

SystemTap Tapset Reference Manual

SystemTap

SystemTap Tapset Reference Manual

by SystemTap

Copyright © 2008-2009 Red Hat, Inc. and others

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

Table of Contents

1. Introduction	1
Tapset Name Format	2
2. Context Functions	3
print_regs	35
execname	36
pid	37
tid	38
ppid	39
pexecname	40
gid	41
egid	42
uid	43
euid	44
cpu	45
pp	46
registers_valid	48
user_mode	49
is_return	50
target	51
stack_size	52
stack_used	53
stack_unused	54
uaddr	55
print_stack	57
probefunc	59
probemod	60
modname	61
symname	63
symdata	65
print_backtrace	67
backtrace	68
caller	69
caller_addr	70
3. Timestamp Functions	71
get_cycles	74
4. Memory Tapset	75
vm_fault_contains	87
vm.pagefault	89
vm.pagefault.return	91
addr_to_node	93
vm.write_shared	94
vm.write_shared_copy	96
vm.mmap	98
vm.munmap	100
vm.brk	102
vm.oom_kill	104
5. IO Scheduler Tapset	106
ioscheduler.elv_next_request	112
ioscheduler.elv_next_request.return	113
ioscheduler.elv_add_request	115
ioscheduler.elv_completed_request	117
6. SCSI Tapset	119
scsi.ioentry	125
scsi.iodispatching	127
scsi.iodone	130
scsi.iocompleted	133

7. Networking Tapset	136
netdev.receive	156
netdev.transmit	158
tcp.sendmsg	160
tcp.sendmsg.return	162
tcp.recvmsg	164
tcp.recvmsg.return	167
tcp.disconnect	170
tcp.disconnect.return	173
tcp.setsockopt	175
tcp.setsockopt.return	178
tcp.receive	180
udp.sendmsg	184
udp.sendmsg.return	186
udp.recvmsg	188
udp.recvmsg.return	190
udp.disconnect	192
udp.disconnect.return	194
ip_ntop	196
8. Socket Tapset	197
socket.send	223
socket.receive	226
socket.sendmsg	229
socket.sendmsg.return	232
socket.recvmsg	236
socket.recvmsg.return	239
socket.aio_write	243
socket.aio_write.return	246
socket.aio_read	250
socket.aio_read.return	253
socket.writev	257
socket.writev.return	260
socket.readv	264
socket.readv.return	267
socket.create	271
socket.create.return	274
socket.close	277
socket.close.return	280
sock_prot_num2str	282
sock_prot_str2num	283
sock_fam_num2str	284
sock_fam_str2num	285
sock_state_num2str	287
sock_state_str2num	288
9. Kernel Process Tapset	289
kprocess.create	297
kprocess.start	299
kprocess.exec	300
kprocess.exec_complete	302
kprocess.exit	304
kprocess.release	306
10. Signal Tapset	308
signal.send	335
signal.send.return	339
signal.checkperm	343
signal.checkperm.return	346
signal.wakeup	348
signal.check_ignored	350

signal.check_ignored.return	352
signal.force_segv	354
signal.force_segv.return	356
signal.syskill	358
signal.syskill.return	360
signal.sys_tkill	361
signal.systkill.return	363
signal.sys_tgkill	364
signal.sys_tgkill.return	366
signal.send_sig_queue	367
signal.send_sig_queue.return	369
signal.pending	370
signal.pending.return	372
signal.handle	373
signal.handle.return	376
signal.do_action	377
signal.do_action.return	379
signal.procmask	380
signal.flush	382

<copyright>
<year>2008-2009</year>
<holder>Red Hat, Inc. and others</holder>
</copyright>
<year>2008-2009</year>
<holder>Red Hat, Inc. and others</holder>
<authorgroup>SystemTap</authorgroup>
SystemTapSystemTap
<contrib>Hackers</contrib>
<legalnotice>

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

</legalnotice>

This documentation is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

Chapter 1. Introduction

SystemTap provides free software (GPL) infrastructure to simplify the gathering of information about the running Linux system. This assists diagnosis of a performance or functional problem. SystemTap eliminates the need for the developer to go through the tedious and disruptive instrument, recompile, install, and reboot sequence that may be otherwise required to collect data.

SystemTap provides a simple command line interface and scripting language for writing instrumentation for a live running kernel. The instrumentation makes extensive use of the probe points and functions provided in the *tapset* library. This document describes the various probe points and functions.

Tapset Name Format

In this guide, tapset definitions appear in the following format:

```
name: return (parameters)
definition
```

The *return* field specifies what data type the tapset extracts and returns from the kernel during a probe (and thus, returns). Tapsets use 2 data types for *return*: *long* (tapset extracts and returns an integer) and *string* (tapset extracts and returns a string).

In some cases, tapsets do not have a *return* value. This simply means that the tapset does not extract anything from the kernel. This is common among asynchronous events such as timers, exit functions, and print functions.

SystemTap provides free software (GPL) infrastructure to simplify the gathering of information about the running Linux system. This assists diagnosis of a performance or functional problem. SystemTap eliminates the need for the developer to go through the tedious and disruptive instrument, recompile, install, and reboot sequence that may be otherwise required to collect data.

SystemTap provides a simple command line interface and scripting language for writing instrumentation for a live running kernel. The instrumentation makes extensive use of the probe points and functions provided in the *tapset* library. This document describes the various probe points and functions.

tapset

Tapset Name Format

In this guide, tapset definitions appear in the following format:

```
name: return (parameters)
definition
```

The *return* field specifies what data type the tapset extracts and returns from the kernel during a probe (and thus, returns). Tapsets use 2 data types for *return*: *long* (tapset extracts and returns an integer) and *string* (tapset extracts and returns a string).

In some cases, tapsets do not have a *return* value. This simply means that the tapset does not extract anything from the kernel. This is common among asynchronous events such as timers, exit functions, and print functions.

