DrHook Manual

ECMWF

Mar 05, 2025

CONTENTS

CHAPTER

ONE

FLAGS & ENVIRONMENT VARIABLES

1.1 DR_HOOK_SHOW_LOCK

1.1.1 Valid Values

valid_value ::= '0' | '1'

1.1.2 Purpose

Used to specify if lock debug info should be output. This is on initialisation only.

1.1.3 Notes

This is done by enabling `OML_DEBUG` in the *OML* library, initialising a new `lockid`, and then restoring the original value.

This debug info takes the following format:
`oml_init_lockid_with_name "lock_name" : lock_ID, lock_address`

1.2 OMP_STACKSIZE

1.2.1 Valid Values

valid_value ::= <number> <suffix>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
suffix ::= '' | 'G' | 'M' | 'K'

1.2.2 Purpose

Specifies an indicative stack size the master thread should not exceed.

1.2.3 Notes

This is an overloaded environment variable, used primarily by *OpenMP*.

It will only be read if *DR_HOOK_TRACE_STACK* is also set.

`OMP_STACKSIZE` is scaled by `opt_trace_stack`, obtained from *DR_HOOK_TRACE_STACK*, to give `drhook_stacksize_threshold`. If `drhook_stacksize_threshold` is exceeded by the master thread during a `random_memstat` check, an abort will occur.

The stack size can be specified in GiB, MiB, or KiB using the suffix `G`, `M`, or `K` respectively. A lack of suffix will imply the stack size is in bytes. If multiple suffixes are specified, then the largest specified will be chosen. The stack size is truncated at the first occurring suffix.

The size is limited to the size of `long long int`.

An invalid or negative input size will result in a default of `0`.

1.3 DR_HOOK_CATCH_SIGNALS

1.3.1 Valid Values

valid_value ::= <signal> | <valid_value> <delim> <signal>
delim ::= ‘,’ | ‘ ‘ | ‘t’ | ‘/’
signal ::= ‘-1’ | <number>
number ::= <digit> | <number> <digit> | <number> ‘0’
digit ::= ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

1.3.2 Purpose

Specifies a list of signals to be caught and handled by drhook.

1.3.3 Notes

If $1 \leq$ `signal` $\leq$ `NSIG`, then it is registered to be caught and handled by drhook - unless it has been set to ignored by *DR_HOOK_IGNORE_SIGNALS*. `NSIG` is defined in `signal.h` and is system dependant.

If `signal` is set to `-1`, then all available catchable signals are registered to be caught and handled by drhook. Any value after `-1` will be discarded.

All other values will be silently discarded.

1.4 DR_HOOK_RESTORE_DEFAULT_SIGNALS

1.4.1 Valid Values

valid_value ::= <signal> | <valid_value> <delim> <signal>
delim ::= ‘,’ | ‘ ‘ | ‘t’ | ‘/’
signal ::= ‘-1’ | <number>
number ::= <digit> | <number> <digit> | <number> ‘0’
digit ::= ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

1.4.2 Purpose

Specifies a list of signals to have their default handler restored, if they haven’t been set to be ignored.

1.4.3 Notes

If $1 \leq$ `signal` $\leq$ `NSIG`, then it’s default signal handler is restored - unless it has been set to ignored by *DR_HOOK_IGNORE_SIGNALS*. `NSIG` is defined in `signal.h` and is system dependant.

If `signal` is set to `-1`, then all available catchable signals have their default signal handler restored. Any value after `-1` will be discarded.

All other values will be silently discarded.

1.5 DR_HOOK_IGNORE_SIGNALS

1.5.1 Valid Values

valid_value ::= <signal> | <valid_value> <delim> <signal>
delim ::= ‘,’ | ‘ ‘ | ‘t’ | ‘/’
signal ::= ‘-1’ | <number>
number ::= <digit> | <number> <digit> | <number> ‘0’
digit ::= ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

1.5.2 Purpose

Specifies a list of signals to be ignored by drhook. This means drhook will not handle these signals when they occur.

1.5.3 Notes

If $1 \leq$ `signal` $\leq$ `NSIG`, then it's set to inactive and ignored by drhook. `NSIG` is defined in `signal.h` and is system dependant.

If `signal` is set to `-1`, then all available catchable signals are set to inactive and be ignored by drhook. Any value after `-1` will be discarded.

All other values will be silently discarded.

1.6 DR_HOOK_SILENT

1.6.1 Valid Values

valid_value ::= '0' | '1'

1.6.2 Purpose

Used to enable or disable debug prints to `STDERR`.

1.6.3 Notes

This is not recommended during production run due to the excessive slowdown caused by printing.

1.7 DR_HOOK_INIT_SIGNALS

1.7.1 Valid Values

valid_value ::= '0' | '1'

1.7.2 Purpose

Specifies if signals should be initialised via drhook (dangerous, but sometimes necessary).

1.7.3 Notes

1.8 DR_HOOK_SHOW_PROCESS_OPTIONS

1.8.1 Valid Values

valid_value ::= <number> | '-1'
number ::= <digit> | <number> <digit> | <number> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.8.2 Purpose

Specifies processIDs for which all options will be output todo{Where?}.

1.8.3 Notes

If this option isn't specified, and *DR_HOOK_SILENT* doesn't evaluate to `True`, then this will default to the process with ID 1.

Specifying `-1` will enable this for all processes.

1.9 ATP_ENABLED

1.9.1 Valid Values

valid_value ::= '0' | '1'

1.9.2 Purpose

Enable Abnormal Termination Processing (ATP) mode for *Cray* systems. This will pass certain signals to ATP (Abnormal Termination Processing) instead of drhook.

