

Logrind User Guide

Christopher January

June 2004

1 What is Logrind 2?

Logrind 2 is a tool for capturing program traces. A program trace is a list of all the things a process does including writing and reading to memory, jumps, system calls, etc.

2 How to install

You can download the most recent version of Logrind from:
<http://www.atomice.com/snapshots>

To compile from sources you should make and install sqlite before compiling valgrind or gdb.

3 How to start

Launch gdb as usual by typing `gdb filename` at a command line. To activate logrind type `target logrind` at the GDB prompt. From now on GDB will trace any programs you run using Logrind. To run the program type `run` at the GDB prompt as usual.

When the process has exited or stopped due to a breakpoint you may examine the program trace using the `logrind history` command.

4 Logrind commands

4.1 `logrind history`

Syntax:

```
logrind history
logrind history COUNT
logrind history -START
logrind history START COUNT
logrind history START COUNT FILTER
logrind history FILTER
```

The `history` command displays a portion of the program trace. The region of the program trace that is displayed is controlled using the `START` and `COUNT` arguments. The `START` argument specifies the first line number to display. The `COUNT` argument specifies how many lines to display.

If the `START` argument is negative the last `START` lines of the program trace will be displayed.

The `FILTER` argument may be used to control which lines are displayed. For example to only show function calls you would use the filter `history /<call *>/`.

Example:

```
logrind history 1 10
```

```
1: [1] _init (???:0) regread ebp: bffff544
2: [1] _init (???:0) regread esp: bffff518
3: [1] _init (???:0) regwrite esp: bffff518 => bffff514
4: [1] _init (???:0) memwrite bffff514: bffff544 => bffff544
5: [1] _init + 1 (???:0) regwrite ebp: bffff544 => bffff514
6: [1] _init + 3 (???:0) regread esp: bffff514
7: [1] _init + 3 (???:0) regwrite esp: bffff514 => bffff50c
8: [1] _init + 6 (???:0) regwrite esp: bffff50c => bffff508
9: [1] _init + 6 (???:0) memwrite bffff508: 40013844 => 40232743
10: [1] _init + 6 (???:0) call call_gmon_start
```

4.1.1 Interpreting the program trace

A program trace line is interpreted as follows:

```
497: [1] main + 19 (hello.c:5)
      memwrite bffff528: 00000020 => 0000000e}
```

497	line number/cursor position
[1]	thread 1
main + 19	function name + offset
hello.c:5	source filename:line number
memwrite	opcode (one of memwrite, regwrite, memread, regread, jump, call, return, syscall, sysret)
bffff528	first argument to memwrite - the address written to
00000020	the pre-image (i.e. what was in memory at that address before it was overwritten)
0000000e	the post-image (i.e. what was written to memory)

4.2 logrind query

Syntax:

```
logrind query QUERY
```

The `logrind query` command can be used to execute arbitrary SQL queries on the program trace database. The results will be returned in a table.

4.3 logrind sources

Syntax:

```
logrind sources
```

`logrind sources` command displays a list of valid program trace sources. The active source may be set with the `set logrind source` command.

4.4 logrind macro define

Syntax:

```
logrind NAME REPLACEMENT
```

```
logrind NAME (ARG1, ..., ARGn) REPLACEMENT
```

Defines a macro for use in a query or filter.

4.5 logrind macro undefine

Syntax:

```
logrind NAME
```

Undefines a previously defined macro.

4.6 set logrind cursor

Syntax:

```
set logrind cursor POSITION
```

The `set logrind cursor` command is used to move the program trace cursor. All GDB memory, register and stack commands operate relative to this cursor. GDB will behave as if your program had stopped at that point in the process' execution because of a breakpoint.

If `POSITION` is positive the cursor will be set to that absolute line number of the program trace. If `POSITION` is negative the cursor will be set to `POSITION` lines from the end of the trace. If `POSITION` is 0 the program cursor will be switched off.

Example:

```
set logrind cursor 100
```

4.7 show logrind cursor

Syntax:

```
show logrind cursor
```

Shows the current position of the program trace cursor.