In this guide, tapset definitions appear in the following format:

```
name: return (parameters)
definition
```

The *return* field specifies what data type the tapset extracts and returns from the kernel during a probe (and thus, returns). Tapsets use 2 data types for *return*: *long* (tapset extracts and returns an integer) and *string* (tapset extracts and returns a string).

returnreturnlongstring

In some cases, tapsets do not have a *return* value. This simply means that the tapset does not extract anything from the kernel. This is common among asynchronous events such as timers, exit functions, and print functions.

return

Chapter 2. Context Functions

The context functions provide additional information about where an event occurred. These functions can provide information such as a backtrace to where the event occurred and the current register values for the processor.

Name

`print_regs` -- Print a register dump.

Synopsis

```
print_regs()
```

Arguments

None

Name

execname -- Returns the execname of a target process (or group of processes).

Synopsis

```
execname:string()
```

Arguments

None

Name

`pid --` Returns the ID of a target process.

Synopsis

```
pid:long()
```

Arguments

None

Name

`tid` -- Returns the thread ID of a target process.

Synopsis

```
tid:long()
```

Arguments

None

Name

ppid -- Returns the process ID of a target process's parent process.

Synopsis

```
ppid:long()
```

Arguments

None

Name

`pexecname` -- Returns the `execname` of a target process's parent process.

Synopsis

```
pexecname:string()
```

Arguments

None

Name

`gid --` Returns the group ID of a target process.

Synopsis

```
gid:long()
```

Arguments

None

Name

`egid --` Returns the effective gid of a target process.

Synopsis

```
egid:long()
```

Arguments

None

Name

`uid --` Returns the user ID of a target process.

Synopsis

```
uid:long()
```

Arguments

None

Name

`eid` -- Return the effective uid of a target process.

Synopsis

```
eid:long()
```

Arguments

None

Name

cpu -- Returns the current cpu number.

Synopsis

```
cpu:long()
```

Arguments

None

Name

pp -- Return the probe point associated with the currently running probe handler,

Synopsis

```
pp:string()
```

Arguments

None

Description

including alias and wildcard expansion effects

Context

The current probe point.

Name

`registers_valid` -- Determines validity of `<command>register</command>` and `<command>u_register</command>` in current context.

Synopsis

```
registers_valid:long()
```

Arguments

None

Description

Return 1 if `register` and `u_register` can be used in the current context, or 0 otherwise. For example, `<command>registers_valid</command>` returns 0 when called from a begin or end probe.

Name

`user_mode` -- Determines if probe point occurs in user-mode.

Synopsis

```
user_mode:long()
```

Arguments

None

Description

Return 1 if the probe point occurred in user-mode.

Name

`is_return` -- Determines if probe point is a return probe.

Synopsis

```
is_return:long()
```

Arguments

None

Description

Return 1 if the probe point is a return probe. *Deprecated.*

Name

target -- Return the process ID of the target process.

Synopsis

```
target:long()
```

Arguments

None

Name

`stack_size` -- Return the size of the kernel stack.

Synopsis

```
stack_size:long()
```

Arguments

None

Name

`stack_used` -- Returns the amount of kernel stack used.

Synopsis

```
stack_used:long()
```

Arguments

None

Description

Determines how many bytes are currently used in the kernel stack.

Name

`stack_unused` -- Returns the amount of kernel stack currently available.

Synopsis

```
stack_unused:long()
```

Arguments

None

Description

Determines how many bytes are currently available in the kernel stack.

Name

`uaddr` -- User space address of current running task. EXPERIMENTAL.

Synopsis

```
uaddr:long()
```

Arguments

None

Description

Returns the address in userspace that the current task was at when the probe occurred. When the current running task isn't a user space thread, or the address cannot be found, zero is returned. Can be used to see where the current task is combined with `usymname` or `symdata`. Often the task will be in the VDSO where it entered the kernel. FIXME - need VDSO tracking support #10080.

Name

`print_stack` -- Print out stack from string.

Synopsis

```
print_stack(stk:string)
```

Arguments

stk *stk*
String with list of hexadecimal addresses.

Description

Perform a symbolic lookup of the addresses in the given `string`, which is assumed to be the result of a prior call to `<command>backtrace</command>`.

Print one line per address, including the address, the name of the function containing the address, and an estimate of its position within that function. Return nothing.

Name

probefunc -- Return the probe point's function name, if known.

Synopsis

```
probefunc:string()
```

Arguments

None

Name

probemod -- Return the probe point's module name, if known.

Synopsis

```
probemod:string()
```

Arguments

None

Name

`modname` -- Return the kernel module name loaded at the address.

Synopsis

```
modname:string(addr:long)
```

Arguments

addr *addr*
The address.

Description

Returns the module name associated with the given address if known. If not known it will return the string “<unknown>”. If the address was not in a kernel module, but in the kernel itself, then the string “kernel” will be returned.

Name

`symname` -- Return the symbol associated with the given address.

Synopsis

```
symname:string(addr:long)
```

Arguments

addr *addr*

The address to translate.

Description

Returns the (function) symbol name associated with the given address if known. If not known it will return the hex string representation of `addr`.

Name

`symdata` -- Return the symbol and module offset for the address.

Synopsis

```
symdata:string(addr:long)
```

Arguments

addr *addr*

The address to translate.

Description

Returns the (function) symbol name associated with the given address if known, plus the module name (between brackets) and the offset inside the module, plus the size of the symbol function. If any element is not known it will be omitted and if the symbol name is unknown it will return the hex string for the given address.

Name

`print_backtrace -- Print stack back trace`

Synopsis

```
print_backtrace()
```

Arguments

None

Description

Equivalent to `<command>print_stack(backtrace)</command>`, except that deeper stack nesting may be supported. Return nothing.

Name

backtrace -- Hex backtrace of current stack

Synopsis

```
backtrace:string()
```

Arguments

None

Description

Return a string of hex addresses that are a backtrace of the stack. Output may be truncated as per maximum string length.

Name

caller -- Return name and address of calling function

Synopsis

```
caller:string()
```

Arguments

None

Description

Return the address and name of the calling function. <emphasis>Works only for return probes at this time.</emphasis>

Name

caller_addr -- Return caller address

Synopsis

```
caller_addr:long()
```

Arguments

None

Description

Return the address of the calling function. **<emphasis> Works only for return probes at this time.</emphasis>**

The context functions provide additional information about where an event occurred. These functions can provide information such as a backtrace to where the event occurred and the current register values for the processor.

