in this window launches the Search dialog labeled 2.1, the first search for View 2. The next Search window for Context View 2 would be labeled Search 2.2. The maximum number of Search windows for each Context View graph is 16.

System Call Summary Window

Before using this window, you need to collect system call information, not collected by default. First run *irixview* as superuser. In the Target window, select Syscall, and collect event data from the local host.

Choosing Windows > System Call Summary then shows system calls that occurred during the collection period, as shown in Figure 6-20.

The screenshot shows a window titled "System Call Summary" with a scrollable list of system calls. The data is as follows:

Name	#Calls	Average Time(ms)	Total Time(ms)
tstamp_wait	798	1372.82	1095508.62
read	208	889.30	184975.28
sginap	46	11855.83	545368.06
select	74	54.66	4044.78
write	132	1396.80	184377.88
ioctl	4062	44.58	181091.67
usync_cntl	48	3760.33	180495.91
fcntl	22	3.12	68.73
msync	2	18.26	36.51
schedctl	1358	0.03	34.64
open	106	851.05	90211.52
tstamp_update	750	0.02	15.67
syssgi	100	0.14	13.79
close	102	0.09	9.35
sprocp	2	2.22	4.44
getpid	684	0.01	4.25
brk	26	6938.09	180390.34
waitsys	30	0.05	1.49
ngetdents	2	0.68	1.35
unlink	2	0.53	1.05
sysmp	20	0.03	0.70
access	4	0.17	0.67
mmap	2	0.13	0.25
sched_setscheduler	2	0.10	0.19
sysget	4	0.05	0.18
sigprocmask	12	0.01	0.17
getpeername	2	0.07	0.13
swapctl	8	0.02	0.13
fstat	2	0.04	0.07
lseek	4	0.02	0.07
gettimeofday	2	0.02	0.04
time	2	0.02	0.03
ulimit	2	0.01	0.02

Figure 6-20 System Call Summary Window

Target Window

This choice is located in the Windows menu, and displays the Target window, as shown in Figure 6-21. This window lets you do the following:

- Specify the target system. This can be *localhost*, the system running IRIXview.
- Select the processor ID, the processor number(s) where you will be collecting data.
- Start and stop event data collection.

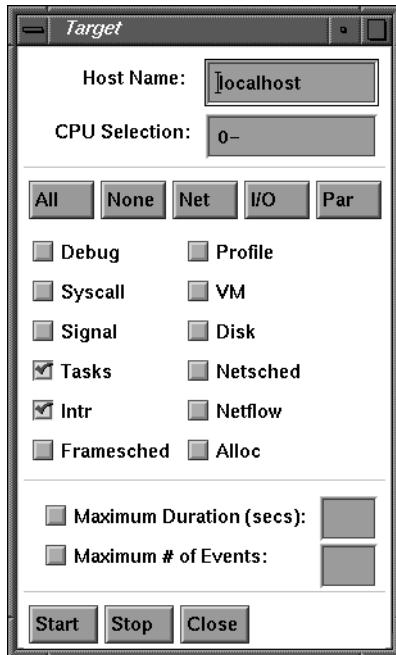


Figure 6-21 Target Window

Time Units Menu Icon

This icon is located in the icon bar of a View Graph, which can be displayed with choices New Context Graph or New CPU Graph in the Windows menu.

Units:  A small rectangular button with a dark background and light text. The text reads "sec" followed by a small downward-pointing arrow icon.

Figure 6-22 Time Units Menu Icon

If you click and hold the left mouse button over this menu icon, you can choose the unit of time displayed in the Timeline and the Detailed Time Information field (see “Using the Context View Graph” on page 22 for information on these locations). The choices are:

<i>sec</i>	seconds (the default)
<i>msec</i>	milliseconds
<i>usec</i>	microseconds
<i>nsec</i>	nanoseconds

View Options Window

This icon is located in the icon bar of a Context or CPU View graph, which are displayed with the “New Graph” choices in the Windows menu.



Figure 6-23 View Control Window Icon

When you click this icon, IRIXview displays the View Control window, as shown in Figure 6-24. This window contains commands that let you change how the event log is treated within the graph subwindow.