1.9.3 Notes

If nothing is specified, then this will default to `0`.

This will also cause drhook to check the following flags:

- *ATP_MAX_CORES*
- *ATP_MAX_ANALYSIS_TIME*
- *ATP_IGNORE_SIGTERM*

The signals passed to ATP are:

- `SIGINT`
- `SIGFPE`
- `SIGILL`
- `SIGTRAP`
- `SIGABRT`
- `SIGBUS`
- `SIGSEGV`
- `SIGSYS`
- `SIGXCPU`
- `SIGXFSZ`
- `SIGTERM`

- `SIGTERM` is only passed to ATP if *ATP_IGNORE_SIGTERM* is set

1.10 ATP_MAX_CORES

1.10.1 Valid Values

valid_value ::= <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.10.2 Purpose

Used as a multiplier for calculating the spin wait time for all cores to dump if ATP is enabled and handling signals.

Also used along with *ATP_ENABLED* to determine if any cores are being dumped by ATP.

1.10.3 Notes

This is an overloaded environment variable, used primarily for ATP. If nothing is specified, then this will default to `20`.

This will only be enable if the *ATP_ENABLED* flag evaluates to `True`.

The size is limited to the size of `int`.

1.11 ATP_MAX_ANALYSIS_TIME

1.11.1 Valid Values

valid_value ::= <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.11.2 Purpose

Used as a base value for calculating the spin wait time for all cores to dump if ATP is enabled and handling signals.

1.11.3 Notes

This is an overloaded environment variable, used primarily for ATP. If nothing is specified, then this will default to `300`.

This will only be enable if the *ATP_ENABLED* flag evaluates to `True`.

The minimum between `ATP_MAX_ANALYSIS_TIME` and `drhook_harakiri_timeout` (set by *DR_HOOK_HARAKIRI_TIMEOUT*) is chosen as the base for calculating the time to spin wait for ATP to finish handling a signal, through the following:

```
int secs = MIN(drhook_harakiri_timeout, atp_max_analysis_time);
secs = 60 + MIN(tdiff * (atp_max_cores - 1), secs);
```

The size is limited to the size of **int**.

1.12 ATP_IGNORE_SIGTERM

1.12.1 Valid Values

valid_value ::= '0' | '1'

1.12.2 Purpose

Determines if `SIGTERM` should be handled by drhook or ATP.

1.12.3 Notes

This is an overloaded environment variable, used primarily for ATP. If nothing is specified, then this will default to `0`.

This will only be enable if the *ATP_ENABLED* flag evaluates to `True`.

1.13 DR_HOOK_ALLOW_COREDUMP

1.13.1 Valid Values

valid_value ::= <number> | '-1'
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.13.2 Purpose

Specifies both if core dumping should be enabled, and if so, which processes.

1.13.3 Notes

If `DR_HOOK_ALLOW_COREDUMP` evaluates to $1 \leq$ `process` $\leq$ `# of processes` then only the specified process has core dumping enabled.

If `DR_HOOK_ALLOW_COREDUMP` evaluates to `-1` then core dumping is enabled for all processes.

If `DR_HOOK_ALLOW_COREDUMP` evaluates to `0` then core dumping is disabled.

1.14 DR_HOOK_PROFILE

1.14.1 Valid Values

valid_value ::= <char> | <valid_value> <char>
char ::= <letter> | <digit> | <symbol>
letter ::= <lower> | <upper>
upper ::= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'
lower ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
symbol ::= ' ' | '!' | '"' | '#' | '$' | '%' | '&' | ''' | '(' | ')' | '*' | '+' | ',' | '-' | '.' | ':' | ';' | '<' | '=' | '>' | '?' | '@' | '[' | '\' | ']' | '^' | '_' | '`' | '{' | '|' | '}' | '~'

1.14.2 Purpose

Specifies where to output profiles.

1.14.3 Notes

While `DR_HOOK_PROFILE` is just to be a valid file name, many of the valid symbols are not recommended for obvious reasons.

If a relative path is used, then it will be relative to the rundir of the binary drhook is linked into.

If `DR_HOOK_PROFILE` doesn't contain the character `%`, then `.%d` will be appended.

If *DR_HOOK_PROFILE_PROC* is set and valid, but `DR_HOOK_PROFILE` is not set, then `DR_HOOK_PROFILE` defaults to `drhook.prof.%d`.

If drhook fails to set the process specific patch string either due to an error or process configuration, then it will default to `drhook.prof.0`.

`DR_HOOK_PROFILE` is also indirectly used in `get_memmon_out`. `-mem` is simply appended to the previously described path.

`DR_HOOK_PROFILE` is also indirectly used in `get_csv_out` for PAPI mode, enabled with *DR_HOOK_OPT*. `.csv` is simply appended to the previously described path.

This option is only used in profiling and memory profiling modes.

1.15 DR_HOOK_PROFILE_PROC

1.15.1 Valid Values

valid_value ::= <number> | '-1'
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.15.2 Purpose

Used to specify which process, or all processes, that should output profiling data.

1.15.3 Notes

If `DR_HOOK_PROFILE_PROC` is `-1`, then all processes will out profiling data. All other valid values will result in the one specified process outputting its data.

If this option isn't specified, then it will default to `-1`.

The size is limited to the size of **`double`**.

This option is only used in profiling and memory profiling modes.

1.16 DR_HOOK_PROFILE_LIMIT

1.16.1 Valid Values

valid_value ::= <number> | <number> '.' <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.16.2 Purpose

Specifies at which percentage of the maximum value data points should be dropped from profiling outputs.