Name

`print_regs` -- Print a register dump.

Synopsis

```
print_regs()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

`print_regs``print_regs(3stap)`

Name

`print_regs` -- Print a register dump.

`print_regs` -- Print a register dump.

Synopsis

```
print_regs()
```

```
print_regs()
```

Arguments

None

None

Name

execname -- Returns the execname of a target process (or group of processes).

Synopsis

```
execname:string()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

execnameexecname(3stap)

Name

execname -- Returns the execname of a target process (or group of processes).

execname -- Returns the execname of a target process (or group of processes).

Synopsis

```
execname:string()
```

```
execname:string()
```

Arguments

None

None

Name

pid -- Returns the ID of a target process.

Synopsis

```
pid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

pidpid(3stap)

Name

pid -- Returns the ID of a target process.

pid -- Returns the ID of a target process.

Synopsis

```
pid:long()
```

```
pid:long()
```

Arguments

None

None

Name

tid -- Returns the thread ID of a target process.

Synopsis

```
tid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

tidtid(3stap)

Name

tid -- Returns the thread ID of a target process.

tid -- Returns the thread ID of a target process.

Synopsis

```
tid:long()
```

```
tid:long()
```

Arguments

None

None

Name

ppid -- Returns the process ID of a target process's parent process.

Synopsis

```
ppid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

ppidppid(3stap)

Name

ppid -- Returns the process ID of a target process's parent process.

ppid -- Returns the process ID of a target process's parent process.

Synopsis

```
ppid:long()
```

```
ppid:long()
```

Arguments

None

None

Name

pexecname -- Returns the execname of a target process's parent process.

Synopsis

```
pexecname:string()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

pexecnamepexecname(3stap)

Name

pexecname -- Returns the execname of a target process's parent process.

pexecname -- Returns the execname of a target process's parent process.

Synopsis

```
pexecname:string()
```

```
pexecname:string()
```

Arguments

None

None

Name

gid -- Returns the group ID of a target process.

Synopsis

```
gid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

gidgid(3stap)

Name

gid -- Returns the group ID of a target process.

gid -- Returns the group ID of a target process.

Synopsis

```
gid:long()
```

```
gid:long()
```

Arguments

None

None

Name

egid -- Returns the effective gid of a target process.

Synopsis

```
egid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

egidegid(3stap)

Name

egid -- Returns the effective gid of a target process.

egid -- Returns the effective gid of a target process.

Synopsis

```
egid:long()
```

```
egid:long()
```

Arguments

None

None

Name

uid -- Returns the user ID of a target process.

Synopsis

```
uid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

uiduid(3stap)

Name

uid -- Returns the user ID of a target process.

uid -- Returns the user ID of a target process.

Synopsis

```
uid:long()
```

```
uid:long()
```

Arguments

None

None

Name

euclid -- Return the effective uid of a target process.

Synopsis

```
euclid:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

euclid(3stap)

Name

euclid -- Return the effective uid of a target process.

euclid -- Return the effective uid of a target process.

Synopsis

```
euclid:long()
```

```
euclid:long()
```

Arguments

None

None

Name

cpu -- Returns the current cpu number.

Synopsis

```
cpu:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

cpucpu(3stap)

Name

cpu -- Returns the current cpu number.

cpu -- Returns the current cpu number.

Synopsis

```
cpu:long()
```

```
cpu:long()
```

Arguments

None

None

Name

pp -- Return the probe point associated with the currently running probe handler,

Synopsis

```
pp:string()
```

Arguments

None

Description

including alias and wildcard expansion effects

Context

The current probe point.

SystemTap Tapset Reference™

<date>February 2010</date>

pppp(3stap)

Name

pp -- Return the probe point associated with the currently running probe handler,

pp -- Return the probe point associated with the currently running probe handler,

Synopsis

```
pp:string()
```

```
pp:string()
```

Arguments

None

None

Description

including alias and wildcard expansion effects

including alias and wildcard expansion effects

Context

The current probe point.

The current probe point.

Name

`registers_valid` -- Determines validity of `<command>register</command>` and `<command>u_register</command>` in current context.

Synopsis

```
registers_valid:long()
```

Arguments

None

Description

Return 1 if `register` and `u_register` can be used in the current context, or 0 otherwise. For example, `<command>registers_valid</command>` returns 0 when called from a begin or end probe.

SystemTap Tapset Reference™

<date>February 2010</date>

`registers_valid``registers_valid(3stap)`

Name

`registers_valid` -- Determines validity of `<command>register</command>` and `<command>u_register</command>` in current context.

`registers_valid` -- Determines validity of `<command>register</command>` and `<command>u_register</command>` in current context. `register``u_register`

Synopsis

```
registers_valid:long()
```

```
registers_valid:long()
```

Arguments

None

None

Description

Return 1 if `register` and `u_register` can be used in the current context, or 0 otherwise. For example, `<command>registers_valid</command>` returns 0 when called from a begin or end probe.

Return 1 if `register` and `u_register` can be used in the current context, or 0 otherwise. For example, `<command>registers_valid</command>` returns 0 when called from a begin or end probe.

`register``u_register``registers_valid`

Name

`user_mode` -- Determines if probe point occurs in user-mode.

Synopsis

```
user_mode:long()
```

Arguments

None

Description

Return 1 if the probe point occurred in user-mode.

SystemTap Tapset Reference™

<date>February 2010</date>

`user_mode`
`user_mode(3stap)`

Name

`user_mode` -- Determines if probe point occurs in user-mode.

`user_mode` -- Determines if probe point occurs in user-mode.

Synopsis

```
user_mode:long()
```

```
user_mode:long()
```

Arguments

None

None

Description

Return 1 if the probe point occurred in user-mode.

Return 1 if the probe point occurred in user-mode.

Name

is_return -- Determines if probe point is a return probe.

Synopsis

```
is_return:long()
```

Arguments

None

Description

Return 1 if the probe point is a return probe. **<emphasis>Deprecated.</emphasis>**
SystemTap Tapset Reference™

<date>February 2010</date>

is_returnis_return(3stap)

Name

is_return -- Determines if probe point is a return probe.

is_return -- Determines if probe point is a return probe.

Synopsis

```
is_return:long()
```

```
is_return:long()
```

Arguments

None

None

Description

Return 1 if the probe point is a return probe. **<emphasis>Deprecated.</emphasis>**

Return 1 if the probe point is a return probe. **<emphasis>Deprecated.</emphasis>**

Name

target -- Return the process ID of the target process.

Synopsis

```
target:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

targettarget(3stap)

Name

target -- Return the process ID of the target process.

target -- Return the process ID of the target process.