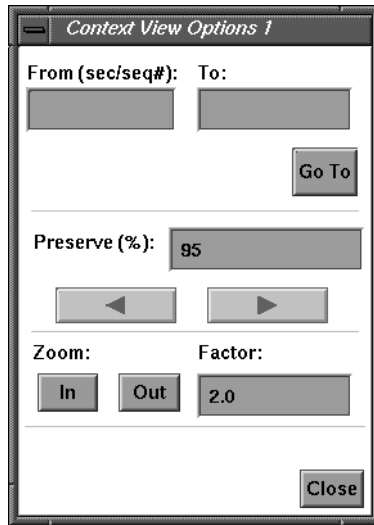


Figure 6-24 View Control Window

To use this window, follow these steps:

1. In the From and To fields, you can specify which time interval, in units of seconds or event sequence numbers, which you would like to examine.

For example, type 1 in the From field and 2.5 in the To field, and then click the Go To button to view the interval from second 1 to second 2.5. Use whole numbers to specify the range of event sequence numbers you want displayed; for example, from 0 to 25, or from 1500 to 2000.

2. The Left and Right Arrow buttons act like the Pan Left and Pan Right icons on the Context Graph (see “Pan Left/Pan Right Icons” on page 72), but are constrained by the Preserve (%) field.

For example, if Preserve is set to 50, the arrows move the view forward or back one-half page at a time (where a *page* is the width of the current time interval). However, if Preserve is set to 90, they move forward and back just 10% of the current time interval at a time. If Preserve is set to 0, they act the same as the icons on the Context Graph; if Preserve is 100, these arrows are disabled.

3. The Zoom In and Zoom Out buttons act like the zoom icons (see “Zoom In/Zoom Out Icons” on page 83) but are constrained by the Factor field.

For example, if Factor is set to 10, this Zoom In displays 10% of the current time interval and this Zoom Out displays 10 times the current time interval. If Factor is set to 2, they act like the zoom icons on the Context Graph (Zoom In displays half the current time interval and Zoom Out displays 2 times the current interval). But if Factor is set to less than 1, the actions of these zoom buttons are reversed.

4. Clicking the Display Events button causes the Display Events/States window to appear; see “Display Events/States Window” on page 62.

Zoom In/Zoom Out Icons

These icons are located in the icon bar of the Context View or CPU View windows, which may be displayed by choosing Windows > New...Graph.

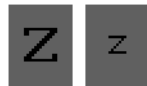


Figure 6-25 Zoom In/Zoom Out Icons

The capital Z (Zoom In) icon lets you focus on details; the lowercase z (Zoom Out) icon lets you focus on a bigger picture. Keyboard shortcuts are available: you can press the **z** key to zoom in or the **z** key to zoom out.

Zoom In halves the time interval displayed, preserving the screen’s midpoint. If a sub-interval is selected, the boundaries of the sub-interval become the time interval’s boundaries.

Zoom Out doubles the current time interval (or a selected sub-interval), preserving the midpoint, if possible.

For information on selecting a sub-interval, see “Selecting Event Data” on page 26.

Glossary

application code

Any user-supplied code operating under IRIX.

collecting event data

See event collection.

context

See context switch, current context, process context.

Context Graph

The IRIXview window that lets you examine event data logged about your software system. In this window, time is represented on the horizontal axis, while the current system's contexts are represented on the vertical axis.

Starting from the top of the screen, the interrupts used in the system are listed first. The interrupts are followed by the processes, with the process first recording events listed first, followed by others as they record events.

The last context shown is the processor's idle thread.

context switch

An operation performed by a multitasking operating system in which the current thread of execution is switched for another. Examples of this are one process preempting another, a process delaying itself or pending on a resource (making the processor available for another thread of execution), or a process being interrupted by an ISR. *See also* current context, process context.

CPU Graph

A graph that displays data of up to X processors. To open a CPU View Graph for current data or for a saved *.irv file, choose Windows > New CPU Graph in the IRIXview main window.