1.16.3 Notes

If this option isn't specified, then it will default to `-10`.

The size is limited to the size of **`double`**.

This option is only used in profiling and memory profiling modes.

1.17 DR_HOOK_FUNCENTER

1.17.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.17.2 Purpose

Used to specify a thread to print memory and stack information to stdout upon entering a function.

1.17.3 Notes

If this option isn't specified, then it will default to `0`.

This will also set `opt_gethwm` and `opt_getstk` to `1`, that are usually handled by *DR_HOOK_OPT*.

To see when a process exits a function, use *DR_HOOK_FUNCEXIT*.

1.18 DR_HOOK_FUNCEXIT

1.18.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.18.2 Purpose

Used to specify a thread to print memory and stack information to stdout upon exiting a function.

1.18.3 Notes

If this option isn't specified, then it will default to `0`.

This will also set `opt_gethwm` and `opt_getstk` to `1`, that are usually handled by *DR_HOOK_OPT*.

To see when a process enters a function, use *DR_HOOK_FUNCENTER*.

1.19 DR_HOOK_TIMELINE

1.19.1 Valid Values

valid_value ::= <number> | '-1'
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.19.2 Purpose

Used to enable timeline mode for either all processes or a specific process.

1.19.3 Notes

If `DR_HOOK_TIMELINE` is `-1`, then all processes will have timeline mode enabled. All other non-zero valid values will result in the one specified process having timeline mode enabled.

If this option isn't specified, then it will default to `0`.

Setting `DR_HOOK_TIMELINE` to a non-zero value also causes drhook to check the following options:

- *DR_HOOK_TIMELINE_THREAD*
- *DR_HOOK_TIMELINE_FORMAT*
- *DR_HOOK_TIMELINE_UNITNO*
- *DR_HOOK_TIMELINE_FREQ*
- *DR_HOOK_TIMELINE_MB*

1.20 DR_HOOK_TIMELINE_THREAD

1.20.1 Valid Values

valid_value ::= <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.20.2 Purpose

Used to specify which threads should output timeline info upon entry and exit of a region.

1.20.3 Notes

If timeline mode is enabled via *DR_HOOK_TIMELINE*, then for all threads in the range $1 \rightarrow n$ (inclusive) drhook will print timeline information on both entry and exit from a region. drhook will also print the sum of *all threads* for the first thread.

If `DR_HOOK_TIMELINE_THREAD` is set to `0`, then all n will be set so as to cover all threads and the behaviour is identical to the above.

While not strictly valid, any value less than `0` will have the same behaviour as `0`.

If this option isn't specified, then it will default to `1`.

1.21 DR_HOOK_TIMELINE_FORMAT

1.21.1 Valid Values

valid_value ::= '0' | '1'

1.21.2 Purpose

Used to specify if timeline information should be output in value only or verbose mode.

1.21.3 Notes

This option is only available if timeline mode is enabled via *DR_HOOK_TIMELINE*.

If `DR_HOOK_TIMELINE_FORMAT` is `1`, then the following value only formatter will be used:

```
"%.6f %.4g %.4g %.4g %.4g"
```

Otherwise, the verbose formatter is used:

```
"wall=%.6f cpu=%.4g hwm=%.4g rss=%.4g curheap=%.4g stack=%.4g vmpeak=%.4g pag=%lld"
```

If this option isn't specified, then it will default to `1`.

1.22 DR_HOOK_TIMELINE_UNITNO

1.22.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.22.2 Purpose

Specifies the unit number to be used for timeline output.

1.22.3 Notes

This option is only available if timeline mode is enabled via *DR_HOOK_TIMELINE*.

`DR_HOOK_TIMELINE_UNITNO` must be an integer between `1` and `99`. Some unit numbers are reserved: `5` is standard input, `6` is standard output.

As the value of `DR_HOOK_TIMELINE_UNITNO` has to be interpreted by `atoi()`, only integer unit numbers are valid.

If `0` is specified, then timeline outputs are silently ignored.

If this option isn't specified, then it will default to `6`.

1.23 DR_HOOK_TIMELINE_FREQ

1.23.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit> | <valid_value> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.23.2 Purpose

Used to specify how many DR_HOOK_TIMELINE_FREQth regions entries/exits should be output during timeline mode.

1.23.3 Notes

This option is only available if timeline mode is enabled via *DR_HOOK_TIMELINE*.

If a drhook region is entered/exited and it is the nth entry/exit, where n is a multiple of `DR_HOOK_TIMELINE`, then the timeline information will be output.

If a drhook region is entered/exited and it isn't the nth call, but the resident set size varies by more than *DR_HOOK_TIMELINE_MB*, then it is output anyway.

A value less than 1 will silently disable timeline mode.

The size is limited to the size of `long long int`.

If this option isn't specified, then it will default to `1000000`.

1.24 DR_HOOK_TIMELINE_MB

1.24.1 Valid Values

valid_value ::= <number> | <number> '.' <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.24.2 Purpose

This specifies how much the resident set size needs to vary by, in MB, before *DR_HOOK_TIMELINE_FREQ* is overridden.

1.24.3 Notes

This option is only available if timeline mode is enabled via *DR_HOOK_TIMELINE*.

A value less than `0` will be set to `1.0`.

The size is limited to the size of `double`.

If this option isn't specified, then it will default to `1.0`.

1.25 DR_HOOK_TRACE_STACK

1.25.1 Valid Values

valid_value ::= <number> | <number> '.' <number>
number ::= <digit> | <number> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.25.2 Purpose

A multiplier for *OMP_STACKSIZE*, to monitor high master thread stack usage.