Synopsis

```
target:long()
```

```
target:long()
```

Arguments

None

None

Name

stack_size -- Return the size of the kernel stack.

Synopsis

```
stack_size:long()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

stack_sizestack_size(3stap)

Name

stack_size -- Return the size of the kernel stack.

stack_size -- Return the size of the kernel stack.

Synopsis

```
stack_size:long()
```

```
stack_size:long()
```

Arguments

None

None

Name

stack_used -- Returns the amount of kernel stack used.

Synopsis

```
stack_used:long()
```

Arguments

None

Description

Determines how many bytes are currently used in the kernel stack.

SystemTap Tapset Reference™

<date>February 2010</date>

stack_usedstack_used(3stap)

Name

stack_used -- Returns the amount of kernel stack used.

stack_used -- Returns the amount of kernel stack used.

Synopsis

```
stack_used:long()
```

```
stack_used:long()
```

Arguments

None

None

Description

Determines how many bytes are currently used in the kernel stack.

Determines how many bytes are currently used in the kernel stack.

Name

stack_unused -- Returns the amount of kernel stack currently available.

Synopsis

```
stack_unused:long()
```

Arguments

None

Description

Determines how many bytes are currently available in the kernel stack.

SystemTap Tapset Reference™

<date>February 2010</date>

stack_unusedstack_unused(3stap)

Name

stack_unused -- Returns the amount of kernel stack currently available.

stack_unused -- Returns the amount of kernel stack currently available.

Synopsis

```
stack_unused:long()
```

```
stack_unused:long()
```

Arguments

None

None

Description

Determines how many bytes are currently available in the kernel stack.

Determines how many bytes are currently available in the kernel stack.

Name

uaddr -- User space address of current running task. EXPERIMENTAL.

Synopsis

```
uaddr:long()
```

Arguments

None

Description

Returns the address in userspace that the current task was at when the probe occurred. When the current running task isn't a user space thread, or the address cannot be found, zero is returned. Can be used to see where the current task is combined with `usymname` or `syndata`. Often the task will be in the VDSO where it entered the kernel. FIXME - need VDSO tracking support #10080.

SystemTap Tapset Reference™

<date>February 2010</date>

uaddr:long()

Name

uaddr -- User space address of current running task. EXPERIMENTAL.

uaddr -- User space address of current running task. EXPERIMENTAL.

Synopsis

```
uaddr:long()
```

```
uaddr:long()
```

Arguments

None

None

Description

Returns the address in userspace that the current task was at when the probe occurred. When the current running task isn't a user space thread, or the address cannot be found, zero is returned. Can be used to see where the current task is combined with `usymname` or `syndata`. Often the task will be in the VDSO where it entered the kernel. FIXME - need VDSO tracking support #10080.

Returns the address in userspace that the current task was at when the probe occurred. When the current running task isn't a user space thread, or the address cannot be found, zero is returned. Can be used to see where the

current task is combined with `usymname` or `symdata`. Often the task will be in the VDSO where it entered the kernel. FIXME - need VDSO tracking support #10080.
`usymnamesymdata`

Name

`print_stack` -- Print out stack from string.

Synopsis

```
print_stack(stk:string)
```

Arguments

stk *stk*
String with list of hexadecimal addresses.

Description

Perform a symbolic lookup of the addresses in the given `string`, which is assumed to be the result of a prior call to `<command>backtrace</command>`.

Print one line per address, including the address, the name of the function containing the address, and an estimate of its position within that function. Return nothing.

SystemTap Tapset Reference™

<date>February 2010</date>

`print_stack``print_stack(3stap)`

Name

`print_stack` -- Print out stack from string.

`print_stack` -- Print out stack from string.

Synopsis

```
print_stack(stk:string)
```

```
print_stack(stk:string)
```

Arguments

stk *stk*
String with list of hexadecimal addresses.

stk *stk*
String with list of hexadecimal addresses.

`<varlistentry>stk`

String with list of hexadecimal addresses.

`</varlistentry>`

`stkstk`

String with list of hexadecimal addresses.

String with list of hexadecimal addresses.

Description

Perform a symbolic lookup of the addresses in the given `string`, which is assumed to be the result of a prior call to `<command>backtrace</command>`.

Print one line per address, including the address, the name of the function containing the address, and an estimate of its position within that function. Return nothing.

Perform a symbolic lookup of the addresses in the given `string`, which is assumed to be the result of a prior call to `<command>backtrace</command>`.

`backtrace`

Print one line per address, including the address, the name of the function containing the address, and an estimate of its position within that function. Return nothing.

Name

probefunc -- Return the probe point's function name, if known.

Synopsis

```
probefunc:string()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

probefuncprobefunc(3stap)

Name

probefunc -- Return the probe point's function name, if known.

probefunc -- Return the probe point's function name, if known.

Synopsis

```
probefunc:string()
```

```
probefunc:string()
```

Arguments

None

None

Name

probemod -- Return the probe point's module name, if known.

Synopsis

```
probemod:string()
```

Arguments

None

SystemTap Tapset Reference™

<date>February 2010</date>

probemodprobemod(3stap)

Name

probemod -- Return the probe point's module name, if known.

probemod -- Return the probe point's module name, if known.