CPU starvation

A state when a process is “starving” for CPU time. In other words, the process never gets to run, because it is never scheduled by the IRIX scheduler.

current context

In operating system jargon, the currently executing process and information needed to restore the process’s state, such as the state of the processor registers, operating system control information, and the stack. For IRIXview, the meaning has been extended to include ISRs and the kernel’s idle thread. *See* process context.

deadlock

When two or more processes keep each other from running; for example, when *processA* is pending on a semaphore waiting to be unblocked by *processB*, but *processB* is pending on another semaphore waiting to be unblocked by *processA*. This is a common bug in software systems, easily diagnosed with IRIXview.

event

In IRIXview, any action undertaken by a process or an ISR that can affect the state of the system. Examples of events are process spawns and deletions, timer expirations, and interrupts. IRIX has been instrumented to log this event information. *See also* event logging, IRIXview logging mode, instrumented code, user-generated events.

event buffer

An area of memory in IRIX that temporarily holds the event data before it is processed by IRIXview, when you are using the GUI to collect event data.

event data

Information that is logged to the IRIXview event buffer.

event data collection

The process of starting event logging under IRIX, and then capturing the event data to the event buffer.

event icon

An icon displayed in the View Graph that correspond to an event. For information on what each event icon represents, see the Legend window. To learn specific information on a particular instance of an icon, use the Event Inspector window.

Event Inspector window

A window that displays information about an event: its name, its timestamp, the context in which it occurred, and any other event information that has been logged. Use the middle mouse button to select any event icon in the View Graph and drag it into the Event Inspector window to display this information.

event log

A finite collection of event data that resides in shared memory. Event log data is collected using *rtmon_client* or by using the Target dialog in IRIXview (choose Windows > Target from the IRIXview menu). You can open previously saved event logs in IRIXview by choosing File > Open.

event logging

The target activity of writing information about events to the IRIXview event buffer as the events occur. You start event logging with *rtmon_client* or with the *Start* button in the IRIXview Target window (choose Windows > Target from the IRIXview menu).

executing state

The state when a process or other context has control of the processor. For a process to be in the executing state, there must be no interrupts to service. ISRs are in the executing state after their interrupt has been acknowledged by the kernel; if there is more than one ISR to service, the one at the processor's highest interrupt level executes. The idle thread is in the executing state when there are no processes to run and no ISRs to service. *See also* current context, process state, process state transition.

execution thread

See thread of execution.

GUI

Graphical user interface: the portion of IRIXview running on the system or X terminal with which you view event data.

idle thread

When there are no processes ready to execute and no interrupts to service, the kernel enters its idle thread. In this "state," the kernel services interrupts and continually checks to see if a process is ready to run. Analyzing the amount of time your application is idle can help you fine-tune the application: too much time in the idle thread may mean the application is not using the processor efficiently; too little time may mean that the application is interrupted too often to run effectively.

instrumented code

Instrumented code is software that has been modified to provide information about its own operation. In the case of IRIX, this information is event data that contains a record of the significant moments in the flow of control within the operating system. Application code can be instrumented with the `rtmon_log_user_tstamp()` routine; *see* user-generated events.

interrupt

An interrupt is a signal from hardware that lets the processor know that something has occurred in the external world. For example, the processor may receive an interrupt when a clock tick occurs or when a character is received on a serial port. *See also* Interrupt Service Routine (ISR).

interrupt handler

See Interrupt Service Routine (ISR).

interrupt latency

Interrupt latency is the amount of time during which the processor's ability to respond to interrupts is inhibited. Both the hardware and software architecture contribute to interrupt latency. Hardware influences on interrupt latency include such things as prioritizing interrupt requests and preventing interrupt handling until the completion of lengthy instructions. Software influences stem from mode switches, context switches, and kernel preemption latencies.

Interrupt Service Routine (ISR)

A routine called when a particular interrupt occurs. Also known as an interrupt handler. For example, when a character is received on a serial port, the associated ISR is called to handle that interrupt. (Handling such an interrupt typically consists of copying the input character to a buffer and clearing the serial device for the next character.) ISRs run in a special interrupt-level context, which is separate from any process's context.

interprocess communication mechanisms (IPCs)

Mechanisms that allow processes to synchronize and communicate so that they can coordinate their activity. The IRIX interprocess communication mechanisms include semaphores, message queues, pipes, sockets, and signals.

intrusion

In IRIXview, the amount of overhead added to the target IRIX system by including instrumented code and starting event logging.

IRIXview logging mode

A mode that shows the current context and where it is switched. The current context is shown as a solid horizontal line. When a context switch occurs, a dotted vertical line connects the previous context's line to the current context's line.