1.25.3 Notes

As described for *OMP_STACKSIZE*, `DR_HOOK_TRACE_STACK` is used to scale the value of *OMP_STACKSIZE* to give `drhook_stacksize_threshold`. If `drhook_stacksize_threshold` is exceeded by the master thread during a `random_memstat` check, an abort will occur.

`DR_HOOK_TRACE_STACK` being defined and non-zero also implies `opt_random_memstat` has a default value of `1` (meaning it will always trigger a `random_memstat` check on entry to a drhook region). However, if *DR_HOOK_RANDOM_MEMSTAT* is defined, it will override this value.

A value less than `0` will be set to `0`. Additionally, `drhook_stacksize_threshold` won't be scaled, and `opt_random_memstat` is not set to `1`.

The size is limited to the size of `double`.

If this option isn't specified, then it will default to `0`.

1.26 DR_HOOK_RANDOM_MEMSTAT

1.26.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.26.2 Purpose

Used to enable random memstat checks, and how often to do it.

1.26.3 Notes

The random memstat check is done when a drhook region is entered and `rand() % opt_random_memstat == 0`, or unconditionally when the feature is explicitly, or implicitly (see *DR_HOOK_TRACE_STACK*) enabled.

Due to the implementation of the random check, the value of `DR_HOOK_RANDOM_MEMSTAT` is important in ways that may not be immediately obvious. For example, a value of `2` will result in a check every other entry to a drhook region on average. Whereas a sufficiently large number will only result in a check every `1/RAND_MAX` times on average.

A value greater than `RAND_MAX` will be set to `RAND_MAX`.

A value less than `0` will be set to `0`, effectively disabling the feature.

The size is limited to the size of `int`.

If this option isn't specified, then it will default to `0`.

1.27 DR_HOOK_HASHBITS

1.27.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.27.2 Purpose

1.27.3 Notes

A value greater than `RAND_MAX` will be set to `RAND_MAX`.

A value less than `0` will be set to `0`.

The value of `DR_HOOK_HASHBITS`, `nhash`, is also used to update the values of `hashsize` and `hashmask`. This is done for `hashsize` via the following:

```
static unsigned int hashsize = ((unsigned int)1<<(nhash));
```

For `hashmask` it is:

```
static unsigned int hashmask = (((unsigned int)1<<(nhash))-1);
```

The size is limited to the size of `int`.

If this option isn't specified, then it will default to the definition of `NHASH`, typically `16`. As such, `hashsize` and `hashmask` default to `65536` and `65535`, respectively.

1.28 DR_HOOK_NCALLSTACK

1.28.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.28.2 Purpose

1.28.3 Notes

This is an overloaded value with 2 functions. The first is to set the precision used by drhook, which also determines which method is used to X. A performance penalty is incurred when using the single precision mode (valid non-zero values), and so is not recommended.

The second function of `DR_HOOK_NCALLSTACK`, is to act as a parameter when in single precision mode. This parameter sets the maximum stack depth allowed. You must be careful when selecting this that a sufficient depth is chosen. If the depth is exceeded, then drhook will abort.

A value less than `0` will be set to `0`, effectively selecting double precision.

The size is limited to the size of `int`.

If this option isn't specified, then it will default to `0`, which enables double precision mode.

1.29 DR_HOOK_HARAKIRI_TIMEOUT

1.29.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.29.2 Purpose

A timeout for when to kill threads that may have hung.

1.29.3 Notes

`DR_HOOK_HARAKIRI_TIMEOUT` is used in three places. The first is when a signal is first caught by drhook; it is used as the timeout for `alarm()`. This is to prevent hangs by triggering a second signal when the `alarm()` expires. This second signal then takes an alternative path for subsequent signals where the second use case of `DR_HOOK_HARAKIRI_TIMEOUT` occurs. Here it is used to spin for `DR_HOOK_HARAKIRI_TIMEOUT` +60 seconds before the thread is killed using `SIGKILL`. This is to allow for tracebacks to complete.

The third place it is used, is in conjunction with *ATP_MAX_ANALYSIS_TIME*. Here it is used in the following way to allow ATP to dump cores:

```
int secs = MIN(drhook_harakiri_timeout, atp_max_analysis_time);
secs = 60 + MIN(tdiff * (atp_max_cores - 1), secs);
```

The timeout is specified in seconds.

A value less than `0` will be set to the definition of `drhook_harakiri_timeout_default`, typically `500`.

The size is limited to the size of **`int`**.

If this option isn't specified, then it will default to the definition of `drhook_harakiri_timeout_default`.

1.30 DR_HOOK_USE_LOCKFILE

1.30.1 Valid Values

valid_value ::= '0' | '1'

1.30.2 Purpose

Toggles the use of a lockfile, `drhook_lock`, for outputting which thread handled a signal first.

1.30.3 Notes

When enabled, this will enable a lockfile when handling signals in drhook. This lockfile, `drhook_lock`, will contain the number of the thread which got the lock first.

`DR_HOOK_USE_LOCKFILE` having a value evaluating to `1`, will also disable some output regarding the use of *DR_HOOK_USE_LOCKFILE* in it's alternative path for subsequent signals.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `1`.

1.31 DR_HOOK_TRAPFPE

1.31.1 Valid Values

valid_value ::= '0' | '1'

1.31.2 Purpose

Toggles if floating point exceptions should be trapped, regardless of compiler settings.