Synopsis

```
probemod:string()
```

```
probemod:string()
```

Arguments

None

None

Name

modname -- Return the kernel module name loaded at the address.

Synopsis

```
modname:string(addr:long)
```

Arguments

addr *addr*
The address.

Description

Returns the module name associated with the given address if known. If not known it will return the string “<unknown>”. If the address was not in a kernel module, but in the kernel itself, then the string “kernel” will be returned.

SystemTap Tapset Reference™

<date>February 2010</date>

modnamemodname(3stap)

Name

modname -- Return the kernel module name loaded at the address.

modname -- Return the kernel module name loaded at the address.

Synopsis

```
modname:string(addr:long)
```

```
modname:string(addr:long)
```

Arguments

addr *addr*
The address.

addr *addr*
The address.

<varlistentry>*addr*

The address.

</varlistentry>

addraddr

The address.

The address.

Description

Returns the module name associated with the given address if known. If not known it will return the string “<unknown>”. If the address was not in a kernel module, but in the kernel itself, then the string “kernel” will be returned.

Returns the module name associated with the given address if known. If not known it will return the string “<unknown>”. If the address was not in a kernel module, but in the kernel itself, then the string “kernel” will be returned.

“<unknown>” “kernel”

Name

`symname` -- Return the symbol associated with the given address.

Synopsis

```
symname:string(addr:long)
```

Arguments

addr *addr*
The address to translate.

Description

Returns the (function) symbol name associated with the given address if known. If not known it will return the hex string representation of `addr`.

SystemTap Tapset Reference™

<date>February 2010</date>

`symnamesymname(3stap)`

Name

`symname` -- Return the symbol associated with the given address.

`symname` -- Return the symbol associated with the given address.

Synopsis

```
symname:string(addr:long)
```

```
symname:string(addr:long)
```

Arguments

addr *addr*
The address to translate.

addr *addr*
The address to translate.

```
<varlistentry>addr  
The address to translate.  
</varlistentry>  
addraddr  
The address to translate.  
The address to translate.
```

Description

Returns the (function) symbol name associated with the given address if known. If not known it will return the hex string representation of *addr*.

Returns the (function) symbol name associated with the given address if known. If not known it will return the hex string representation of *addr*.

Name

`symdata --` Return the symbol and module offset for the address.

Synopsis

```
symdata:string(addr:long)
```

Arguments

addr *addr*
The address to translate.

Description

Returns the (function) symbol name associated with the given address if known, plus the module name (between brackets) and the offset inside the module, plus the size of the symbol function. If any element is not known it will be omitted and if the symbol name is unknown it will return the hex string for the given address.

SystemTap Tapset Reference™

<date>February 2010</date>

`symdata``symdata(3stap)`

Name

`symdata --` Return the symbol and module offset for the address.

`symdata --` Return the symbol and module offset for the address.

Synopsis

```
symdata:string(addr:long)
```

```
symdata:string(addr:long)
```

Arguments

addr *addr*
The address to translate.

addr *addr*
The address to translate.

```
<varlistentry>addr  
The address to translate.  
</varlistentry>  
addraddr  
The address to translate.  
The address to translate.
```

Description

Returns the (function) symbol name associated with the given address if known, plus the module name (between brackets) and the offset inside the module, plus the size of the symbol function. If any element is not known it will be ommitted and if the symbol name is unknown it will return the hex string for the given address.

Returns the (function) symbol name associated with the given address if known, plus the module name (between brackets) and the offset inside the module, plus the size of the symbol function. If any element is not known it will be ommitted and if the symbol name is unknown it will return the hex string for the given address.

Name

`print_backtrace` -- Print stack back trace

Synopsis

```
print_backtrace()
```

Arguments

None

Description

Equivalent to `<command>print_stack(backtrace)</command>`, except that deeper stack nesting may be supported. Return nothing.

SystemTap Tapset Reference™

<date>February 2010</date>

`print_backtrace``print_backtrace(3stap)`

Name

`print_backtrace` -- Print stack back trace

`print_backtrace` -- Print stack back trace

Synopsis

```
print_backtrace()
```

```
print_backtrace()
```

Arguments

None

None

Description

Equivalent to `<command>print_stack(backtrace)</command>`, except that deeper stack nesting may be supported. Return nothing.

Equivalent to `<command>print_stack(backtrace)</command>`, except that deeper stack nesting may be supported. Return nothing.

`backtrace`

Name

backtrace -- Hex backtrace of current stack

Synopsis

```
backtrace:string()
```

Arguments

None

Description

Return a string of hex addresses that are a backtrace of the stack. Output may be truncated as per maximum string length.

SystemTap Tapset Reference™

<date>February 2010</date>

backtracebacktrace(3stap)

Name

backtrace -- Hex backtrace of current stack

backtrace -- Hex backtrace of current stack

Synopsis

```
backtrace:string()
```

```
backtrace:string()
```

Arguments

None

None

Description

Return a string of hex addresses that are a backtrace of the stack. Output may be truncated as per maximum string length.

Return a string of hex addresses that are a backtrace of the stack. Output may be truncated as per maximum string length.

Name

caller -- Return name and address of calling function

Synopsis

```
caller:string()
```

Arguments

None

Description

Return the address and name of the calling function. **<emphasis>Works only for return probes at this time.</emphasis>**

SystemTap Tapset Reference™

<date>February 2010</date>

callercaller(3stap)

Name

caller -- Return name and address of calling function

caller -- Return name and address of calling function

Synopsis

```
caller:string()
```

```
caller:string()
```

Arguments

None

None

Description

Return the address and name of the calling function. **<emphasis>Works only for return probes at this time.</emphasis>**

Return the address and name of the calling function. **<emphasis>Works only for return probes at this time.</emphasis>**

Name

caller_addr -- Return caller address

Synopsis

```
caller_addr:long()
```

Arguments

None

Description

Return the address of the calling function. <emphasis> Works only for return probes at this time.</emphasis>
SystemTap Tapset Reference™

<date>February 2010</date>

caller_addr caller_addr(3stap)

Name

caller_addr -- Return caller address

caller_addr -- Return caller address

Synopsis

```
caller_addr:long()
```

```
caller_addr:long()
```

Arguments

None

None

Description

Return the address of the calling function. <emphasis> Works only for return probes at this time.</emphasis>

Return the address of the calling function. <emphasis> Works only for return probes at this time.</emphasis>

Chapter 3. Timestamp Functions

Each timestamp function returns a value to indicate when a function is executed. These returned values can then be used to indicate when an event occurred, provide an ordering for events, or compute the amount of time elapsed between two time stamps.