Legend window

IRIXview window that shows what each event icon and state stipple represents. It can be displayed with the Legend Window icon on the View Graph.

mode switch

The time it takes for a thread to switch from kernel mode to user mode.

page

In a View Graph, the width of the current time interval; that is, the portion of the event log currently displayed in the View Graph.

pended state

A state when a process attempts to obtain an object or resource but the object or resource was not available; for example, if it made a call to obtain a semaphore, but the semaphore was not available. A process in this state is also known as a blocked process. *See also* process state transition.

preemption

When a process becomes ready to execute and has a higher priority than the currently executing process, and the "new" process preempts the current process. That is, the operating system saves the current process's context and switches to the context of the higher-priority process. *See also* process state transition.

preemptive priority scheduling

In IRIX, each process is assigned a priority, and the kernel ensures that the processor is allocated to the highest-priority ready process. The scheduling is preemptive in that if a process of a higher priority than the executing process becomes ready, the kernel preempts the current process and switches to the higher-priority process. *See also* context switch, process state transition.

priority

The standard IRIX scheduler provides 256 process priority levels, numbered 0 (highest) through 255 (lowest). Processes are assigned a priority when created; however, while executing, a process may change its priority using `schedctl(2)`.

process

An independent program or application that has its own job to perform, such as the management of a robot arm. Each process has its own context, which is the processor environment and system resources the process “sees” each time the kernel schedules it to run. On a context switch, a process’s context is saved. Processes can communicate and synchronize with each other through interprocess communication mechanisms (IPCs).

process composition

A process consists of an address space containing the program text and data, and a number of process attributes managed by the IRIX kernel. A few examples of process attributes are: a unique process ID number; machine register contents, representing the current instruction and stack level as well as working data; UNIX user and group identities; current working directory for file searches; and signal-handling status.

process state

For applications, process states include the following: executing, ready, and suspended.

process state transition

This term refers to the action of a process exiting from one state and entering into another (this is different from context switch, which refers to a change in the controlling context within the processor). A process state transition may or may not result in a context switch, depending on the states of other processes in the system when the process in question makes its transition between states.

race condition

A blockage that occurs when the outcome of a process is erroneously determined by one of two or more events (for example, is a particular variable read first, or updated first?). This kind of problem can often be avoided by using mutual exclusion semaphores to synchronize the process’ use of the resource.

ready state

A state when a process is waiting for no other resources besides the processor—it is on the run queue, but has not yet executed. *See also* process state, process state transition.

resource

A system object for which a process may contend with other processes. Examples of resources are memory pool data structures, message queues, and semaphores. If the resource is not available, a process contending for that resource makes a transition to the pending state. *See also* process state transition.

running state

See executing state.

scheduling

See preemptive priority scheduling.

signal

A predefined message sent between two processes or from the kernel to a process. IRIX supports UNIX BSD-style signals as well as POSIX-compatible signals for asynchronous transfer of control within a process, based on hardware or software exceptions. The IRIX software signaling facility provides a set of 31 distinct signals. A process can specify a signal handler routine to take appropriate action when the associated signal is received. When signal handling is complete, normal process execution resumes, unless the signal corresponds to an exception.

socket

A UNIX BSD 4.3-compatible interface for transferring byte streams between processes, regardless of location in a networked application.

state

See process state.

state stipples

The horizontal lines on the View Graph that show the current state of each process in the system. The Legend window provides information about stipple types.

sub-time interval

The space between two time instants. Choose the first time instant with the left mouse button, then choose the second instant in the same way. Two vertical lines are displayed in the event log, and details about the sub-interval are displayed in the Detailed Time Information field. It can be useful to know the amount of time that has occurred between events, or to select a sub-interval that you zoom in on with the Zoom In icon.

system code

In an IRIX system, this term refers to any code that is not application code. It includes the IRIX kernel, IRIX system libraries, device drivers, and so on.

system clock

The hardware timer, which runs continuously and emits a periodic interrupt known as a tick. IRIX uses the system clock to manage process scheduling, process delays, and so on.

target

The system where IRIXview is collecting events.

thread of execution

The sequence of instructions that a particular process (such as a process or ISR) executes to carry out its job.

time instant

A single point in time, selected with a click of the left mouse button in the View Graph. A vertical line appears in the event log, and details about the time instant are displayed in the Detailed Time Information field.

time interval

The portion of the event log currently displayed in the View Graph. If timestamping is enabled, the time interval is an amount of time. If sequential event display is used, the time interval is a number of events.