1.31.3 Notes

If `DR_HOOK_TRAPFPE` evaluates to `1`, then drhook will trap floating point exceptions regardless of compilation settings. A value of `0` will instead rely on the compiler flags used, e.g. `-Ktrap=fp`.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `1`.

1.32 DR_HOOK_TRAPFPE_INVALID

1.32.1 Valid Values

valid_value ::= '0' | '1'

1.32.2 Purpose

Toggles whether invalid operation floating point exceptions should be trapped.

1.32.3 Notes

This will only be enabled if *DR_HOOK_TRAPFPE* evaluates to `1` or the compiler had trapping of floating point exceptions enabled.

These invalid operations are defined by IEEE 754 standard.

If the exception is not trapped, then the result of the operation is `NaN`.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `1`.

1.33 DR_HOOK_TRAPFPE_DIVBYZERO

1.33.1 Valid Values

valid_value ::= '0' | '1'

1.33.2 Purpose

Toggles whether divide by zero floating point exceptions should be trapped.

1.33.3 Notes

This will only be enabled if *DR_HOOK_TRAPFPE* evaluates to 1 or the compiler had trapping of floating point exceptions enabled.

This exception occurs when a finite nonzero number is divided by zero.

Any non-zero valid integer value will be set to 1.

If this option isn't specified, then it will default to 1.

1.34 DR_HOOK_TRAPFPE_OVERFLOW

1.34.1 Valid Values

valid_value ::= '0' | '1'

1.34.2 Purpose

Toggles whether overflow floating point exceptions should be trapped.

1.34.3 Notes

This will only be enabled if *DR_HOOK_TRAPFPE* evaluates to 1 or the compiler had trapping of floating point exceptions enabled.

This exception occurs when the result of a calculation cannot be represented as a finite value in the precision format of the destination. This behaviour is affected by rounding modes.

Any non-zero valid integer value will be set to 1.

If this option isn't specified, then it will default to 1.

1.35 DR_HOOK_TIMED_KILL

1.35.1 Valid Values

valid_value ::= <formatted> | <valid_value> <delim> <formatted>
delim ::= ',' | ' ' | 't' | '/'
formatted ::= <id> ':' <id> ':' <integer> ':' <double>
double ::= <double_part> | 'double_part' '.' <double_part>
double_part ::= <digit_0> | <double_part> <digit_0>
digit_0 ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
id = <integer> | '-1'
integer ::= <digit> | <integer> <digit> | <integer> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.35.2 Purpose

Set timers for specific, or all, threads to trigger signals. This is a developer option.

1.35.3 Notes

The parameters provided to `DR_HOOK_TIMED_KILL` (`target_process`, `target_oml_thread_id`, `target_signal`, and `start_time`) are in the following pattern:

`target_process:target_oml_thread_id:target_signal:start_time`

Note that:

- `target_process` can be a valid integer for a single process, or `-1` for all.
- `target_oml_thread_id` can be a valid integer for a single thread, or `-1` for all.
- `target_signal` must be valid for your system.
- `start_time` must be non-zero.

Any set of parameters that are invalid, will be silently discarded.

The timers will be started as part of drhook's initialisation, and are specified in seconds.

If this option isn't specified, then it will default to `NULL`.

1.36 DR_HOOK_DUMP_SMAPS

1.36.1 Valid Values

valid_value ::= '0' | '1'

1.36.2 Purpose

Specifies whether `/proc/<process_ID>/smaps` should be dumped when handling a signal or not. This is a developer option.

1.36.3 Notes

`smaps` will be dumped after the first signal is handled by drhook, but before signals are handled by other handlers, e.g. ATP. It will only be dumped for the process which raises the signal.

`smaps` will be dumped to `/proc/<process_ID>/smaps`.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.37 DR_HOOK_DUMP_MAPS

1.37.1 Valid Values

valid_value ::= '0' | '1'

1.37.2 Purpose

Specifies whether `/proc/<process_ID>/maps` should be dumped when handling a signal or not. This is a developer option.

1.37.3 Notes

`maps` will be dumped after the first signal is handled by drhook, but before signals are handled by other handlers, e.g. ATP. It will only be dumped for the process which raises the signal.

`maps` will be dumped to `/proc/<process_ID>/maps`

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.38 DR_HOOK_DUMP_BUDDYINFO

1.38.1 Valid Values

valid_value ::= '0' | '1'

1.38.2 Purpose

Specifies whether `/proc/buddyinfo` should be dumped when handling a signal or not. This is a developer option.

1.38.3 Notes

`buddyinfo` will be dumped after the first signal is handled by drhook, but before signals are handled by other handlers, e.g. ATP.

`buddyinfo` can also be dumped during the dumping of hugepages(enabled via *DR_HOOK_DUMP_HUGEPAGES*). This occurs during the handling of signals, and when drhook enters a region.

`buddyinfo` will be dumped to `/proc/buddyinfo`.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.39 DR_HOOK_DUMP_MEMINFO

1.39.1 Valid Values

valid_value ::= '0' | '1'

1.39.2 Purpose

Specifies whether `/proc/meminfo` should be dumped when handling a signal or not. This is a developer option.

1.39.3 Notes

`meminfo` will be dumped after the first signal is handled by drhook, but before signals are handled by other handlers, e.g. ATP.

`meminfo` can also be dumped during the dumping of hugepages (enabled via *DR_HOOK_DUMP_HUGEPAGES*). This occurs during the handling of signals, and when drhook enters a region.