4.8 logrind cursor backwards

Syntax:

```
logrind cursor backwards
```

```
logrind cursor backwards COUNT
```

The `logrind cursor backwards` command moves the program trace cursor backwards `COUNT` rows. If the `COUNT` argument is omitted the cursor is moved back one row.

4.9 logrind cursor forward

Syntax:
`logrind cursor forward`
`logrind cursor forward COUNT`

The `logrind cursor forward` command moves the program trace cursor forward `COUNT` rows. If the `COUNT` argument is omitted the cursor is moved forward one row.

4.10 set logrind source

Syntax:
`set logrind source ID`

Each time you run your program using Logrind a new program trace is generated. The `set logrind source` command is used to change which program trace is currently active.

4.11 show logrind source

Syntax:
`show logrind source`

Show the id of the currently active source.

4.12 set logrind filter

Syntax:
`set logrind filter`
`set logrind filter FILTER`

The `set logrind filter` command sets a default filter to use with the `logrind history` command if no `FILTER` argument is supplied.

If the command is used without any arguments no default filter is used.

A filter is an SQL boolean expression.

4.13 show logrind filter

Syntax:
`show logrind filter`

Shows the default filter for the `logrind history` command.

4.14 set logrind suppressions

Syntax:

```
set logrind suppressions FILENAME
```

Tells the Logrind 2 skin to use the given suppressions file when launching applications. This command only works before the `target logrind` command.

4.15 show logrind suppressions

Syntax

```
show logrind suppressions
```

Shows the filename of the suppressions file currently in use.

4.16 set logrind max-database-size

Syntax:

```
set logrind max-database-size SIZE
```

Sets the maximum size of the program trace database to `SIZE` bytes. This command only works before the `target logrind` command.

4.17 show logrind max-database-size

Syntax:

```
show logrind max-database-size
```

Show the maximum size of the program trace database.

4.18 target logrind

Syntax:

```
target logrind
```

```
target logrind DATABASE
```

Use the `target logrind` to switch on program trace capture.

Logrind will automatically create a new program trace database if you do not supply one.

5 Query syntax

Queries in Logrind are extended SQL queries. A number of program trace specific functions are available for use both in queries and filters. A query is a complete SQL command, whereas a filter is an SQL boolean expression. Anything that can appear after `where` in an SQL `select` statement is a valid filter.

6 Functions

eval(sequence, 'expr')

Evaluates a source language expression at a particular point in the program trace.

addr2line(pc)

Returns the source filename and line numbers corresponding to the given program counter address.

line2addr('file:line')

Returns the start address of the given line.

symbol2beginaddr(sequence, 'symbol')

Returns the start address of the given symbol.

symbol2endaddr(sequence, 'symbol')

Returns the end address of the given symbol.

addr2symbol(sequence, address)

Returns the name of the symbol at the given address plus some offset.

inscope(sequence, 'symbol')

Returns true if the given symbol is in scope else false.

current_cursor()

Returns the current cursor position.

current_source()

Returns the id of the active source.

base64decode2int('data')

Converts the given base 64 data to an integer.

base64decode2uint('data')

Converts the given base 64 data to an unsigned integer.

hex2int('hex')

Converts the given hexadecimal number to an integer.

hex2uint('hex')

Converts the given hexadecimal number to an unsigned integer.

int2hex('hex')

Converts the given integer to a hexadecimal number.

uint2hex('hex')

Converts the given unsigned integer to a hexadecimal number.

7 Suppressions

You may set a suppressions file using the `set logrind suppressions` command. A suppressions file specifies what program trace events you want to be recorded. The contains one or more blocks in the following format:

```
{
  All libraries
  Logrind: All
  obj:/lib/*
}
```

The first line is a label or comment for the suppression and is ignored.

The second line specifies what events are to be suppressed. The permitted values are:

all, register read, register write, memory read, memory write, jump, call, return, system call, general register write, stack frame register write, general register read, stack frame register read, read, register, memory

Multiple values may be separated using commas. Prefixing any value with the allow keyword explicitly allows traces of that type to be recorded, overriding any previous suppressions.

See the `logrind.supp` file for more examples.