SpyKer User’s Guide

SpyKer Release 1.0

DOC-0392-00D

Contents

Preface

Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to filenames and commands are case sensitive and should be typed accurately.

Font and Description	Examples
Garamond 10 pts. - body text; *italicized* for emphasis, new terms, book titles, and text that would be replaced with a real name or value (9 pts. used for table text)	Refer to the *LynxOS User's Guide*. `cat` *filename* `mv` *file1 file2*

`Courier New` 9 pts. (used within body text) - commands, options, filenames, parameter names, pathnames, and other computer-generated data	`ls`
	`-l`
	`myprog.c`
	`/dev/null`

`Courier New 7 pts.` - blocks of text that appear on the display screen after entering instructions or commands	see below

```
Loading file /tftpboot/shell.kdi
into 0x4000

.....................

File loaded. Size is 1314816

Copyright 2001 LynuxWorks, Inc.

All rights reserved.


LynxOS (ppc) created Mon Jan 22
17:50:22 GMT 2000

user name:
```

Font and Description	Examples

`Courier New Bold` 9 pts. (used within examples) - command lines and options entered on the computer	`login:` **`myname`**
	`cd /usr/home`

VERDANA 9 pts. (all caps) - environment variables	DISPLAY
Verdana Bold 9 pts. - keyboard options, button names, and menu sequences	**Enter, Ctrl-C**
Verdana Italics 9 pts. - function or method names	*getenv()*

Special Notes

The following notations highlight any key points and cautionary notes that may appear in this manual.

NOTE: *These callouts note important or useful points in the text.*

CAUTION! *Used for situations that present minor hazards that may interfere with or threaten equipment/performance.*

Technical Support

LynuxWorks Technical Support is available Monday through Friday (holidays excluded) between 8:00 AM and 5:00 PM Pacific Time (U.S. Headquarters) or between 9:00 AM and 6:00 PM Central European Time (Europe).

The LynuxWorks World Wide Web home page provides additional information about our products, Frequently Asked Questions (FAQs), and LynuxWorks news groups.

LynuxWorks U.S. Headquarters

Internet: **`mailto:support@lynuxworks.com`**

Phone: (408) 879-3940
Fax: (408) 879-3945

LynuxWorks Europe

Internet: **`mailto:tech_europe@lynuxworks.com`**
Phone: (+33) 1 30 85 06 00
Fax: (+33) 1 30 85 06 06

World Wide Web

`http://www.lynuxworks.com`

CHAPTER 1

Introduction and Installation

Introduction

Thank you for purchasing a license to use SpyKer, LynuxWorks, Inc.'s advanced event tracing tool for LynxOS. Future releases of SpyKer will support Linux. SpyKer is designed to provide the maximum power to diagnose and debug embedded systems based on LynxOS while providing an easy to use interface and setup.

This chapter reviews how to install and execute SpyKer, which is composed of two major parts:

- Development Host Display Tool
- Target System Data Capture

The Development Host is the computer system on which LynxOS software is being developed. This can be a Microsoft Windows-based PC, a Linux-based PC, or a Solaris-based Sun workstation. The target system is the computer system where LynxOS is going to run. To fully utilize SpyKer, both parts must be properly installed and operating. Please refer to "Installing on a Development Host" on page 2 and "Installing on LynxOS Target Systems" on page 5 for complete installation instructions.

Installing on a Development Host

Installing on Microsoft Windows

NOTE: *When installing on Windows NT, the user must be in the administrator group for the installation to succeed. If a non-administrator performs the procedure, the installation fails trying to add SpyKer to the Start menu. If this occurs, exit the installation, uninstall the product, add the user to the administrator group, and reinstall.*

1. Insert the SpyKer CD-ROM into the CD-ROM reader.
2. InstallShield (`Setup.exe`) should automatically execute, but if it does not, double click on:

   cdrom`:Setup.exe`

3. Follow the instructions given in the subsequent installation dialog boxes.

Installing on Linux or Solaris Hosts

1. Insert the SpyKer CD-ROM into the CD-ROM reader.
2. Mount the CD-ROM, on **`/mnt/cdrom`**, for example.

NOTE: *On some systems, the mount automatically occurs upon inserting a CD-ROM. If this does not happen, the user may have to become* `root` *or superuser to successfully mount the CD-ROM. For example, to mount the CD-ROM on a Linux system, use this command:* **`mount -r /dev/cdrom /mnt/cdrom`**

3. Create a directory in which to install SpyKer. This directory may be located anywhere, but LynuxWorks recommends installing SpyKer where it may be easily found. For example, on Linux systems it should be installed under the `/usr/local` directory. To create this directory, execute:

```
# mkdir /usr/local/spyker
```

4. Change directory to the SpyKer installation directory. For example:

   ```
   # cd /usr/local/spyker
   ```

5. Locate the file `spyker.tar` on the CD-ROM and execute the following command:

   ```
   # tar -xvf /mnt/cdrom/spyker.tar
   ```

6. To execute the SpyKer display tool, execute the following commands:

   ```
   # cd /usr/local/spyker/

   # ./spyker
   ```

Changing Default Behavior

By default, the current working directory is searched to locate all necessary SpyKer display tool components that comprise the executable. To change this default behavior, do the following:

1. Edit the file spyker in the installation directory. Locate the line:

   ```
   java spyker 20 . netscape
   ```

2. Substitute this line with the following:

   ```
   export CLASSPATH=$CLASSPATH:/usr/local\
   /spyker
   java spyker 20 installation_dir/spyker netscape
   ```

Where *installation_dir* is the directory that is created in step 3 of “Installing on Linux or Solaris Hosts” on page 2.

NOTE: *To run SpyKer without typing the full pathname, simply add the SpyKer installation directory to the execution path environment variable of the appropriate* `shell` *command.*

For the license key mechanism to support more than one user on a single development host, each user needs a private installation of the SpyKer display tool.

The SpyKer display tool on a Linux development host requires JRE 1.3 (Java Runtime Environment) from Sun Microsystems, which can be downloaded from the **`www.java.sun.com`** web site.

To ensure that the correct version of JRE will be run, insert the path component `/usr/java/jdk1.3/jre/bin` to the execution path environment variable. This path component needs to proceed the `/bin` component.

Uninstalling from the Development Host

Uninstalling from Microsoft Windows Hosts

To uninstall SpyKer from Microsoft Windows Hosts, do the following:

1. From the Windows control panel select **Add/Remove** programs.
2. Select the SpyKer software.
3. Select **Add/Remove**.

Uninstalling from Linux or Solaris Hosts

To uninstall SpyKer, simply remove the entire contents of the installation directory. For example:

```
# rm -fr /usr/local/spyker
```

All saved settings or data files in this directory are lost.

Installing on LynxOS Target Systems

Installing from CD-ROM

SpyKer may be installed on a running LynxOS target system with a CD-ROM reader and hard disk.

1. Insert the SpyKer CD-ROM into the CD-ROM reader.
2. Mount the CD-ROM, to an available mount point of **/mnt** or **/mnt/cdrom**, for example

   ```
   # mount -o ro /dev/cdrom /mnt
   ```

3. Create a directory to install the SpyKer target software. This directory may be located anywhere, but LynuxWorks recommends installing SpyKer under the `/usr/local` directory. To create this directory, execute:

   ```
   # mkdir /usr/local/spyker
   ```

4. Change directory to the SpyKer installation directory. For example:

   ```
   # cd /usr/local/spyker
   ```

5. Locate the file `LYNXOS/`*target*`target.tar` on the CD-ROM and execute the following command (for LynxOS running on the Intel IA-32 processor, for example):

   ```
   # tar -xvf /mnt/LYNXOS/X86TAR~1.TAR
   ```

   Where *target* may be any one of those in the following table.

Table 1-1: LynxOS Targets

Target	Description
x86	Intel IA-32 or x86 processor family
ppc	Motorola/IBM PowerPC processor family
pquicc	Motorola PowerQUICC processor family

Installing from the Host System

In those cases where the LynxOS target system does not have a CD-ROM reader, installing SpyKer's target components is done from the host development system (Microsoft Windows, Linux, or Solaris).

The first step is to locate the appropriate target system `tar` file on the SpyKer CD-ROM. On Microsoft Windows systems, the appropriate `tar` file is (assuming the CD-ROM reader is drive **`D:`**):

```
D:\LynxOS\targettarget.tar
```

On Linux or Solaris development hosts, the appropriate `tar` file is found under the following directory (assuming the CD-ROM is still mounted at `/mnt/cdrom`):

```
/mnt/cdrom/LynxOS/targettarget.tar
```

Where *target* may be any one of those in the following table.

Table 1-2: LynxOS Targets

Target	Description
`x86`	Intel IA-32 or x86 processor family
`ppc`	Motorola/IBM PowerPC processor family
`pquicc`	Motorola PowerQUICC processor family

Depending on the target system's configuration, proceed to one of the following, "Target Systems with No Hard Disk" on page 6 or "Target Systems with Hard Disk" on page 7.

Target Systems with No Hard Disk

For target systems without any hard disks, SpyKer components must be installed in the proper directory of a kernel downloadable image (KDI) root file system. Refer to the appropriate LynxOS documentation for specific instructions. The files that need to be installed are found in the appropriate target `tar` file as described previously. Locate the appropriate `tar` file and execute a command on the host system to extract the files. For instance, the `tar` command for LynxOS for an Intel IA-32 target would be:

```
# tar -xvf /mnt/LYNXOS/X86TARr~1.TAR
```

The required files are now located in the current folder and/or directory. If using Microsoft Windows, refer to the appropriate LynxOS manuals for details on finding and executing the `tar` command.

Target Systems with Hard Disk

For target systems with a hard disk, the appropriate target `tar` file, as described above, needs to be placed on the target systems' hard disk. How this is accomplished is target-system dependent. For example, if the target system is Ethernet enabled, including a full TCP/IP protocol stack and applications, the `tar` file can be transmitted to the target system using `ftp` or other file transfer mechanism. See the appropriate LynxOS documentation on how to set up the local area network and how to use `ftp` or other networking tools.

Once the appropriate `target.tar` file is located on the target system, follow the same steps as installing from CD-ROM, except use the pathname for `target.tar` file on the target system's hard disk instead of that on the CD-ROM.