`meminfo` will be dumped to `/proc/meminfo`.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.40 DR_HOOK_DUMP_HUGEPAGES

1.40.1 Valid Values

valid_value ::= <id> ',' <double>
double ::= <double_part> | 'double_part' '.' <double_part>
double_part ::= <digit_0> | <double_part> <digit_0>
digit_0 ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
id = <integer> | '-1'
integer ::= <digit> | <integer> <digit> | <integer> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.40.2 Purpose

1.40.3 Notes

The parameters provided to `DR_HOOK_DUMP_HUGEPAGES` (`target_process`, and `frequency`) are in the following pattern:

```
target_process, frequency
```

Note that:

- `target_process` can be a valid integer for a single process, or `-1` for all.
- `frequency` is given in seconds. This is the minimum time that must pass before huge pages are dumped - unless the function has been called with the `enforced` flag. Although, enforcement isn't currently used within drhook.

If either parameters are invalid, both will be silently discarded.

Hugepages are only dumped by the first thread.

Hugepage dumping can occur when handling a signal or entering a drhook region.

`/proc/buddyinfo` and `/proc/meminfo` can also be dumped with hugepages, if *DR_HOOK_DUMP_BUDDYINFO* and *DR_HOOK_DUMP_MEMINFO* are enabled respectively.

The dumping of hugepages is handled by ec_meminfo.

If this option isn't specified, then it will default to `0,0`.

1.41 DR_HOOK_GENCORE

1.41.1 Valid Values

valid_value ::= '0' | '1'

1.41.2 Purpose

Specifies whether generating core dumps is enabled or not.

1.41.3 Notes

Any non-zero valid integer value will be treated as `1`.

If this option isn't specified, then it will default to `0`.

1.42 DR_HOOK_GENCORE_SIGNAL

1.42.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit> | <valid_value> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.42.2 Purpose

Specifies the signal which triggers generating a core dump.

1.42.3 Notes

To be valid then it must be $1 \leq$ `signal` $\leq$ `NSIG`, and not equal to `SIGABRT`. `NSIG` and `SIGABRT` are defined in `signal.h` and are system dependant.

All invalid values will be silently discarded.

If this option isn't specified, then it will default to `0`, which causes it to be ignored.

1.43 DR_HOOK_STRICT_REGIONS

1.43.1 Valid Values

valid_value ::= '0' | '1'

1.43.2 Purpose

Specifies if drhook region entry and exit calls must use the same label.

1.43.3 Notes

drhook region entry and exit calls can take a string, which is treated as a label for the region. However, while drhook doesn't use the exit call label itself, extensions that hijack drhook calls may need the label as the unique identifier. This flag will cause drhook to crash instead of passing it to the extension.

Any non-zero valid integer value will be treated as `1`.

If this option isn't specified and *DR_HOOK_NVTX* is disabled, then it will default to `0`. If *DR_HOOK_NVTX* is enabled, then *DR_HOOK_STRICT_REGIONS* will be enabled regardless of the value given.

1.44 DR_HOOK_NVTX

1.44.1 Valid Values

valid_value ::= '0' | '1'

1.44.2 Purpose

Specifies if NVTX should be enabled at runtime or not.

1.44.3 Notes

Enables Nvidia's NVTX at runtime. This assumes that NVTX has been enabled and requested at compile time of drhook using ENABLEDR_HOOK_NVTX.

If *DR_HOOK_NVTX* is enabled, then *DR_HOOK_STRICT_REGIONS* will also be enabled. It will also enable `walltime` and `count` from *DR_HOOK_OPT*.

Setting `DR_HOOK_TIMELINE` to a non-zero value also causes drhook to check the following options:

- *DR_HOOK_NVTX_SPAM_CALL_COUNT*
- *DR_HOOK_NVTX_SPAM_WT*

Any non-zero valid integer value will be treated as `1`.

If this option isn't specified, then it will default to `0`.

1.45 DR_HOOK_NVTX_SPAM_CALL_COUNT

1.45.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit> | <valid_value> '0'
digit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.45.2 Purpose

Specifies the spam call count for NVTX.

1.45.3 Notes

Spam call count is the number of times a drhook region has to be called, without a cumulative runtime longer than the *DR_HOOK_NVTX_SPAM_WT* time, for drhook to skip subsequent calls of that region for the functionality of NVTX. It will not skip core drhook profiling.

This option is only available if NVTX is enabled via *DR_HOOK_NVTX*.

A value less than `0` will be set to the definition of `nvtx_SCC_default`, typically `10`. While it is possible to specify `0`, this will effectively disable NVTX as all regions will be skipped - unless they can accumulate sufficient runtime to satisfy *DR_HOOK_NVTX_SPAM_WT* in their first call.

The size is limited to the size of `int`.

If this option isn't specified, then it will default to the definition of `nvtx_SCC_default`.

1.46 DR_HOOK_NVTX_SPAM_WT

1.46.1 Valid Values

valid_value ::= <number> | 'number' '.' <number>
number ::= <digit> | <double_part> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.46.2 Purpose

Specifies the spam wall time for NVTX.

1.46.3 Notes

Spam wall time is the cumulative runtime a drhook region must have if it is not to be skipped for NVTX functionality after exceeding *DR_HOOK_NVTX_SPAM_CALL_COUNT*. It will not skip core drhook profiling.

This option is only available if NVTX is enabled via *DR_HOOK_NVTX*.

DR_HOOK_NVTX_SPAM_WT is measured in seconds.

A value less than `0` will be set to the definition of `nvtx_SWT_default`, typically `0.001`.

The size is limited to the size of `double`.

If this option isn't specified, then it will default to the definition of `nvtx_SWT_default`.