Name

`get_cycles` -- Processor cycle count.

Synopsis

```
get_cycles:long()
```

Arguments

None

Description

Return the processor cycle counter value, or 0 if unavailable.

Timestamp Functions

Each timestamp function returns a value to indicate when a function is executed. These returned values can then be used to indicate when an event occurred, provide an ordering for events, or compute the amount of time elapsed between two time stamps.

Name

get_cycles -- Processor cycle count.

Synopsis

```
get_cycles:long()
```

Arguments

None

Description

Return the processor cycle counter value, or 0 if unavailable.

SystemTap Tapset Reference™

<date>February 2010</date>

get_cyclesget_cycles(3stap)

Name

get_cycles -- Processor cycle count.

get_cycles -- Processor cycle count.

Synopsis

```
get_cycles:long()
```

```
get_cycles:long()
```

Arguments

None

None

Description

Return the processor cycle counter value, or 0 if unavailable.

Return the processor cycle counter value, or 0 if unavailable.

Chapter 4. Memory Tapset

This family of probe points is used to probe memory-related events. It contains the following probe points:

Name

`vm_fault_contains` -- Test return value for page fault reason

Synopsis

```
vm_fault_contains:long (value:long, test:long)
```

Arguments

value *value*

The fault_type returned by `vm.page_fault.return`

test *test*

The type of fault to test for (VM_FAULT_OOM or similar)

Name

vm.pagefault -- Records that a page fault occurred.

Synopsis

vm.pagefault

Values

<i>write_access</i>	<i>write_access</i>
	Indicates whether this was a write or read access; <code><command>1</command></code> indicates a write, while <code><command>0</command></code> indicates a read.

<i>address</i>	<i>address</i>
	The address of the faulting memory access; i.e. the address that caused the page fault.

Context

The process which triggered the fault

Name

`vm.pagefault.return` -- Indicates what type of fault occurred.

Synopsis

`vm.pagefault.return`

Values

<i>fault_type</i>	<i>fault_type</i>
0	VM_FAULT_OOM
2	VM_FAULT_MINOR
3	VM_FAULT_MAJOR
1	VM_FAULT_SIGBUS

Returns either 0 (VM_FAULT_OOM) for out of memory faults, 2 (VM_FAULT_MINOR) for minor faults, 3 (VM_FAULT_MAJOR) for major faults, or 1 (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

Name

`addr_to_node` -- Returns which node a given address belongs to within a NUMA system.

Synopsis

```
addr_to_node:long(addr:long)
```

Arguments

addr *addr*

The address of the faulting memory access.

Name

vm.write_shared -- Attempts at writing to a shared page.

Synopsis

```
vm.write_shared
```

Values

```
address    address
    The address of the shared write.
```

Context

The context is the process attempting the write.

Description

Fires when a process attempts to write to a shared page. If a copy is necessary, this will be followed by a <command>vm.write_shared_copy</command>.

Name

`vm.write_shared_copy` -- Page copy for shared page write.

Synopsis

`vm.write_shared_copy`

Values

zero *zero*
Boolean indicating whether it is a zero page (can do a clear instead of a copy).

address *address*
The address of the shared write.

Context

The process attempting the write.

Description

Fires when a write to a shared page requires a page copy. This is always preceded by a
<command>vm.shared_write</command>.

Name

vm.mmap -- Fires when an <command>mmap</command> is requested.

Synopsis

vm.mmap

Values

length *length*
The length of the memory segment

address *address*
The requested address

Context

The process calling <command>mmap</command>.

Name

vm.munmap -- Fires when an <command>munmap</command> is requested.

Synopsis

vm.munmap

Values

length *length*
The length of the memory segment

address *address*
The requested address

Context

The process calling <command>munmap</command>.

Name

vm.brk -- Fires when a <command>brk</command> is requested (i.e. the heap will be resized).

Synopsis

vm.brk

Values

length *length*
The length of the memory segment

address *address*
The requested address

Context

The process calling <command>brk</command>.

Name

vm.oom_kill -- Fires when a thread is selected for termination by the OOM killer.

Synopsis

```
vm.oom_kill
```

Values

```
task task
```

The task being killed

Context

The process that tried to consume excessive memory, and thus triggered the OOM. <remark>(is this correct?)</remark>

This family of probe points is used to probe memory-related events. It contains the following probe points:

Name

vm_fault_contains -- Test return value for page fault reason

Synopsis

```
vm_fault_contains:long (value:long, test:long)
```

Arguments

<i>value</i>	<i>value</i> The fault_type returned by vm.page_fault.return
<i>test</i>	<i>test</i> The type of fault to test for (VM_FAULT_OOM or similar)

SystemTap Tapset Reference™

<date>February 2010</date>

vm_fault_containsvm_fault_contains(3stap)

Name

vm_fault_contains -- Test return value for page fault reason

vm_fault_contains -- Test return value for page fault reason

Synopsis

```
vm_fault_contains:long (value:long, test:long)
```

```
vm_fault_contains:long (value:long, test:long)
```

Arguments

<i>value</i>	<i>value</i> The fault_type returned by vm.page_fault.return
<i>test</i>	<i>test</i> The type of fault to test for (VM_FAULT_OOM or similar)

<i>value</i>	<i>value</i> The fault_type returned by vm.page_fault.return
<i>test</i>	<i>test</i> The type of fault to test for (VM_FAULT_OOM or similar)

<varlistentry>*value*

The fault_type returned by vm.page_fault.return

</varlistentry>

valuevalue

The fault_type returned by vm.page_fault.return

The fault_type returned by vm.page_fault.return

<varlistentry>*test*

The type of fault to test for (VM_FAULT_OOM or similar)

</varlistentry>

testtest

The type of fault to test for (VM_FAULT_OOM or similar)

The type of fault to test for (VM_FAULT_OOM or similar)

Name

vm.pagefault -- Records that a page fault occurred.