CHAPTER 2

Operational Overview

SpyKer Trace Tool

SpyKer belongs to a class of software development tools known as tracing tools. Tracing tools, typically, provide a system developer the ability to record or capture operating system and application events in a running system, and to display these events in a graphical user interface (GUI). By capturing event trace data and displaying it with an easy-to-read GUI, trace tools provide system developers with powerful system tuning and debugging capabilities.

Some tracing tools are based on in-circuit emulators and other hardware devices, but these tools are expensive and difficult to learn and use. SpyKer is a software-only trace tool. Unlike most software-only trace tools, SpyKer does not require manually inserting event capture code into the software being traced. It does not even require the software being traced to be recompiled and relinked. For the LynxOS kernel, SpyKer can be installed and executed on a running system without any prior preparation or configuration, except for required memory and disk space. SpyKer can even be installed and executed on production software in shipping products.

The SpyKer trace tool has been designed from the ground up to support high performance event tracing of the LynxOS kernel and applications. SpyKer is composed of two major parts:

- Development Host Display Tool
- Target System Data Capture

To take full advantage of SpyKer's power and flexibility it is important to fully understand its design and architecture. This chapter provides an overview of SpyKer and how it is used.

Theory of Operation

SpyKer captures event trace data in a unique fashion. Taking advantage of LynxOS dynamic device driver installation, SpyKer's event trace capture software installs itself into a running LynxOS kernel. Once installed, SpyKer is prepared to take over certain critical function entry and exit points. During a trace, SpyKer intercepts the execution of these functions. Upon each interception, SpyKer records the event that has occurred, the time it occurred, and additional data, as needed. This additional data is known as an event's *payload*. For example, one entry point that is intercepted is the system call interface. The payload is the number of the system call that is being invoked.

By default, SpyKer intercepts and captures trace data for a large number of critical functions, including system calls, interrupts, return from interrupts, context switches, and so on. See Appendix B, "SpyKer Events and Payloads" for a complete list of predefined event types.

Once the SpyKer driver is installed on the target system, it must be configured to tell it which event types to capture. This configuration process may be done in a number of ways, but the simplest is to use the SpyKer display tool on the development host system. This easy-to-use program provides a graphical user interface to configure SpyKer event capture.

The simplest configuration of SpyKer is to capture all events. However, this approach causes the greatest impact on the target system. This impact, called *intrusion*, represents the amount of system overhead that is incurred when capturing event trace data. There are two forms of intrusion: *performance intrusion* and *space intrusion*. Most trace tools do not provide any way to configure or tune intrusion to get the most faithful and useful event capture data. For example, most trace tools enable capturing all events or none at all. When all events are being captured, system performance is impacted by all of the data collection activity and the amount of memory required to store the trace data. This high level of intrusion may affect other system activities and may substantially alter performance characteristics and timing.

SpyKer, on the other hand, is designed to minimize intrusion in several ways. It allows the user to specifically set the amount of memory to use for captured event data. The size of this buffer determines the total amount of trace data that may be captured. Reducing the size of this buffer limits the amount of trace data that is collected, but it also reduces space intrusion.

SpyKer can also be configured to buffer captured event data to disk files or network interfaces. In this case, both performance and space intrusion is increased, but virtually an unlimited amount of event trace data can be captured.

Additionally, SpyKer can be configured to minimize performance intrusion by capturing event trace data on selected events only. The amount of intrusion for events not being captured is zero because SpyKer does not intercept those locations in the target system.

Minimizing intrusion is not something that is required every time SpyKer is used to capture event data. In many cases, the problem or tuning issue that is being pursued can be found when SpyKer is capturing all events. However, since capturing all events maximizes intrusion, it is possible that the intrusion may mask or avoid the problem or tuning issue being pursued. For example, since system timing changes with the level of performance intrusion, any timing-related problems are affected and therefore may be masked unless low levels of performance intrusion is selected. The overhead of a trace point is about 0.4 microseconds on a 433 MHz Pentium system.

Chapter 5, "Basic Capture Features" and Chapter 6, "Advanced Capture Features" discuss how to configure and run SpyKer event data capture. Chapter 5 discusses basic, most commonly used, event capture techniques. Chapter 6 discusses advanced event capture techniques, including postmortem event trace capture.

Once event data is captured and transmitted to the development host system, the SpyKer display tool can be used to display the information in a graphical format. The display tool allows the user to examine and measure captured events. Chapter 3, "Running the SpyKer Display Tool" explains in detail how to use the SpyKer display tool's graphical user interface to configure event capture and to inspect and evaluate trace data.

CHAPTER 3

Running the SpyKer Display Tool

Invoking the Display Tool

The SpyKer display tool may be invoked from Microsoft Windows, Linux, or Solaris. Use one of the procedures below to invoke the display tool.

Invoking on Microsoft Windows

From Microsoft Windows, the SpyKer display tool is invoked on the development host system as follows:

1. Go to the **Start** menu.
2. Select **Programs**.
3. Select **LynuxWorks, Inc.**
4. Select **SpyKer**.

Invoking on Linux or Solaris

To invoke the SpyKer display tool on Linux or Solaris, choose one of the two options that follow:

Type the full pathname of the executable file, for example:

```
/usr/local/spyker/spyker
```

Or:

Include the directory containing SpyKer in the shell's execution path environment variable by simply typing:

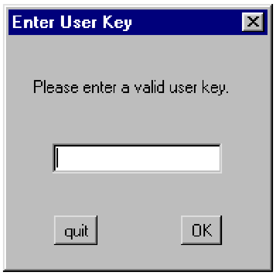
```
spyker.
```

The SpyKer display tool is written in Java and uses the system's Java Virtual Machine (JVM). On some systems, the JVM may have to be upgraded if problems are encountered.

Entering the User Key

The first time the SpyKer display tool is invoked, it asks for a user key. A user key is provided with the SpyKer CD-ROM and can be found printed on a label attached to the back of the CD-ROM jewel case. This key must be used if SpyKer is reinstalled, so care should be taken to store this key in a safe location for later use.

Enter the user key in the **Enter User Key** dialog box (shown below) and click on the **OK** button. The key is saved and it does not need to be re-entered.



Please enter a valid user key.

The SpyKer Main Window

Upon startup, the SpyKer display tool displays its main window containing the following components as illustrated in the figure below.

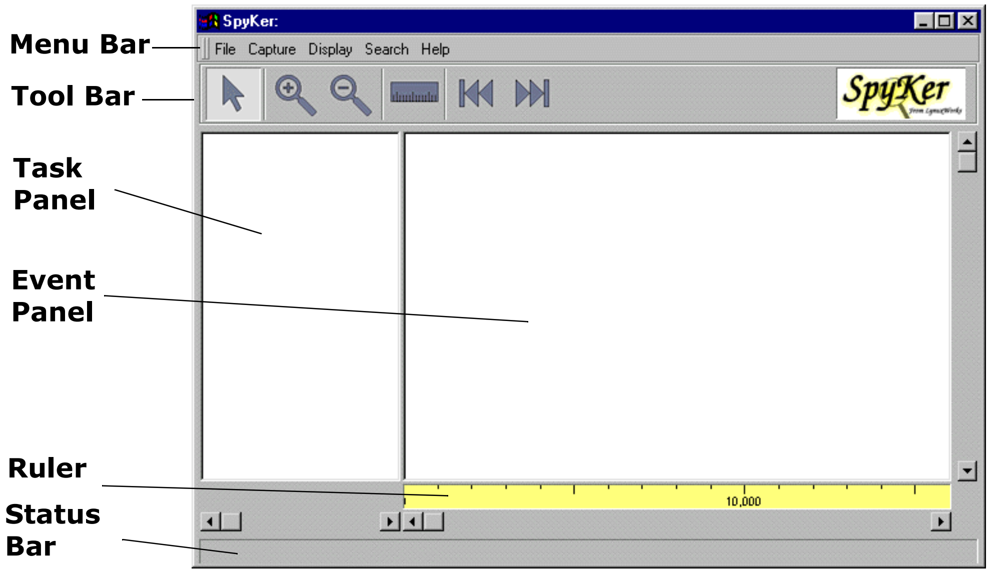


Table 3-1: Main Window Components

Component	Description
Menu bar	The menu bar contains pull down menus of SpyKer commands.
Tool bar	The tool bar contains a set of buttons for commonly used SpyKer commands.
Task panel	The task panel shows the names and IDs of processes and threads involved in the captured events.

Table 3-1: Main Window Components

Component	Description
Event panel	The event panel shows a graphical display of the captured event data.
Ruler	The ruler shows the time offset from the start of the event trace, in microseconds.
Status Bar	The status bar shows information about commands, search status (if no event is found) and other useful information.

The SpyKer main window also includes the normal window manager icons such as those for close window, minimize, and maximize. It also includes three scroll bars, shown below:

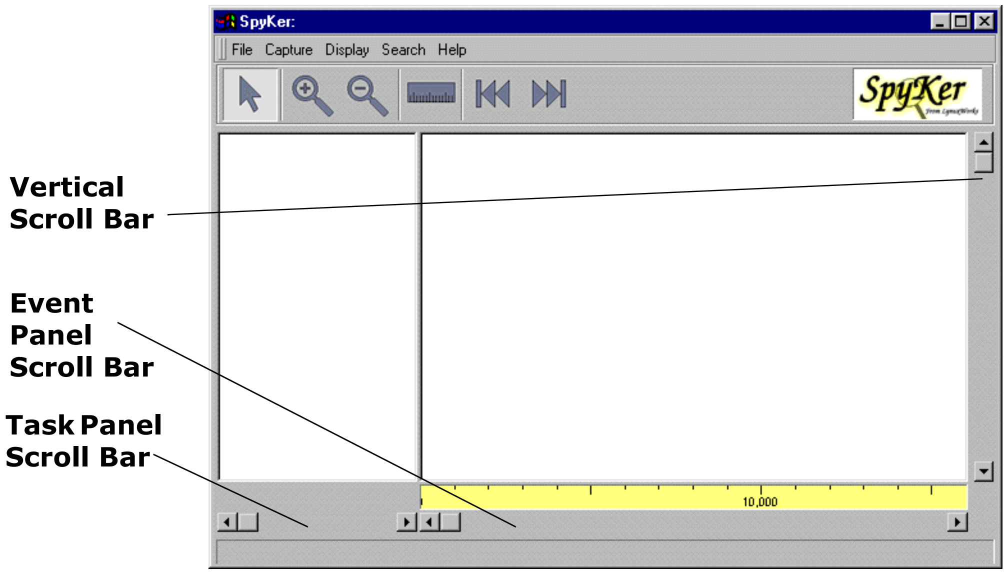


Table 3-2: Additional Main Window Components

Component	Description
Vertical scroll bar	This scroll bar, on the right of the main window, scrolls the task panel and event panel up and down to show all processes and threads.
Event panel scroll bar	This scroll bar, on the bottom right of the main window, scrolls the event panel and ruler to show earlier and later time periods of the event traces.
Task panel scroll bar	This scroll bar, on the bottom left of the main window, scrolls the task panel right and left to see the entire names and IDs of processes and threads.