time slice

The amount of time each process is normally allowed to execute without being preempted. By default, the time slice is 3 ticks, or 30 milliseconds. Often, a typical process will be blocked for I/O before reaching the end of its time slice. At the end of a time slice, the kernel chooses which process to run next on the same processor, based on process priorities. When runnable processes have the same priority, the kernel runs them in turn.

timestamp

The time recorded for an event. When the instrumented IRIX kernel is run with the accompanying *rtmond* monitoring daemon, certain logged events are tagged with a high-resolution timestamp. The events are displayed in the View Graph along a timeline showing when they occurred based on their timestamps.

transition

See process state transition.

unblocked

A process that is pended (blocked) on a resource is unblocked when the resource becomes available, its timeout expires, or it is explicitly unblocked.

user-generated events

Custom application-specific events that IRIXview can record and display. When event logging has been started, you can have IRIXview show these events by inserting calls to the `rtmon_log_user_tstamp()` function into your application source code.

user interface

See GUI.

Index

A

- About IRIXview command, 61
- allocate events, displaying, 64
- analyzing event data, 37
- annotate events, displaying, 64
- architecture
 - host-side activities, 4
 - target-side activities, 3

C

- chkconfig and rtmnd daemon, 7
- commands
 - irixview*, 4
 - IRIXview GUI, 59, 72, 83
 - locating, 60
 - About IRIXview, 61
 - Display Events, 62-64
 - Event Inspector, 65
 - Exchange icon, 17
 - Legend Window icon, 18, 66
 - New Graph, 60, 68, 69
 - Open, 70, 71
 - Pan icons, 17
 - Pop icon, 17, 72
 - Push icon, 17, 72
 - Quit, 5, 73
 - Save, 73-74
 - Search Accelerator icons, 17, 76
 - Search Window icon, 17, 76-78

- Target, 8, 80
- Time Units Menu icon, 81
- View Control Window icon, 81-83
- View Options Window icon, 26
- Zoom icons, 17, 83
- context switches, 2, 35
- Context Switch events, 35
 - current context, 35
- conventions
 - typographical, xiv
- current context, 2, 35

D

- defaultUser* event, 55
- disk events
 - displaying, 64
- Display Events command, 62-64
- documentation conventions, xiv

E

- event buffer
 - overflow, 10
- event data, 2
 - analyzing, 37
 - displaying, 4
 - examining, 30
 - importing, 7
 - event buffer overflow, 10

- event data collection
 - event buffer
 - overflow, 10
 - starting and stopping, 80
 - event data importing, 7
 - event icons, 19, 33, 35, 45
 - definitions for, 18, 66
 - Event Inspector window, 33, 65
 - using, 65
 - event logging, 3
 - see also* Context and Mode Switch events, Process State Transition events, user events
 - see also* event logs
 - context switches, 35
 - event buffer overflow, 10
 - Process State Transitions, 35
 - timestamping, 3
 - event logs
 - creating, 21
 - opening, 21, 70, 71
 - and *rtmon-client*, 11
 - saving, 73-74
 - events, 33-36, 45-57
 - see also individual events*
 - displaying, 62-64
 - interrupt service routine (ISR), 48-49
 - process, 53
 - selecting, 29
 - signal, 51
 - sub-time intervals, 28
 - time instants, 27
 - time intervals, 26
 - unknown, 55
 - Exchange icon, 17, 72
 - exiting (IRIXview), 5, 6, 73

F

- File menu, 5, 71
 - event files, saving, 73-74
 - event logs
 - opening, 70
 - exiting IRIXview, 73
 - Open Event File window, displaying, 70
- FRS events
 - displaying, 64

H

- help
 - product, xv
 - support, xv
- Help menu, 5
 - version and copyright information, displaying, 61

I

- icon bar (View Graph), 60
 - Exchange icon, 17, 72
 - Legend Window icon, 18, 66
 - Pan icons, 17
 - Pop icon, 17, 72
 - Push icon, 17, 72
 - Search Accelerator icons, 17, 76
 - Search Window icon, 17, 29, 76
 - Time Units Menu icon, 81
 - View Control Window icon, 16, 27, 83
 - Zoom icons, 17, 83
- icons, *see* event icons, icon bar, and individual View Graph icon bar icons

idle thread, 19
intEnt event, 48
interrupt events, 63
 displaying, 64
interrupt service routines (ISR), 48-49, 64
 entrances, 63
 exits, 63
interrupt transitions, 63
 displaying, 64
intExit event, 49
IRIXview
 events, 45-57
 exiting, 5, 6, 73
 Main window, 5
irixview command, 4
 IRIXview GUI commands, locating, 60
IRIXview GUI
 see also commands, event data collection, event
 data importing, File menu, Windows menu,
 Help menu
 commands, locating, 60
irixview program
 see irixview command