Synopsis

vm.pagefault

Values

write_access

write_access

Indicates whether this was a write or read access; `<command>1</command>` indicates a write, while `<command>0</command>` indicates a read.

address

address

The address of the faulting memory access; i.e. the address that caused the page fault.

Context

The process which triggered the fault

SystemTap Tapset Reference™

`<date>February 2010</date>`

vm.pagefaultvm.pagefault(3stap)

Name

vm.pagefault -- Records that a page fault occurred.

vm.pagefault -- Records that a page fault occurred.

Synopsis

vm.pagefault

vm.pagefault

Values

write_access

write_access

Indicates whether this was a write or read access; `<command>1</command>` indicates a write, while `<command>0</command>` indicates a read.

address

address

The address of the faulting memory access; i.e. the address that caused the page fault.

write_access

write_access

Indicates whether this was a write or read access; `<command>1</command>` indicates a write, while `<command>0</command>` indicates a read.

address

address

The address of the faulting memory access; i.e. the address that caused the page fault.

<varlistentry>write_access

Indicates whether this was a write or read access; *<command>1</command>* indicates a write, while *<command>0</command>* indicates a read.

</varlistentry>

write_accesswrite_access

Indicates whether this was a write or read access; *<command>1</command>* indicates a write, while *<command>0</command>* indicates a read.

Indicates whether this was a write or read access; *<command>1</command>* indicates a write, while *<command>0</command>* indicates a read.

<varlistentry>address

The address of the faulting memory access; i.e. the address that caused the page fault.

</varlistentry>

addressaddress

The address of the faulting memory access; i.e. the address that caused the page fault.

The address of the faulting memory access; i.e. the address that caused the page fault.

Context

The process which triggered the fault

The process which triggered the fault

Name

vm.pagefault.return -- Indicates what type of fault occurred.

Synopsis

vm.pagefault.return

Values

fault_type

fault_type

Returns either `<command>0</command>` (VM_FAULT_OOM) for out of memory faults, `<command>2</command>` (VM_FAULT_MINOR) for minor faults, `<command>3</command>` (VM_FAULT_MAJOR) for major faults, or `<command>1</command>` (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

SystemTap Tapset Reference™

`<date>February 2010</date>`

vm.pagefault.returnvm.pagefault.return(3stap)

Name

vm.pagefault.return -- Indicates what type of fault occurred.

vm.pagefault.return -- Indicates what type of fault occurred.

Synopsis

vm.pagefault.return

vm.pagefault.return

Values

fault_type

fault_type

Returns either `<command>0</command>` (VM_FAULT_OOM) for out of memory faults, `<command>2</command>` (VM_FAULT_MINOR) for minor faults, `<command>3</command>` (VM_FAULT_MAJOR) for major faults, or `<command>1</command>` (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

fault_type

fault_type

Returns either `<command>0</command>` (VM_FAULT_OOM) for out of memory faults, `<command>2</command>` (VM_FAULT_MINOR) for minor faults, `<command>3</command>` (VM_FAULT_MAJOR) for major faults, or `<command>1</command>` (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

<varlistentry>fault_type

Returns either *<command>0</command>* (VM_FAULT_OOM) for out of memory faults, *<command>2</command>* (VM_FAULT_MINOR) for minor faults, *<command>3</command>* (VM_FAULT_MAJOR) for major faults, or *<command>1</command>* (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

</varlistentry>

fault_typefault_type

Returns either *<command>0</command>* (VM_FAULT_OOM) for out of memory faults, *<command>2</command>* (VM_FAULT_MINOR) for minor faults, *<command>3</command>* (VM_FAULT_MAJOR) for major faults, or *<command>1</command>* (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

Returns either *<command>0</command>* (VM_FAULT_OOM) for out of memory faults, *<command>2</command>* (VM_FAULT_MINOR) for minor faults, *<command>3</command>* (VM_FAULT_MAJOR) for major faults, or *<command>1</command>* (VM_FAULT_SIGBUS) if the fault was neither OOM, minor fault, nor major fault.

Name

`addr_to_node` -- Returns which node a given address belongs to within a NUMA system.

Synopsis

```
addr_to_node:long(addr:long)
```

Arguments

addr *addr*
The address of the faulting memory access.

SystemTap Tapset Reference™

<date>February 2010</date>

`addr_to_node``addr_to_node(3stap)`

Name

`addr_to_node` -- Returns which node a given address belongs to within a NUMA system.

`addr_to_node` -- Returns which node a given address belongs to within a NUMA system.

Synopsis

```
addr_to_node:long(addr:long)
```

```
addr_to_node:long(addr:long)
```

Arguments

addr *addr*
The address of the faulting memory access.

addr *addr*
The address of the faulting memory access.

<varlistentry>*addr*

The address of the faulting memory access.

</varlistentry>

*addr**addr*

The address of the faulting memory access.

The address of the faulting memory access.

Name

vm.write_shared -- Attempts at writing to a shared page.

Synopsis

vm.write_shared

Values

<i>address</i>	<i>address</i>
	The address of the shared write.

Context

The context is the process attempting the write.

Description

Fires when a process attempts to write to a shared page. If a copy is necessary, this will be followed by a `<command>vm.write_shared_copy</command>`.

SystemTap Tapset Reference™

<date>February 2010</date>

vm.write_sharedvm.write_shared(3stap)

Name

vm.write_shared -- Attempts at writing to a shared page.

vm.write_shared -- Attempts at writing to a shared page.

Synopsis

vm.write_shared

vm.write_shared

Values

<i>address</i>	<i>address</i>
	The address of the shared write.

<i>address</i>	<i>address</i>
	The address of the shared write.

```
<varlistentry>address
The address of the shared write.
</varlistentry>
addressaddress
The address of the shared write.
The address of the shared write.
```

Context

The context is the process attempting the write.