Loading a Sample Trace File

To learn how the various menu and tool bar commands work, it is helpful to use some captured event trace data. Included with SpyKer is a sample trace file for this purpose. To load the sample trace data, do the following:

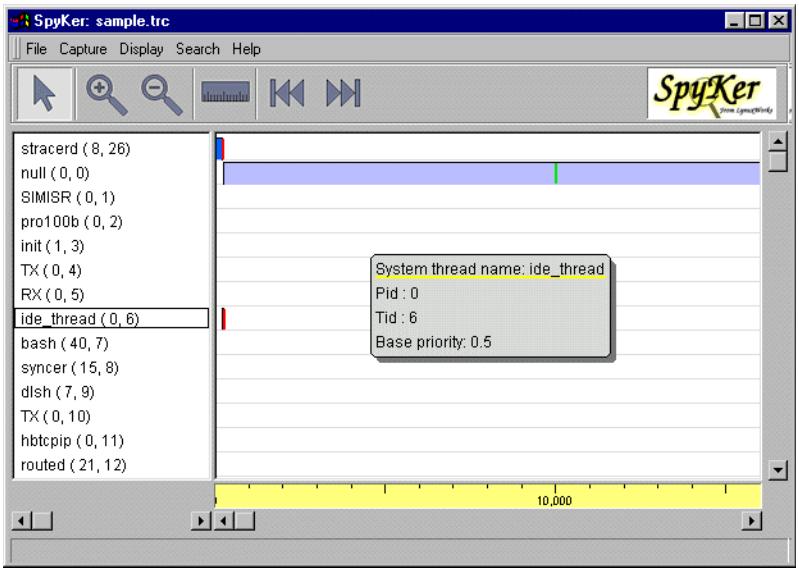
1. Click on the **File** menu.
2. Select the **Open** command.
3. Using the file selection dialog, navigate to the SpyKer installation director or folder and select the file `sample.trc`.
4. Click on the **Open** button.

When SpyKer loads the sample trace data, it immediately shows the trace data in the event panel and the process and thread ids in the task panel.

The Task Panel

The task panel shows the processes and threads involved in a trace. In this document, the word "task" refers to either a process or a thread. Each task is identified by name, followed by the process id and thread id, respectively. Under some conditions, such as when a user configures SpyKer not to capture certain events, a task name may not be captured. If this occurs, only the process and thread ids are displayed.

Tasks are initially displayed in order of thread id, except the SpyKer daemon. It is possible to rearrange the order of tasks to better organize event trace data in the event panel. To move a task up or down, use the mouse to move the cursor over a task in the task panel, click and hold the left mouse button, and drag the task up or down in the list. To view detailed task identification information, using the left mouse button, click on the desired task in the task panel. In the following figure, detailed task information for the system thread named *ide_thread* is displayed.



stracerd (8, 26)
null (0, 0)
SIMISR (0, 1)
pro100b (0, 2)
init (1, 3)
TX (0, 4)
RX (0, 5)
ide_thread (0, 6)
bash (40, 7)
syncer (15, 8)
dlsh (7, 9)
TX (0, 10)
hbtcpip (0, 11)
routed (21, 12)

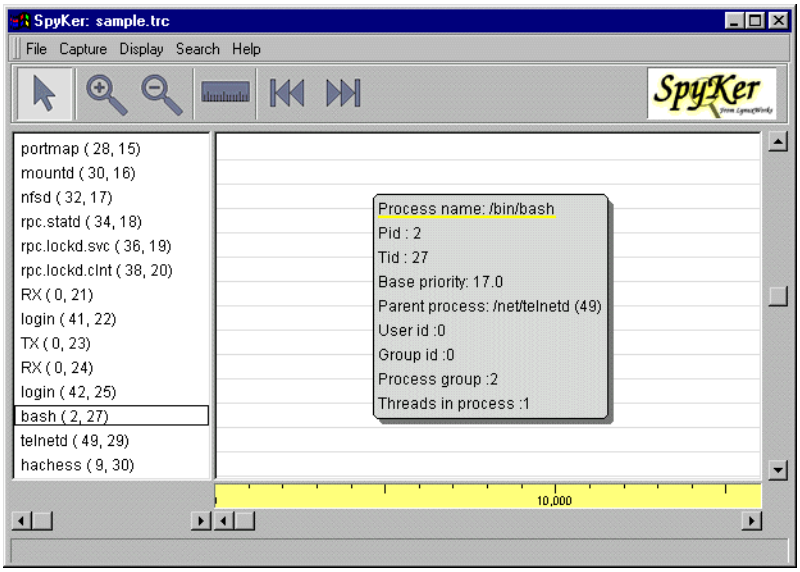
System thread name: ide_thread
Pid : 0
Tid : 6
Base priority: 0.5

For system threads, the following information is displayed.

Table 3-3: System Thread Display Information

Display Information	**Description**
Thread name	This matches the information in the task panel
Process id	This matches the information in the task panel.
Thread id	This matches the information in the task panel.
Base Priority	This shows the system thread's base, or initial, priority used by the system scheduler to dispatch threads. The thread's actual priority at any given time may change during system execution depending on system activities.

The following figure shows detailed task information for the user task named *bash*.



portmap (28, 15)
mountd (30, 16)
nfsd (32, 17)
rpc.statd (34, 18)
rpc.lockd.svc (36, 19)
rpc.lockd.clnt (38, 20)
RX (0, 21)
login (41, 22)
TX (0, 23)
RX (0, 24)
login (42, 25)
bash (2, 27)
telnetd (49, 29)
hachess (9, 30)

Process name: /bin/bash
Pid : 2
Tid : 27
Base priority: 17.0
Parent process: /net/telnetd (49)
User id :0
Group id :0
Process group :2
Threads in process :1

For user tasks, in addition to the information displayed for system threads, the following information is displayed.

Table 3-4: User Task Display Information

Display Information	Description
Parent process	This shows the name and id of the task's parent process.
User id	This shows the user id of the task. The user id is used for enforcing access controls and may be used to include or exclude event captures (see Chapter 5, "Basic Capture Features" for configuring event captures). User id 0 indicates `root` or superuser.
Group id	This shows the group id of the task. The group id is used for enforcing access controls and may be used to include or exclude event captures. Group `id 0` indicates the `root` or superuser group.
Process group	This shows the task's process group id. This may be used to include or exclude event captures.
Threads in process	Indicates the number of threads that are part of this process.

The Event Panel

The event panel is where captured events are displayed. For each task, a horizontal event bar shows all events that are captured for that task. When context switch events are captured, SpyKer shades the event bar to show the task that is executing at each point in time. Captured events are displayed in the event bars as color coded vertical lines.

NOTE: *When the event panel is zoomed out, two or more captured events may appear to the user to be only one event. To ensure that there is not more than one event at a location, the user should zoom in to the highest possible zoom level.*

The operation of the event panel is dependent upon the tool that is selected. There are six tools available as shown in the tool bar.

Table 3-5: Event Panel Tools

Tool Icon	Tool Name	Description
↖	**Info**	This tool is used to display information about individual events.
🔍	**Zoom In**	This tool is used to zoom in on the event display. Zooming in increases the magnification of the event display showing progressively more details, but smaller time spans.
🔍	**Zoom Out**	This tool is used to zoom out on the event display. Zooming out decreases the magnification of the event display showing progressively fewer details but larger time spans.
📏	**Measure**	This tool is used to measure the time, in microseconds, between selected points.
⏮	**Search Left**	This tool, with the leftward pointing double arrow, searches for captured events backwards along the time ruler (searching from the current location backwards towards the beginning of the trace).
⏭	**Search Right**	This tool, with the rightward pointing double arrow, searches for captured events forward along the time ruler (search from the current location forward towards the end of the trace).

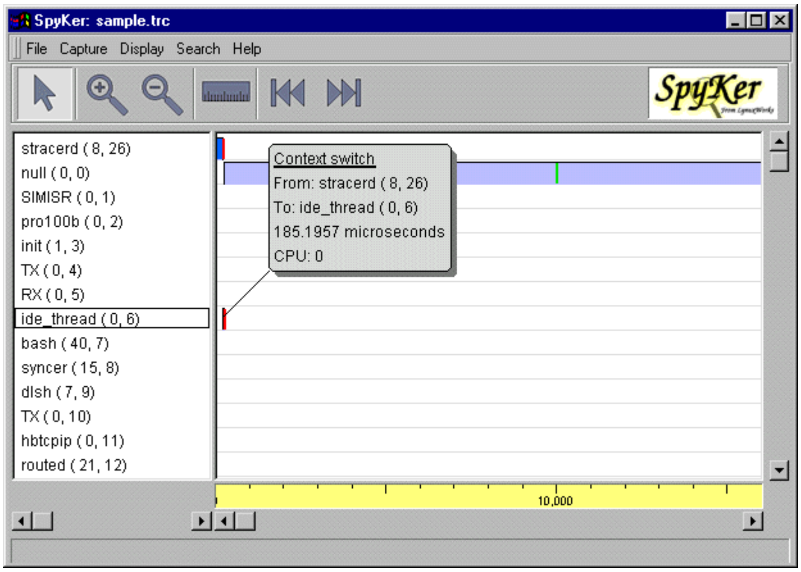
Detailed explanations of the event panel's behavior under these tools is described below. To use any of the event panel's tools, simply click on the tool icon in the tool bar.