1.47 DR_HOOK_OPT

1.47.1 Valid Values

valid_value ::= <option> | <valid_value> <delim> <option>
delim ::= ',' | ' ' | 't' | '/'
option ::= <all> | <memory> | <heap> | <stack> | <rss> | <paging> | <walltime> | <cputime> | <count> | <memprof> | <cycles> | <cpuprof> | <trim> | <self> | <noself> | <noprop> | <nosize> | <cluster> | <callpath> | <papi>
all ::= 'ALL'
memory ::= 'MEM' | 'MEMORY'
time ::= 'TIME' | 'TIMES'
heap ::= 'HWM' | 'HEAP'
stack ::= 'STK' | 'STACK'
rss ::= 'RSS'
paging ::= 'PAG' | 'PAGING'
walltime ::= 'WALL' | 'WALLTIME'
cputime ::= 'CPU' | 'CPUTIME'
count ::= 'CALLS' | 'COUNT'
memprof ::= 'MEMPROF'
cycles ::= 'PROF' | 'WALLPROF' | 'CYCLES'
cpuprof ::= 'CPUPROF'
trim ::= 'TRIM'

self ::= ‘SELF’
noself ::= ‘NOSELF’
noprop ::= ‘NOPROP’ | ‘NOPROPAGATE’ | ‘NOPROPAGATE_SIGNALS’
nosize ::= ‘NOSIZE’ | ‘NOSIZEINFO’
cluster ::= ‘CLUSTER’ | ‘CLUSTERINFO’
callpath ::= ‘CALLPATH’
papi ::= ‘COUNTERS’

1.47.2 Purpose

Specifies a range of options relating to profiling parameters and outputs.

1.47.3 Notes

Table 1: DR_HOOK_OPT Options

Flag	Implements	Enables	Increments any_memstat?	None
all	None	• heap • stack • rss • paging • walltime • cputime • cycles • count • papi	Yes	None
memory	None	• heap • stack • rss • count	Yes	None
times	None	• walltime • cputime • count	No	None
heap	• Track and print current and max heap memory usage.	• count	Yes	• Can also be enabled by *DR_HOOK_FUNCENTER* or *DR_HOOK_FUNCEXIT* being enabled.

continues on next page

Table 1 – continued from previous page

Flag	Implements	Enables	Increments any_memstat?	None
stack	• Track and print current and max stack memory usage.	• count	Yes	• Can also be enabled by *DR_HOOK_FUNCENTER* or *DR_HOOK_FUNCEXIT* being enabled.
rss	• Track and print current and max resident set size.	• count	Yes	None
paging	• Track and print current number of allocated pages. • Also tracks the maximum number of pages allocated per drhook region.	• count	Yes	None
walltime	• Tracks total and per drhook region walltime.	• count	No	• `walltime` is measured in seconds.
cputime	• Tracks total and per drhook region cputime per process.	• count	No	• `cputime` is measured in CPU ticks. • Terminated child processes' time will also be included in this figure, i.e., multiprocessing will be reflected in the count.
count	• Tracks the total number of drhook regions entered (`calls`). • Also tracks how deep drhook is in nested regions (`status`).	None	No	• drhook uses `status` purely for tracking if it is currently in a region, and does not care about the depth.
memprof	• Unsure	• heap • stack • rss • paging • count	Yes	None

continues on next page

Table 1 – continued from previous page

Flag	Implements	Enables	Increments any_memstat?	None
cycles	• Tracks CPU cycles (`cycles`). • Also tracks walltime (`wallprof`) for each drhook region.	• wall-time • count	No	• Adds printing functions to `atexit()`. • Disables cpuprof.
cpuprof	• Unsure	• cputime • count	No	• Disables cycles. • Also adds printing functions to `atexit()`.
trim	• Trims drhook region names to remove leading spaces and characters after the first subsequent space. – e.g. " Hello World!" becomes "Hello". • Also converts drhook region names to upper case.	None	No	None
self	• Includes drhook in profiling, also prints it.	None	No	• Very expensive • The default is to include drhook in profiling, but doesn't print it.
noself	• Excludes drhook from profiling.	None	No	• The default is to include drhook in profiling, but doesn't print it.
noprop	• Does not propagate handled signals beyond drhook.	None	No	None
nosize	• Unsure	None	No	None
cluster	• Prints cluster ID and size.	None	No	None

continues on next page

Table 1 – continued from previous page

Flag	Implements	Enables	Increments any_memstat?	None
callpath	• Tracks and outputs the callpath and depth of drhook regions.	None	No	• Creates *much* more overhead. • Enables callpath mode and associated arguments. • By default, this is to a max depth of 50, but can be changed by *DR_HOOK_CALLPATH_DEPTH*.
papi	• Enables PAPI mode	• cycles • wall-time • calls • cycles	No	• Start and stops PAPI instrumentation with drhook regions • Counters being monitored by PAPI can be set with *DR_HOOK_PAPI_COUNTERS* • Disables cpuprof

1.48 DR_HOOK_CALLPATH_INDENT

1.48.1 Valid Values

valid_value ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8'

1.48.2 Purpose

Specifies the number of spaces to indent callpath output by.

1.48.3 Notes

This will only be enabled if *DR_HOOK_OPT* has `CALLPATH` set.

If `DR_HOOK_CALLPATH_INDENT` $<$ 1 or `DR_HOOK_CALLPATH_INDENT` $>$ 8, then it will be set to `callpath_indent_default`, which evaluates to `2`.

If this option isn't specified, then it will default to `callpath_indent_default`.

1.49 DR_HOOK_CALLPATH_DEPTH

1.49.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.49.2 Purpose

Set the maximum depth for tracking callpaths of nested drhook regions.