K

kill event, 52

L

Legend Window icon, 18, 66

M

menus, *see* File menu, Help menu, Windows menu

N

net events
 displaying, 64
New Graph command, 60, 68, 69
 see also View Graphs
nonexecuting states
 displaying, 64

O

Open command, 70
Options menu, 5

P

page, 17, 27, 72, 82
Pan icons, 17
par command for event data collection, 12
Pop icon, 17, 72
processDelete event, 53
process events, 53
 displaying, 64
Process State Transition events, 35
 state stipples, 35
 timestamping, 36
product support, xv
profiling events
 displaying, 64
Push icon, 17, 72

Q

Quit command, 5, 73

R

rtmon-client tool, 11
rtmond daemon and chkconfig, 7

S

Save command, 73-74
Search Accelerator icons, 17, 76
searching, 17
 accelerators, using, 17
 contexts, 76-78
 events, 76, 76-78
 objects, 76-78
Search Window icon, 17, 29, 76-78
 using, 76
signal events, 51
 displaying, 64
Signal Receive event, 51
state stipples, 19, 35
 definitions for, 18, 66
stipples, *see* state stipples
sub-intervals, *see* sub-time intervals
sub-time intervals
 cancelling, 29
 selecting, 28
syscall events
 displaying, 64
system clock ticks, 63

T

Target command, 8, 80
TCP/IP, 4
time instants
 cancelling, 29
 selecting, 27

time intervals

 doubling, 17
 exchanging, 17
 halving, 17
 moving, 17, 27, 72, 82
 popping, 17
 pushing, 17
 selecting, 26

Timeline

 for View Graphs, 20

timestamps, 36

 resolution, 3

Time Units Menu icon, 81

transition lines

 displaying, 64

typographical conventions, xiv

U

unknown event, 55
user-defined event icon, 55
user events, 36, 55
 displaying, 64

V

View Control

 events, displaying, 62-64
 states, displaying, 62-64
 time intervals
 selecting, 26

View Control Window icon, 16, 81-83

window

 displaying, 16, 81-83
 zooming in and out, 27, 83

View Graphs, 20, 33
 displaying, 68, 69
 event icons, 19, 33
 definitions for, 18, 66
 events, displaying, 62-64
 Exchange icon, 17, 72
 icon bar, 60
 see also icon bar and individual icon bar icons
 idle thread, 19
 Legend window, displaying, 18
 Pan icons, 17
 Pop icon, 17, 72
 processes in priority order, listing, 19
 Push icon, 17, 72
 refreshing, 68, 69
 scrolling, 20
 search accelerators, 17
 searching, 76, 76-78
 Search window, displaying, 17
 states, displaying, 62-64
 state stipples, 19
 definitions for, 18, 66
 time information, displaying detailed, 20
 time instants, selecting, 27
 time intervals, displaying, 68, 69
 Timeline, 20
 timeline, 3
 time units, specifying, 81
 View Control window, displaying, 16, 27, 83
 zooming, 17, 83

View Options
 View Options Window icon, 26
 window
 displaying, 26

VM events
 displaying, 64

W

Windows menu, 5
 event data collection, starting and stopping, 80
 Event Inspector window, using the, 65
 event logging, setting up for, 8
 event logging mode, choosing, 80
 event port numbers, examining, 80
 targets, specifying RPC request, 80
 View Graphs, displaying, 68, 69

Z

Zoom icons, 17, 30, 83
 for View Control, 27, 83
 View Graphs, using, 17, 83

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-2824-002.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
 - On the Internet: techpubs@sgi.com
 - For UUCP mail (through any backbone site): *[your_site]!sgi!techpubs*
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

Technical Publications
Silicon Graphics, Inc.
2011 North Shoreline Boulevard, M/S 535
Mountain View, California 94043-1389