Info Tool

The **Info Tool** provides detailed information about individual captured events. When using the **Info Tool**, the cursor in the event panel looks like an arrowhead.

Event Data

To view captured event data, use the mouse to move the cursor over an event vertical bar (including context switch bars). A pop-up window displays the event data. The following image shows the event data for a selected context switch event from the *ide_thread* task.



stracerd (8, 26)
null (0, 0)
SIMISR (0, 1)
pro100b (0, 2)
init (1, 3)
TX (0, 4)
RX (0, 5)
ide_thread (0, 6)
bash (40, 7)
syncer (15, 8)
dlsh (7, 9)
TX (0, 10)
hbtcpip (0, 11)
routed (21, 12)

Context switch
From: stracerd (8, 26)
To: ide_thread (0, 6)
185.1957 microseconds
CPU: 0

The exact data that is displayed is dependent on the event type. However, the event data pop-up window contains the following information:

Table 3-6: Event Data Display Information

Display Information	**Description**
Event type name	See Appendix B, "SpyKer Events and Payloads" for a list of SpyKer predefined event types. The event type will be underlined with a color coded line. The color coding matches the color coding of the event.
Thread name **Process id** **Thread id**	This data matches the thread data in the applicable task in the task panel. On a context switch, the task ids for both the old task (**From**) and the new task (**To**) are displayed.
Timestamp	This is the time that the event occurred, in microseconds, since the start of the trace.
Data	This shows the data payload for the event. See Appendix B, "SpyKer Events and Payloads" for a list of SpyKer predefined event types and their default payloads.

Scrolling

When using the **Info Tool**, the event panel can be scrolled right and left, and up and down by clicking anywhere in the event panel and dragging the cursor right or left, or up and down. The event panel may also be scrolled right or left at any time by using the event panel scroll bar below the ruler. It may be scrolled up and down at any time by using the vertical scroll bar to the right of the event panel. Additionally, the arrow keys on the keyboard can be used to scroll in any direction.

Zoom In and Zoom Out Tools

The **Zoom In** and **Zoom Out** tools allow the user to zoom in and zoom out the captured event trace data. When using the zoom tools, the cursor in the event panel looks like crosshairs.

To zoom in or zoom out using the zoom tools, simply click anywhere in the event panel. In both zoom in and zoom out modes, SpyKer keeps the selected time location fixed in the event panel. For example, if a user selects the **Zoom In** tool and clicks on a particular event, after the zoom in operation, the selected event continues to be under the cursor (minor “movement” may occur due to screen resolution).

The event panel can be zoomed in or zoomed out at any time, even when not using the zoom tools, by using the **Zoom In** or **Zoom Out** commands in the **Display** menu. These commands zoom in or out fixing the time location at the left side of the event panel.

NOTE: *For performance reasons, SpyKer may not zoom out far enough to see all captured events. At the zoom out limit, the event panel may still need to be scrolled to view certain events.*

Measure Tool

The **Measure** tool allows a user to measure the time, in microseconds, between selected locations in the event trace. The cursor looks like crosshairs when using the **Measure** tool.

To measure the time between two locations in the event panel, simply move the cursor to either the starting or ending location, click and hold the left mouse button, and drag the cursor to the other location. The **Measure** tool draws a line between the first and second locations and displays the time difference between these points.

The **Measure** tool is very useful for checking the time intervals between critical events, such as when an interrupt occurs and when the resulting return-from-interrupt occurs.

Search Tools

The **Search** tools provide powerful event search capabilities. When using the **Search** tools, the cursor resembles a two-headed arrow on Windows hosts and a left or right arrow on UNIX and Linux hosts. If the event found as a result of a search is touched with the information tool cursor, the vertical line will disappear. If the vertical line appears on the screen, a search starts from the found event, regardless of the cursor’s horizontal position.

Default Searches

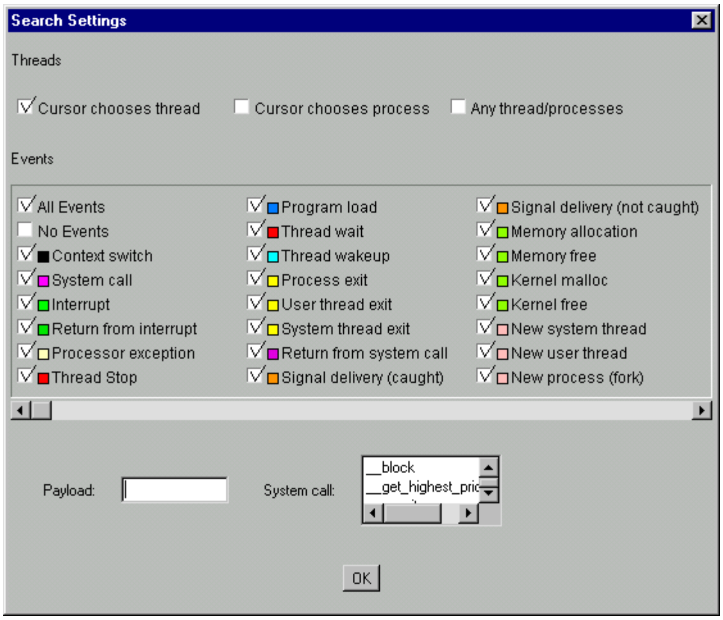
In the default mode, the **Search** tools search left or right (depending on which **Search** tool is selected), looking for the first event associated with the task, in the task panel to the left of the cursor. That is, in this mode the search is performed on a per-task basis. For instance, to find the next event associated with a particular task, select the **Search Right** tool, use the mouse to move the cursor to the event bar to the right of the selected task and click.

When using the **Search** tools, if an event is found that matches the search criteria, then that event is displayed in the event panel attached to a vertical line, from the top of the event panel to the bottom. The event panel automatically scrolls right and left, and up and down, to display the matched event. If no events are found that match the search criteria, then the event panel does not change.

Advanced Searches

The **Search** tools use the **Search Settings** dialog box to look for events in the trace. The **Search Settings** default to perform searches on a per-task basis and to search for any event type. However, the user can change the **Search Settings** to provide more selective searching.

To change the **Search Settings**, select the **Search Settings** command in the **Search** menu. This command brings up the **Search Settings** dialog box, shown below:



Search Settings

Threads

[✓] Cursor chooses thread [ ] Cursor chooses process [ ] Any thread/processes

Events

[✓] All Events	[✓] 🟦 Program load	[✓] 🟧 Signal delivery (not caught)
[ ] No Events	[✓] 🟥 Thread wait	[✓] 🟩 Memory allocation
[✓] ⬛ Context switch	[✓] 🟦 Thread wakeup	[✓] 🟩 Memory free
[✓] 🟪 System call	[✓] 🟨 Process exit	[✓] 🟩 Kernel malloc
[✓] 🟩 Interrupt	[✓] 🟨 User thread exit	[✓] 🟩 Kernel free
[✓] 🟩 Return from interrupt	[✓] 🟨 System thread exit	[✓] 🟥 New system thread
[✓] 🟨 Processor exception	[✓] 🟪 Return from system call	[✓] 🟥 New user thread
[✓] 🟥 Thread Stop	[✓] 🟧 Signal delivery (caught)	[✓] 🟥 New process (fork)

Payload:

System call: __block, __get_highest_pric

OK

The **Search Settings** dialog box has three parts (from top to bottom):

- Thread mode selector.
- Events selector.
- Payload selector.

Thread Mode Selector

The thread mode selector allows the user to choose from one of three search modes:

Table 3-7: Thread Mode Selector Search Modes

Search Mode	Description
Cursor chooses thread	In this mode, the **Search** tools only search for events associated with the thread, in the task panel to the left of the cursor. This is the default mode when SpyKer is started or a new main window is created.
Cursor chooses process	In this mode, the **Search** tools search for events associated with the thread, in the task panel to the left of the cursor, and all other threads belonging to the same process.
Any threads/processes	In this mode, the **Search** tools search for events regardless of thread or process. Use this mode to step event-by-event through the trace.

NOTE: *In* **Cursor chooses process** *mode, the event panel may scroll vertically, possibly leaving the cursor over a thread belonging to another process. To repeat the search on the originally selected process, the cursor may need to be moved.*

Events Selector

The **Events** selector is used to select the event types for which to search. See Appendix B, “SpyKer Events and Payloads” for a list of SpyKer predefined events and default payloads. The **Events** selector shows a list of all SpyKer event types by name, with their color coding color, that conforms to the color coding used in the event panel, and a check box. Note that the **Events** selector scrolls right and left to display all SpyKer event types.

The first two event types shown are labeled **All Events** and **No Events**. If the **All Events** box is checked, by moving the cursor over the box and clicking the left mouse button, then all events (except **No Events**) are selected. In this state, which is the default state when SpyKer starts, the **Search** tools search for all events. If the **No Events** box is checked, then all event types, except **No Events** are de-selected. In this state, the **Search** tools do not search for any events and all searches fail.

NOTE: *The* **All Events** *and* **No Events** *boxes are used to quickly select all event types or none. They are not event types themselves.*

Use the **Event** selector to select the event types for which to search. The **Search** tools search only for those event types that are selected.

Payload Selector

The **Payload** selector lets the user further refine a search by specifying a specific event payload for which to search. See Appendix B, “SpyKer Events and Payloads” for a list of predefined SpyKer event types and payloads. If the **Payload** selector is blank, which is the default state when SpyKer starts, then an event’s payload is ignored during searches. However, if a payload value is entered, then searches look for only those events whose payloads match the entered value.

NOTE: *If a search payload is specified and if multiple event types are selected, then all events must match both a selected event type and the specified payload. Since the meaning of payloads differs from event type to event type, specifying a search payload may not be meaningful when searching for multiple event types.*

To the right of the **Payload Selector** is scrolling list of system call names. This list can be used to specify a search payload that matches the named system call. For instance, to search for the *read()* system call event, scroll down the **System Call** menu until the *read()* system call is visible, and then double click on it. The search payload `23` is automatically entered into the payload selector. The LynxOS system call number for the *read()* system call is `23`.

Performing Advanced Searches

When performing advanced searches by changing the **Search Settings**, the **Search** tools find only those events that match all three conditions: thread mode, selected events, and payload. In this way, searches can be narrowed down to specific threads, processes, event types, and payloads.