1.49.3 Notes

This will only be enabled if *DR_HOOK_OPT* has `CALLPATH` set.

Caution should be used when picking this value, as being too high will greatly increase both the processing and memory overhead of drhook.

The size is limited to the size of `int`.

If `DR_HOOK_CALLPATH_DEPTH` is less than 0, then it will be set to `callpath_depth_default`, which evaluates to `50`. A size of zero will only print the first drhook region in the callpath.

If this option isn't specified, then it will default to `callpath_depth_default`.

1.50 DR_HOOK_CALLPATH_PACKED

1.50.1 Valid Values

valid_value ::= '0' | '1'

1.50.2 Purpose

Outputs callpath information in a more compact format.

1.50.3 Notes

This will only be enabled if *DR_HOOK_OPT* has `CALLPATH` set.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.51 DR_HOOK_CALLTRACE

1.51.1 Valid Values

valid_value ::= '0' | '1'

1.51.2 Purpose

Specifies if calltrace mode should be enabled or not.

1.51.3 Notes

This will only be enabled if *DR_HOOK_OPT* has `CALLPATH` set.

Any non-zero valid integer value will be set to `1`.

If this option isn't specified, then it will default to `0`.

1.52 DR_HOOK_WATCH_PRINT_MAX

1.52.1 Valid Values

valid_value ::= <digit> | <valid_value> <digit>
digit ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

1.52.2 Purpose

Specifies the max number of elements per array to be printed per watchpoint.

1.52.3 Notes

If `DR_HOOK_WATCH_PRINT_MAX` ≥ 0, and less than the number of elements, then the number of elements up to the value of `DR_HOOK_WATCH_PRINT_MAX` will be printed.

The size is limited to the size of **`int`**.

If this option isn't specified, then it will default to `-1`.

1.53 DR_HOOK_PAPI_COUNTERS

1.53.1 Valid Values

valid_value ::= <counter> | <valid_value> <delim> <counter>
delim ::= ',' | ' ' | 't' | '/'
counter ::= <char> | <char> <char>
char ::= <letter> | <digit> | <symbol>

letter ::= <lower> | <upper>
upper ::= ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ | ‘Q’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘W’ | ‘X’ | ‘Y’ | ‘Z’
lower ::= ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ | ‘q’ | ‘r’ | ‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’
digit ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’
symbol ::= ‘_’

1.53.2 Purpose

Specifies the counters to be monitored when in PAPI mode.

1.53.3 Notes

PAPI counters are machine specific and can displayed with the **`papi avail`** command.

PAPI mode is enabled through *DR_HOOK_OPT*.

The output from PAPI mode is in csv format and the file path is described, and changed, by *DR_HOOK_PROFILE*.

A maximum of 4 counters can be specified. Any counters over this limit will be silently dropped.

If this option isn’t specified, then it will default to the following 4 counters:

```
API_TOT_CYC
PAPI_FP_OPS
PAPI_L1_DCA
PAPI_L2_DCM
```

CHAPTER

TWO

DEFINITIONS

2.1 _DRHOOK_C_

2.1.1 Value

1

2.1.2 Purpose

Never used within `drhook.c`

2.1.3 Preprocessor Guards

None

2.2 _DRHOOK_FILE_

2.2.1 Value

`drhook.c`

2.2.2 Purpose

Used within error handling to specify the source file where it occurred.

2.2.3 Preprocessor Guards

None

2.3 _GNU_SOURCE

2.3.1 Value

No value defined

2.3.2 Purpose

Never used within `drhook.c`.

2.3.3 Preprocessor Guards

None

2.4 HOST_NAME_MAX

2.4.1 Value

`_POSIX_HOST_NAME_MAX`

2.4.2 Purpose

Used to set the max size of char arrays for node HOSTNAME.

2.4.3 Preprocessor Guards

```
!defined(HOST_NAME_MAX) && defined(_POSIX_HOST_NAME_MAX)
```

2.5 HOST_NAME_MAX

2.5.1 Value

`_SC_HOST_NAME_MAX`

2.5.2 Purpose

Used to set the max size of char arrays for node HOSTNAME.

2.5.3 Preprocessor Guards

```
!defined(HOST_NAME_MAX) && defined(_SC_HOST_NAME_MAX)
```

2.6 EC_HOST_NAME_MAX

2.6.1 Value

```
HOST_NAME_MAX
```

2.6.2 Purpose

Used to set the max size of char arrays for node HOSTNAME.

2.6.3 Preprocessor Guards

```
defined(HOST_NAME_MAX)
```

2.7 EC_HOST_NAME_MAX

2.7.1 Value

```
512
```

2.7.2 Purpose

Used to set the max size of char arrays for node HOSTNAME.

2.7.3 Preprocessor Guards

```
!defined(HOST_NAME_MAX)
```

CHAPTER

THREE

GLOBAL VARIABLES

3.1 thread_cycles

3.1.1 Type

```
static long long int*
```

3.1.2 Value

```
NULL
```

3.1.3 Purpose

Stores the number of cycles for each thread.

3.2 drhook_lhook

3.2.1 Type

```
int
```

3.2.2 Value

```
1
```

3.2.3 Purpose

CHAPTER

FOUR

FUNCTIONS

4.1 backtrace()

4.1.1 Signature

```
static int backtrace(void **buffer, int size)
```

4.1.2 Purpose

Assumption: Used to provide a dummy function implementation, which is commonly defined by compilers other than the *NEC* compiler

4.1.3 Notes

Behind preprocessor guard `#ifdef__NEC__`