NOTE: *Once the Search Settings are modified, the Search Tools continue to use these settings until they are modified again. The Search Settings are associated with a SpyKer main window. When a SpyKer main window is closed, the Search Settings are discarded.*

CHAPTER 4

SpyKer Display Tool Menus

This chapter reviews the SpyKer menus and commands.

File Menu

The File menu contains commands for opening trace files, closing SpyKer main windows, saving trace data, and quitting SpyKer.

Open

The **File** menu **Open** command is used to open an event trace file and load the event trace data into a SpyKer main window. SpyKer uses the convention of ending all trace file names with the suffix `.trc`. Use the **Open** command to start the **Open Trace File** dialog. In the **Open Trace File** dialog box, navigate directories or folders and select the desired trace file. If the current main window is displaying trace data, SpyKer creates another main window to display the file.

Close

The **File** menu **Close** command is used to close the SpyKer main window. If the main window is displaying trace data that has not yet been saved, SpyKer asks whether the trace data should be saved. If the user chooses to save the trace data, the **Save Trace File** dialog starts. From this dialog, select a directory or folder and a file name, and click on **Save**.

Save As

The current trace data, whether newly captured or loaded from a trace file, can be saved at any time by using the **File** menu **Save As** command. This command starts the **Save Trace File** dialog box. From this dialog, select a directory or folder and a file name, and click on **Save**.

Save As Text

Normally SpyKer saves trace data in a compact format. The **File** menu **Save As Text** command saves the trace data in a text editor friendly text format. Data saved in this form cannot be reloaded into SpyKer.

Quit

The **File** menu **Quit** command causes all SpyKer main windows to be closed and SpyKer to exit. If any main windows contain trace data that is not yet saved, SpyKer asks whether the data should be saved.

Capture Menu

The **Capture** menu contains only the Connect command. See Chapter 5, "Basic Capture Features" and Chapter 6, "Advanced Capture Features" for information on the trace capture.

Display Menu

The **Display** menu contains the following commands:

New Display

The **Display** menu **New Display** command creates a new SpyKer main window displaying the same trace data that is in the current SpyKer main window. Using the **New Display** command, a user can open multiple SpyKer main windows to view a single trace or multiple traces. Viewing a single trace in multiple windows is very useful when comparing or viewing multiple events that are separated by long time gaps.

Zoom In

The **Display** menu **Zoom In** command zooms in the event panel fixing the time location at the left hand side of the event panel.

Zoom Out

The **Display** menu **Zoom Out** command zooms out the event panel fixing the time location at the left hand side of the event panel.

NOTE: *For performance reasons, SpyKer may not zoom out far enough to see all captured events. At the zoom out limit, the event panel may still need to be scrolled to view certain events.*

Search Menu

The **Search** menu contains the **Search Settings** command. The **Search Settings** command is described in detail under the event panel **Search Tool** on page 24 in Chapter 3, “Running the SpyKer Display Tool.”

Help Menu

The **Help** menu contains commands to access on-line help information, including this document, keyboard shortcuts, and the SpyKer copyright window.

Key Shortcuts

The **Help** menu **Key Shortcuts** command displays a window showing SpyKer keyboard shortcuts. See Appendix C, “Keyboard Shortcuts” for a list of the SpyKer display tool’s keyboard shortcuts.

Documentation

The **Help** menu **Documentation** command starts a Web browser to display this manual. On Microsoft Windows, SpyKer uses the default Web browser. On Linux or Solaris, SpyKer uses the Netscape browser by default. To change this option, edit the `spyker` shell script in the SpyKer installation directory.

About SpyKer

The **Help** menu **About SpyKer** command brings up the SpyKer copyright window.

CHAPTER 5

Basic Capture Features

Capturing trace data using SpyKer is exceptionally easy. This chapter explains how to perform the most commonly used captures. The next chapter explains how to perform advanced trace captures that require more complex configuration and setup.

Preparing a Target for Trace Capture

The trace capture software on the target system comprises two parts:

- Dynamically loaded device driver, `stracedrvr`: this device driver performs the actual event tracing.
- SpyKer event trace daemon, `stracerd`: this daemon initializes the SpyKer dynamically loaded device driver and communicates the with the SpyKer display tool on the development host system. See Chapter 3, "Running the SpyKer Display Tool" for more information.

NOTE: *It is possible to perform event tracing without using the SpyKer display tool. This capability is described in Chapter 6, "Advanced Capture Features"*

To perform event capture under the control of the SpyKer display tool on the development host system, the SpyKer device driver must be loaded and the SpyKer daemon must be started. To load the driver, on the target LynxOS system execute the following commands (as `root` or superuser):

```
# cd /usr/local/spyker/spykertarget
# ./install_spyker
```

CAUTION: *Only execute* `install_spyker` *when the driver is not already installed. Running* `install_spyker` *on a target system with the driver already installed may cause system crashes.*

This script creates a special device node called `tr` that is used by the SpyKer daemon to communicate with the SpyKer device driver. To start the daemon, execute the following commands:

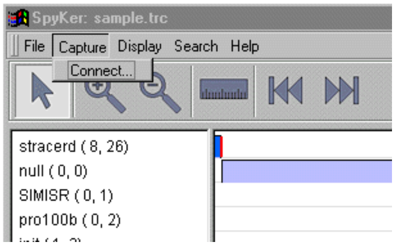
```
# cd /usr/local/spyker

# ./stracerd -n tr
```

The `-n` option to `stracerd` tells the SpyKer daemon to connect to the SpyKer display tool over a network to initialize and control the SpyKer device driver. See Chapter 6, “Advanced Capture Features” for additional `stracerd` options.

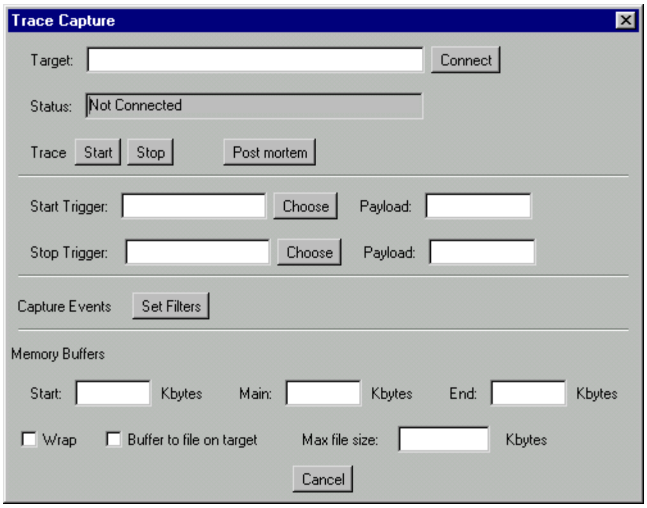
In order to exit the `stracerd` session, type **Ctrl-\** on the keyboard.

Basic Trace Capture

On the host system, start the SpyKer display tool and select the **Connect** command from the **Capture** menu, as shown below.



This command starts the **Trace Capture** dialog box, shown below.



Enter the name of the target system, as known to the network name server, or the target system's IP address and click on the **Connect** button. The **Status** box shows the connection status. If the connection to the target fails, check the following:

- The correct system name or IP address is being used.
- The target system is running and connected to the network. This can be done by remotely logging into the target system using `telnet` or by logging into the remote system console and `ping` the development host system. See LynxOS documentation on setting up and managing network connections.
- The SpyKer daemon is running.
- The SpyKer driver is installed.

If no errors were found in the above items, reboot the LynxOS system, reinstall the SpyKer driver, and restart the SpyKer daemon. If this still does not clear the connection problem, contact LynuxWorks Technical Support for assistance.

Once a connection is successfully opened with the target system, SpyKer will display the current default buffer sizes. See "Memory Buffer Management" on page 44 to change these settings.

NOTE: *Any changes to the memory buffer sizes prior to connecting to a target are discarded.*

Try running a simple trace. Without entering any other data or clicking on any other buttons, click on the **Start** button, wait a few seconds, and then click on the **Stop** button. If the event panel is already displaying trace data, then a new main window is created with the newly captured trace data. If the event panel is not already displaying trace data, then the newly captured trace data is displayed without opening a new main window.

Congratulations! The first trace is successfully completed. Unless there are non-trivial programs running on the target system, the trace data is minimal.

To view a more complex trace, repeat the steps above, but before clicking on **Stop**, execute some simple commands, such as `ls` and `ps`, on the target system. This trace displays more events.

SpyKer supports the ability to finely control event capture. The remainder of this chapter explains the most commonly used settings. Chapter 6, "Advanced Capture Features" reviews advanced subjects, including postmortem event capture.

NOTE: *All* **Trace Capture** *dialog box settings are preserved on a SpyKer main window by main window basis. When re-running a trace capture, all settings from the previous trace capture are used unless modified prior to starting a new trace capture. When SpyKer opens a new main window to display captured trace data, the* **Trace Capture** *dialog box settings for the new main window are set to the default settings.*

Setting Event Triggers

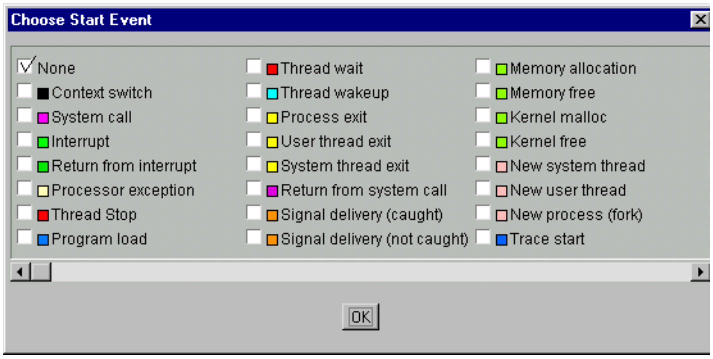
SpyKer includes the ability to automatically start and stop event capture based on event triggers. There are two triggers:

- Start trigger - The specified event, with the specified payload if present, starts event capture on the target system.
- Stop trigger - The specified event, with the specified payload if present, stops the event capture on the target system.

Start and stop triggers are not used by default. They must be configured by the user.

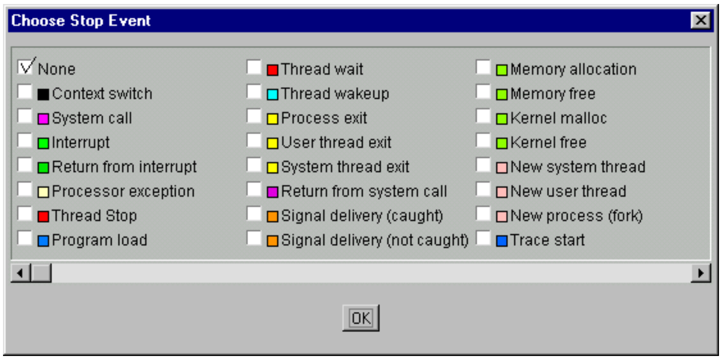
To configure start and stop triggers for a trace, enter the number of the desired event type in the text boxes to the right of **Start Trigger** and **Stop Trigger**, respectively. To select event types by name, simply click on the **Choose** button to the right of the desired text box. This starts the **Choose Start Event** or **Choose Stop Event** dialog box, as appropriate. In the **Choose Event** dialog box, click on the desired event type or **None** if no trigger is desired. Only one event type can be selected.

The following figure shows the **Choose Start Event** dialog box.



☑ None	☐ 🟥 Thread wait	☐ 🟩 Memory allocation
☐ ■ Context switch	☐ 🟦 Thread wakeup	☐ 🟩 Memory free
☐ 🟪 System call	☐ 🟨 Process exit	☐ 🟩 Kernel malloc
☐ 🟩 Interrupt	☐ 🟨 User thread exit	☐ 🟩 Kernel free
☐ 🟩 Return from interrupt	☐ 🟨 System thread exit	☐ 🟥 New system thread
☐ 🟨 Processor exception	☐ 🟪 Return from system call	☐ 🟥 New user thread
☐ 🟥 Thread Stop	☐ 🟧 Signal delivery (caught)	☐ 🟥 New process (fork)
☐ 🟦 Program load	☐ 🟧 Signal delivery (not caught)	☐ 🟦 Trace start

The next figure shows the **Choose Stop Event** dialog box.


Choose Stop Event

[✓ None]	🟥 Thread wait	🟩 Memory allocation
■ Context switch	🟦 Thread wakeup	🟩 Memory free
🟪 System call	🟨 Process exit	🟩 Kernel malloc
🟩 Interrupt	🟨 User thread exit	🟩 Kernel free
🟩 Return from interrupt	🟨 System thread exit	🟥 New system thread
🟨 Processor exception	🟪 Return from system call	🟥 New user thread
🟥 Thread Stop	🟧 Signal delivery (caught)	🟥 New process (fork)
🟦 Program load	🟧 Signal delivery (not caught)	🟦 Trace start

Try the following experiment to illustrate the start and stop triggers. From the **Trace Capture** dialog box:

1. Set the start trigger by clicking on the **Choose** button to the right of **Start Trigger**. In the **Choose Start Event** dialog box, select **Program load**, and click on the **OK** button. The **Program load** event type number, 6, now is in the **Start Trigger** text box.

2. Set the stop trigger by clicking on the **Choose** button to the right of **Stop Trigger**. In the **Choose Stop Event** dialog box, select **Process exit**, and click on the **OK** button. The **Process exit** event type number, 9, is now in the **Stop Trigger** text box.

3. Connect to the target system and click on the **Start** button. This starts the trace session but no events are captured until a program load event occurs.

4. On the target system, execute an `ls` command. This triggers the trace start when the `ls` command loads and it triggers the trace to stop when it exits.

5. At this point, SpyKer should show a complete trace from the instant the start trigger occurred to the instant the stop trigger occurred.

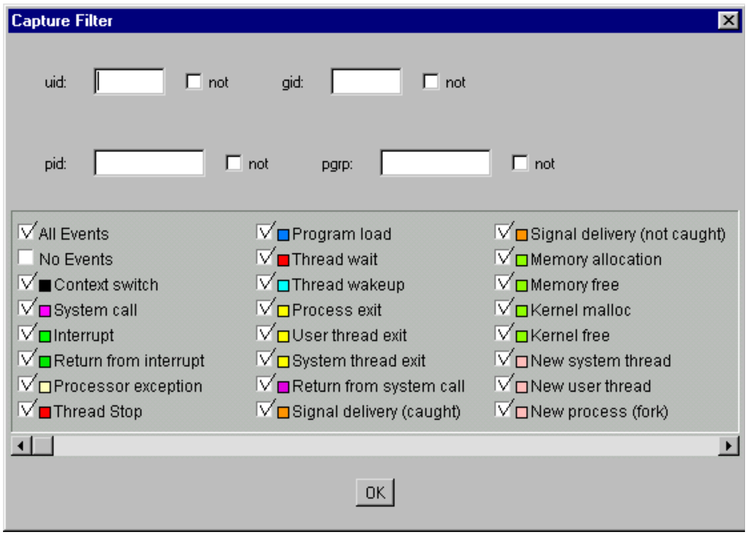
Both the start trigger and the stop trigger can be further qualified by including a payload value on which to trigger. See Appendix B, “SpyKer

Events and Payloads" for a list of SpyKer predefined events and default payloads.

Capture Events

By default, SpyKer captures all events. See Appendix B, "SpyKer Events and Payloads" for a list of predefined SpyKer events and default payloads.

A powerful feature of SpyKer is the ability to capture only those events of interest, and therefore reduce overhead and intrusion. To select event types to capture in a trace, click on the **Set Filters** button in the **Trace Capture** dialog box. This starts the **Capture Filter** dialog box as shown in the following figure.


Capture Filter

uid: ☐ not gid: ☐ not

pid: ☐ not pgrp: ☐ not

✓ All Events	✓ 🟦 Program load	✓ 🟧 Signal delivery (not caught)
☐ No Events	✓ 🟥 Thread wait	✓ 🟩 Memory allocation
✓ ⬛ Context switch	✓ 🟦 Thread wakeup	✓ 🟩 Memory free
✓ 🟪 System call	✓ 🟨 Process exit	✓ 🟩 Kernel malloc
✓ 🟩 Interrupt	✓ 🟨 User thread exit	✓ 🟩 Kernel free
✓ 🟩 Return from interrupt	✓ 🟨 System thread exit	✓ 🟥 New system thread
✓ 🟨 Processor exception	✓ 🟪 Return from system call	✓ 🟥 New user thread
✓ 🟥 Thread Stop	✓ 🟧 Signal delivery (caught)	✓ 🟥 New process (fork)

The **Capture Filter** dialog box allows the user to capture only those events that meet all of the specified criteria. The filter criteria are described in the table below.

Table 5-1: Filter Criteria

Criteria	Description
uid	This is the user id of the currently executing thread. Enter the user id in the text box. If a user id is entered, SpyKer captures only events from threads with that user id. If the **not** check box is selected, SpyKer only captures events from threads whose user id is not the entered user id. If no user id is entered, which is the default case, then SpyKer captures events for all user ids.
gid	This is the group id of the currently executing thread. Enter the group id in the text box. If a group id is entered, SpyKer captures only events from threads with that group id. If the **not** check box is selected, SpyKer captures only events from threads whose group id is not the entered group id. If no group id is entered (the default case), then SpyKer captures events for all group ids.
pid	This is the process id of the currently executing thread. Enter the process id in the text box. If a process id is entered, SpyKer only captures events from threads with that process id. If the **not** check box is selected, SpyKer only captures events from threads whose process id is not the entered process id. Care should be taken when using this filter criteria since process ids may not be predictable under all circumstances. If no process id is entered (the default case), SpyKer captures events for all processes.

Table 5-1: Filter Criteria (Continued)

Criteria	Description
pgrp	This is the process group id of the currently executing thread. Enter the process group id in the text box. If a process group id is entered, SpyKer only captures events from threads with that process group id. If the **not** check box is selected, SpyKer only captures events from threads whose process group id is not the entered process group id. Care should be taken when using this filter criteria since process group ids may not be predictable under all circumstances. If no process group id is entered (the default case), SpyKer captures events for all processes.
events	From the list of event types, select which ones to capture. Click on **All Events** (the default case) to select all event types except **No Events**. Click on **No Events** to deselect all event types. Click on individual event types to select or deselect them. SpyKer will only capture selected event types.

The **All Events** and **No Events** boxes are used to quickly select all event types or none. They are not event types themselves.

If **No Events** is selected and no individual events are subsequently selected, SpyKer does not capture any events and the trace is empty.

NOTE: *For SpyKer to show full process and thread ids in the task panel, the* **Existing process** *and* **Existing thread** *events must be selected.*

When all desired event types are selected, and all desired ids are entered, click on the **done** button to return to the **Trace Capture** dialog box. Until modified again, all **Capture Filter** settings are used for all new traces started from this **Trace Capture** dialog box and main window.

Memory Buffer Management

Upon capturing event trace data, SpyKer saves the data in kernel memory buffers managed by the SpyKer driver. SpyKer manages three separate buffers:

- Start buffer (disabled by default)
- Main buffer (always enabled)
- End buffer (disabled by default)

Main Buffer

By default, the Start buffer and End buffers are disabled (size set to zero) and all event capture data is placed in the Main buffer. After data is placed in the Main buffer, one of three possible actions occur:

- The SpyKer daemon empties the Main buffer after event tracing is finished and transmits the data over the network to the SpyKer display tool on the host development system. If event capture occurs so fast that the SpyKer daemon cannot keep up, overruns occur in the Main buffer. A buffer overrun occurs when the Main buffer becomes full and new event data is captured. The new event is lost. This is the default action. Under normal circumstances, the SpyKer daemon empties the Main buffer sufficiently fast to avoid overruns.
- The SpyKer daemon empties the Main buffer while event tracing is occurring and writes the data to a file on a hard disk in the target system. If event capture occurs so fast that the SpyKer daemon cannot keep up, overruns occur in the Main buffer. Buffering trace data to a hard disk creates less system overhead and intrusion than using the network to send data back to the development host. To enable writing trace data to a hard disk, click on **Buffer to file on target** in the **Trace Capture** dialog box. Captured trace data is written to the file `out.trc` on the target system in the directory where the SpyKer daemon is executing. To prevent the target system's hard disk from becoming full, set a maximum file size in the **Trace Capture** dialog box. Enter the size, in Kbytes, of the maximum trace file size permitted into the **Max file size** text box. If no size is specified (the default case), then the file

size is not limited by SpyKer. If a maximum file size is entered, then no additional event captures occur after the maximum size is reached.

- The Main buffer is configured to "wrap" and new captured event data is allowed to overwrite the oldest event data in the buffer. When the trace stops, the SpyKer daemon, or other program sends the captured event data to a hard disk file or to the SpyKer display tool on the development host. Allowing the Main buffer to wrap substantially reduces system overhead and intrusion since the daemon does not need to empty the buffer while event tracing is occurring. To enable wrapping in the Main buffer, click on **Wrap** in the **Trace Capture** dialog box. If the Start buffer size is zero, it will be automatically set to 40 Kbytes.

NOTE: *Whenever the Main buffer is configured to "wrap," the Start buffer should be enabled to capture SpyKer trace header information.*

The size of the Main buffer is configurable on a per trace basis. To change the size of the Main buffer, enter the desired size, in Kbytes, in the **Main** text box of the **Trace Capture** dialog box.

NOTE: *The Main buffer size must be at least 1 Kbyte.*

The default Main buffer size is 976 Kbytes. See "maketraceinfo" on page 49 for changing the default Main buffer size.

Start Buffer

When enabled, SpyKer uses the Start buffer to save captured event trace data at the start of the trace, including SpyKer trace header information. If a start trigger is enabled, then event trace captures do not start until the trigger event occurs. In either case, SpyKer places captured event trace data in the Start buffer until the buffer is full. After that, SpyKer places newly captured event trace data into the Main buffer.

Captured event trace data in the Start buffer cannot be overrun and is preserved regardless of whether data in the Main buffer is intentionally wrapped or accidentally overrun.

To enable the Start buffer, enter the desired size of the Start buffer, in Kbytes, in the **Start** text box of the **Trace Capture** dialog box.

End Buffer

When enabled, SpyKer uses the End buffer to save captured event trace data after the End event trigger occurs. SpyKer places captured event trace data in the End buffer until the buffer is full.

Enabling the End buffer lets SpyKer capture event trace data both before and after the End event trigger. The Main buffer contains the captured event trace data up to and including the End event trigger, and the End buffer contains captured event data afterward. The End buffer cannot be overrun.

To enable the End buffer, enter the desired size of the End buffer, in Kbytes, in the **End** text box in the **Trace Capture** dialog box.

CHAPTER 6

Advanced Capture Features

Postmortem Trace

Among the most difficult problems to diagnose and resolve in embedded systems is when the system hangs or crashes without a clear indicator of the cause. SpyKer can be used to help diagnose these situations by capturing event trace data right up to point of the hang or crash and then later loading that data in to the SpyKer display tool for analysis.

Configuring and Running Postmortem Trace

SpyKer can be configured to capture event trace data and store it in non-volatile persistent RAM to allow postmortem analysis of system crashes or hangs. To accomplish this task requires five steps:

1. Configure the SpyKer driver on the target system to save captured event trace data to a memory region that will not be erased or corrupted across a system reset. On typical Intel IA-32 based PCs, this requires the addition of an external PCI memory board. On many other systems, it is possible to configure the system firmware and LynxOS to avoid using certain memory regions which will not be erased or corrupted during a reset and reboot. Detailed directions on how to configure the SpyKer driver is given below.
2. Use the SpyKer display tool to configure and start an event trace. Do not use the **Postmortem** button, do not set memory buffer sizes, and do not select the **Buffer to file on target** option. All other **Trace Capture** options are available.

3. Run the target system until the system hang or crash occurs, reset the target system hardware, and reboot the operating system.
4. Configure the SpyKer driver on the target system with the same settings as in step 1.
5. Use the SpyKer display tool to read and display the trace data up to the point of the crash. To do this, start the **Capture Trace** dialog box by executing the **Connect** command in the **Capture** menu. After connecting to the target system, click on the **Postmortem** button.

NOTE: *The* **Postmortem** *button can be used at any time to upload and display the last captured event trace data in the SpyKer trace buffers, if any.*

Configuring the SpyKer Driver for Postmortem Trace

To setup the SpyKer driver for postmortem trace, execute the following commands on the target system as `root` or superuser.

```
# cd /usr/local/spyker
# ./maketraceinfo tracedev
# ./install_spyker
```

The `maketraceinfo` program prompts for the following information:

- Start buffer size
- Main buffer size
- End buffer size
- Address of non-volatile memory to use as trace buffer

For postmortem analysis, the Start buffer should be at least one Kbyte. No end buffer is required so the End buffer size can be zero (End buffer disabled).

The `install_spyker` script loads the SpyKer driver and configures it to use the configured non-volatile persistent memory. As long as the buffer sizes and the location of the non-volatile RAM don't change, it is not necessary to rerun `maketraceinfo` for each postmortem trace.

NOTE: *To return to normal, non-postmortem tracing, rerun* `maketraceinfo` *and do not enter a non-volatile memory address.*

Once the SpyKer driver is installed and configured, then the SpyKer daemon needs to be started by executing:

```
# cd /usr/local/spyker
# ./stracerd -n tr &
```

The `-n` option to `stracerd` tells the SpyKer daemon to connect to the SpyKer display tool over the network to initialize and control the SpyKer device driver.

maketraceinfo

The `maketraceinfo` program can also be used to change the default Main buffer size for normal, non-postmortem tracing. To change the default Main buffer size to 256 Kbytes, execute the following commands:

```
# cd /usr/local/spyker
# ./maketraceinfo tracedev
Start buffer size: 0
Main buffer size: 262144
End buffer size: 0
Address of non-volatile memory to use as trace
buffer (0 for none): 0
# ./install_spyker
```

NOTE: *The buffer values in the example above are provided as number of bytes.*

The default Main buffer size continues to be 256 Kbytes unless it is changed back again using the `maketraceinfo` command.

Trace Command File

SpyKer normally uses the SpyKer display tool to configure and run traces. However, there are circumstances where the target system is inaccessible to the host development system. For example, if the target system does not have a LAN connection, then it is impossible to use the SpyKer display tool to configure and run traces.

SpyKer supports the ability to configure traces on the target system without the use of the SpyKer display tool. The SpyKer daemon `stracerd` can be executed and provided a command file (a simple text file) to configure and start traces. The `stracerd` command file has the following syntax: a series of commands with the final command being **`:start`**:

```
command
command
command
...
command
:start
```

To start a trace with a command file *command_file*, execute the following:

```
# cd /usr/local/spyker
# ./stracerd -c command_file tr &
```

If the *command_file* is `-`, then `stracerd` reads commands from its standard input. In this case, do not fork the `stracerd` command. For example: execute:

```
# cd /usr/local/spyker
# ./stracerd -c - tr
```

stracerd Commands

In the command descriptions below, the following conventions are used. All other syntax in the command descriptions must be entered as shown.

Table 6-1: Command Description Conventions

Convention	Meaning
`...`	A list of values.
`[ ]`	Denotes optional items; they are not literally typed in. If the optional item is not provided, nothing should be entered. If an options item is provided, the square brackets are not included.
event	Any legal event type number, in decimal. See Appendix B, "SpyKer Events and Payloads" on 59 for a list of SpyKer predefined events and default payloads.
payload	Any 32-bit payload value, in decimal.
max_file_size	Maximum size of the trace file in Kbytes, in decimal.
start_buffer _size	Start buffer size in Kbytes, in decimal. If the Start buffer size is `0`, the Start buffer is disabled.
main_buffer_size	Main buffer size in Kbytes, in decimal. The Main buffer size cannot be `0`.
end_buffer_size	End buffer size in Kbytes, in decimal. If the End buffer size is 0, the End buffer is disabled.
–	Indicates that the user id, group id, process id, or process group id is to be excluded from, rather than included in, captured events.
user_id	Any valid user id on the target system, in decimal.
group _id	Any valid group id on the target system, in decimal.
process_id	Any valid process id on the target system, in decimal.

Table 6-1: Command Description Conventions (Continued)

Convention	Meaning
process_group	Any valid process group id on the target system, in decimal.
#_event_masks	The number of event masks that follow, in decimal. This number must be less than or equal to `16`. Only two event masks are required to select or deselect the predefined SpyKer event types described in Appendix B, "SpyKer Events and Payloads," since two event masks cover up to 64 event types.
event_mask	A 32-bit mask representing 32 consecutive event types, in hexadecimal. A one bit selects an event type and a zero bit deselects it. For example, `0000000F` selects event types `0`, `1`, `2`, and `3`.
buffer_wrapping_state	If `1`, turn on wrapping. If `0`, turn off wrapping.

The following commands are supported by the SpyKer daemon `stracerd`.

NOTE: *For each of the commands below, there is a space between the terminating "`;`" and the preceding argument.*

To set a Start event trigger:

:strig *event* **[***payload***] ;**

To set a Stop event trigger:

:etrig *event* **[***payload***] ;**

To set the maximum trace file size:

:file [*max_file_size***] ;**

To set buffer sizes:

:buffers [*start_buffer_size main_buffer_size end_buffer_size***] ;**

To set a capture filter:

:filter [uid [-] *user_id***] [gid [-]** *group_id***] \**

[pid [-] *process_id***][pgrp [-]** *process_group***] \**

[events *#_event_masks ... #_event_masks***] ;**

To configure wrapping in the **Main** buffer:

:wrap b*uffer_wrapping_state* **;**

To start tracing (this must be the final command):

```
:start
```

For postmortem tracing, substitute the following command for the **:start** command:

```
:postmortem
```

User Event Capture

SpyKer has been designed to automatically intercept kernel functions to capture events. In addition to these events, SpyKer can be used to capture events manually, referred to as user events, by adding calls to the SpyKer *__trace()* function (two underscores). The *__trace()* function can be called from either kernel or application code.

To capture user events in the kernel, the SpyKer driver must be statically linked to the kernel. See the appropriate LynxOS documentation on how to statically link drivers.

To capture user events in application code, the file `ustrace.o` must be linked into the application. To avoid changing `make` files, `ustrace.o` can be added to the standard C run-time library. For example, on LynxOS the following commands add `ustrace.o` to the standard C run-time library:

```
# ar r /lib/libc.a ustrace.o
# ranlib /lib/libc.a
```

To capture user events, call the *__trace()* command with the arguments shown below:

```
void __trace(int event_type_number,
             int short_payload,
             char *long_payload,
             int long_payload_length );
```

The SpyKer display tool interprets the long payload of user events to be a null terminated character string. Thus, to account for the length of the string and the null terminator:

```
long_payload_length = strlen(long_payload)+1;
```

The maximum long payload length is `252` bytes, including the null terminator. If there is no long payload, the long payload length should be `0`.

The SpyKer display tool predefines 9 user event types, 37 through 45. See Appendix B, "SpyKer Events and Payloads," for a list of SpyKer predefined event types. If additional event types are required, custom events can be created. See "Creating Custom Events" below.

Creating Custom Events

SpyKer predefines event types 0 through 45, with event types 37 through 45 allocated to user events. The SpyKer driver and daemon support event types 0 through 511. The SpyKer display tool only supports event types 0 through 45 in its default mode. The SpyKer display tool can be configured to recognize and display event types greater than 45.

To add custom event types, edit the file `eventnames.txt` found in the SpyKer installation directory or folder on the development host system. Only add custom event names to the end of the file. The syntax for each line is:

event_name color_code

Where *event_name* can be any normal character text including spaces and *color_code* is a six digit hexadecimal constant in RGB format. For custom events, LynuxWorks recommends using RGB code `00c0c0`.

NOTE: *A backup copy of* `eventnames.txt` *should be created before attempting to edit the file. If event types are inserted into the* `eventnames.txt` *anywhere but at the end, SpyKer may not correctly recognize or display event types correctly.*

Intercepting Kernel Functions

In addition to user events, SpyKer can be extended to automatically intercept new kernel functions to capture new events. However, only statically linked functions maybe intercepted. To add new interceptions and event captures, the file `stracepatches.c` on the target system needs to be edited and the SpyKer driver be rebuilt. Before editing this file, a backup copy should be created.

The following code needs to be added to `stracepatches.c` to intercept new functions and add new events.

```
static int function_blen = 0;
static int function_insb(arguments...)
{
    DUMMYINS
}

proxy_function (arguments...)
{
    DRTOC();
    trace(event_number, short_payload,
           long_payload, long_payload_size);
    OSTOC();
    function_insb(arguments...);
}
```

In the above code, *function* is the name of the function being intercepted and *arguments...* must exactly match the arguments of the intercepted function. See “User Event Capture” on page 53 for a description of the arguments to *__trace()*.

The following code needs to be added to the function *patch_routine()* in the file `stracepatches.c`.

```
if (trace_ev_ok(s, event_number) )
     PATCH(function, proxy_function,function_insb,
    function_blen );
```

Lastly, the following code needs to be added to the function *unpatch_routines()* in the file `stracepatches.c`.

```
UNPATCH(function, function_insb, function_blen );
```

APPENDIX A *Menu Reference*

SpyKer Menus

The following table shows all of the menus available on the SpyKer tool display, as well as references elsewhere in this document where more information can be found.

Table A-1: SpyKer Menu Reference

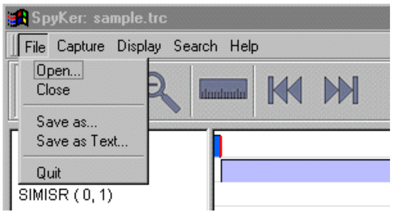
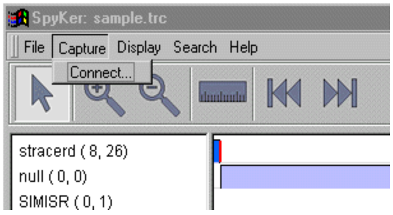
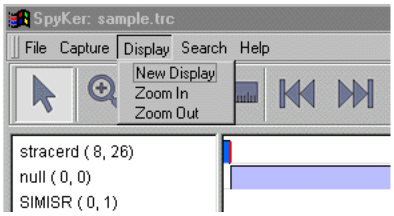
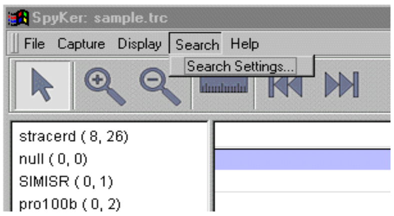
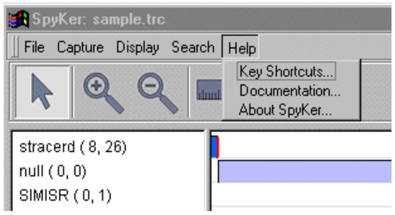
Menu	Option	References
File Menu  SpyKer: sample.trc File Capture Display Search Help Open... Close Save as... Save as Text... Quit SIMISR (0, 1)	**Open**	Page 17 Page 31
	Close	Page 31
	Save as	Page 32 Page 32
	Save as Text	Page 32
	Quit	Page 32
Capture Menu  SpyKer: sample.trc File Capture Display Search Help Connect... stracerd (8, 26) null (0, 0) SIMISR (0, 1)	**Connect**	Page 32 Page 36

Table A-1: SpyKer Menu Reference (Continued)

Menu	Option	References
Display Menu  SpyKer: sample.trc File Capture Display Search Help New Display Zoom In Zoom Out stracerd (8, 26) null (0, 0) SIMISR (0, 1)	**New Display**	Page 32
	Zoom In	Page 23 Page 33
	Zoom Out	Page 23 Page 33
Search Menu  SpyKer: sample.trc File Capture Display Search Help Search Settings... stracerd (8, 26) null (0, 0) SIMISR (0, 1) pro100b (0, 2)	**Search Settings**	Page 25 Page 33
Help Menu  SpyKer: sample.trc File Capture Display Search Help Key Shortcuts... Documentation... About SpyKer... stracerd (8, 26) null (0, 0) SIMISR (0, 1)	**Key Shortcuts**	Page 33
	Documentation	Page 34
	About Spyker...	Page 34

APPENDIX B

SpyKer Events and Payloads

Each SpyKer predefined event type is capable of delivering two forms of additional data called payloads. A short payload has a single 32-bit value. A long payload is event type-dependent and its size can be event-specific. That is, two events of the same event type may have different size long payloads.

SpyKer event filtering or searching may specify a value for the short payload only. The SpyKer display tool displays both short and long payloads, and assumes that user event long payloads are null terminated character strings.

Each event type has a unique event type number, also referred to as an event number. The user event types, defined as event types 37 through 45, are predefined event types available for application-specific events. The following table shows event numbers, event descriptions and payload information.

Table B-1: SpyKer Events and Payloads

Event Number	**Event Description**	**Payloads**	
		Short	**Long**
0	`Context Switch`	Superpid[1]	N/A
1	`System Call`	System call #	N/A
2	`Interrupt`	Interrupt #	N/A
3	`Return from interrupt`	Interrupt #	N/A
4	`Processor exception`	Exception #	N/A
5	`Thread stop`	N/A	N/A
6	`Program load`	Parent process' superpid	Name of program loaded

Table B-1: SpyKer Events and Payloads (Continued)

Event Number	Event Description	Payloads	
		Short	**Long**
7	`Thread wait`	N/A	N/A
8	`Thread wakeup`	Superpid of thread being awakened	N/A
9	`Process exit`	N/A	N/A
10	`User thread exit`	N/A	N/A
11	`System thread exit`	Thread id	N/A
12	`Return from system call`	System call return value	System call #
13	`Signal delivery (caught)`	Signal #	N/A
14	`Signal delivery (not caught)`	Signal #	N/A
15	`Memory allocation`	# of page[2] requested	Current # of free pages (before allocation)
16	`Memory free`	# of pages being freed	Current # of free pages (before free)
17	`Kernel malloc`	# of bytes requested	Return value
18	`Kernel free`	# of bytes freed	Address of memory
19	`New system thread`	New thread id	Thread name
20	`New user thread`	Superpid of new thread	N/A
21	`New process (fork)`	Superpid of new process	N/A
22	`Trace start`[3]	N/A	N/A
23	`Existing process`[4]	N/A	N/A
24	`Existing thread`[5]	N/A	N/A

Table B-1: SpyKer Events and Payloads (Continued)

Event Number	Event Description	Payloads	
		Short	Long
25	`Unknown event`	Unrecognized event type #	N/A
26	`Reserved 26`	N/A	N/A
27	`Reserved 27`	N/A	N/A
28	`Reserved 28`	N/A	N/A
29	`Reserved 29`	N/A	N/A
30	`Reserved 30`	N/A	N/A
31	`POSIX Message Send`	Message queue id	Message size (in bytes)
32	`POSIX Message receive`	Message queue id	Message size (in bytes)
33	`Mutex enter`	pthreads mutex id	N/A
34	`Mutex exit`	pthreads mutex id	N/A
35	`Condition wait`	pthreads condition variable id	N/A
36	`Condition signal`	pthreads condition variable id	N/A
37	`User 37`	Undefined	Undefined
38	`User 38`	Undefined	Undefined
39	`User 39`	Undefined	Undefined
40	`User 40`	Undefined	Undefined
41	`User 41`	Undefined	Undefined
42	`User 42`	Undefined	Undefined
43	`User 43`	Undefined	Undefined

Table B-1: SpyKer Events and Payloads (Continued)

Event Number	Event Description	Payloads	
		Short	Long
44	`User 44`	Undefined	Undefined
45	`User 45`	Undefined	Undefined

1. The LynxOS superpid is an encoding of the process id and the thread id.
2. Page size is normally 4 Kbytes
3. SpyKer private event.
4. SpyKer private event.
5. SpyKer private event.

APPENDIX C

Keyboard Shortcuts

The following table enumerates the keyboard shortcuts for the SpyKer display tool.

Table 0-1: Keyboard Shortcuts

Key	Action
Space bar	Switches tools in event panel
Control-O	**File** menu **Open** command
Control-W	**File** menu **Close** command
Control-S	**File** menu **Save As** command
Control-Q	**File** menu **Quit** command
Control-F	**Search** menu **Search Settings** command
Control-T	Starts **Trace Capture** dialog box
Control-K	**Help** menu **Key Shortcuts** command

Index