The context is the process attempting the write.

Description

Fires when a process attempts to write to a shared page. If a copy is necessary, this will be followed by a `<command>vm.write_shared_copy</command>`.

Fires when a process attempts to write to a shared page. If a copy is necessary, this will be followed by a `<command>vm.write_shared_copy</command>`.

Name

vm.write_shared_copy -- Page copy for shared page write.

Synopsis

vm.write_shared_copy

Values

<i>zero</i>	<i>zero</i>
	Boolean indicating whether it is a zero page (can do a clear instead of a copy).
<i>address</i>	<i>address</i>
	The address of the shared write.

Context

The process attempting the write.

Description

Fires when a write to a shared page requires a page copy.

This is always preceded by a

<command>vm.shared_write</command>.

SystemTap Tapset Reference™

<date>February 2010</date>

vm.write_shared_copyvm.write_shared_copy(3stap)

Name

vm.write_shared_copy -- Page copy for shared page write.

vm.write_shared_copy -- Page copy for shared page write.

Synopsis

vm.write_shared_copy

vm.write_shared_copy

Values

<i>zero</i>	<i>zero</i>
	Boolean indicating whether it is a zero page (can do a clear instead of a copy).
<i>address</i>	<i>address</i>
	The address of the shared write.

<i>zero</i>	<i>zero</i>
	Boolean indicating whether it is a zero page (can do a clear instead of a copy).

address *address*
The address of the shared write.

<varlistentry>*zero*
Boolean indicating whether it is a zero page (can do a clear instead of a copy).
</varlistentry>
zerozero
Boolean indicating whether it is a zero page (can do a clear instead of a copy).
Boolean indicating whether it is a zero page (can do a clear instead of a copy).
<varlistentry>*address*
The address of the shared write.
</varlistentry>
addressaddress
The address of the shared write.
The address of the shared write.

Context

The process attempting the write.

The process attempting the write.

Description

Fires when a write to a shared page requires a page copy. This is always preceded by a
<command>vm.shared_write**</command>**.

Fires when a write to a shared page requires a page copy. This is always preceded by a
<command>vm.shared_write**</command>**.

Name

vm.mmap -- Fires when an <command>mmap</command> is requested.

Synopsis

vm.mmap

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

Context

The process calling <command>mmap</command>.
SystemTap Tapset Reference™
<date>February 2010</date>
vm.mmapvm.mmap(3stap)

Name

vm.mmap -- Fires when an <command>mmap</command> is requested.

vm.mmap -- Fires when an <command>mmap</command> is requested.

Synopsis

vm.mmap

vm.mmap

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<varlistentry>length
The length of the memory segment
</varlistentry>
lengthlength
The length of the memory segment
The length of the memory segment
<varlistentry>address
The requested address
</varlistentry>
addressaddress
The requested address
The requested address

Context

The process calling *<command>mmap</command>*.

The process calling *<command>mmap</command>*.

Name

vm.munmap -- Fires when an <command>munmap</command> is requested.

Synopsis

vm.munmap

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

Context

The process calling <command>munmap</command>.
SystemTap Tapset Reference™
<date>February 2010</date>
vm.munmapvm.munmap(3stap)

Name

vm.munmap -- Fires when an <command>munmap</command> is requested.

vm.munmap -- Fires when an <command>munmap</command> is requested.

Synopsis

vm.munmap

vm.munmap

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<varlistentry>length
The length of the memory segment
</varlistentry>
lengthlength
The length of the memory segment
The length of the memory segment
<varlistentry>address
The requested address
</varlistentry>
addressaddress
The requested address
The requested address

Context

The process calling *<command>munmap</command>*.

The process calling *<command>munmap</command>*.

Name

vm.brk -- Fires when a `<command>brk</command>` is requested (i.e. the heap will be resized).

Synopsis

vm.brk

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

Context

The process calling `<command>brk</command>`.
SystemTap Tapset Reference™
<date>February 2010</date>
vm.brkvm.brk(3stap)

Name

vm.brk -- Fires when a `<command>brk</command>` is requested (i.e. the heap will be resized).

vm.brk -- Fires when a `<command>brk</command>` is requested (i.e. the heap will be resized).

Synopsis

vm.brk

vm.brk

Values

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<i>length</i>	<i>length</i> The length of the memory segment
<i>address</i>	<i>address</i> The requested address

<varlistentry>length
The length of the memory segment
</varlistentry>
lengthlength
The length of the memory segment
The length of the memory segment
<varlistentry>address
The requested address
</varlistentry>
addressaddress
The requested address
The requested address

Context

The process calling *<command>brk</command>*.

The process calling *<command>brk</command>*.

Name

vm.oom_kill -- Fires when a thread is selected for termination by the OOM killer.

Synopsis

```
vm.oom_kill
```

Values

<i>task</i>	<i>task</i>
	The task being killed

Context

The process that tried to consume excessive memory, and thus triggered the OOM. <remark>(is this correct?)</remark>

SystemTap Tapset Reference™

<date>February 2010</date>

vm.oom_killvm.oom_kill(3stap)

Name

vm.oom_kill -- Fires when a thread is selected for termination by the OOM killer.

vm.oom_kill -- Fires when a thread is selected for termination by the OOM killer.

Synopsis

```
vm.oom_kill
```

```
vm.oom_kill
```

Values

<i>task</i>	<i>task</i>
	The task being killed

<i>task</i>	<i>task</i>
	The task being killed

<varlistentry>*task*
The task being killed
</varlistentry>
tasktask
The task being killed
The task being killed

Context

The process that tried to consume excessive memory, and thus triggered the OOM. <remark>(is this correct?)</remark>

The process that tried to consume excessive memory, and thus triggered the OOM. <remark>(is this correct?)</remark>

Chapter 5. IO Scheduler Tapset

This family of probe points is used to probe IO scheduler activities. It contains the following probe points:

Name

`ioscheduler.elv_next_request` -- Fires when a request is retrieved from the request queue

Synopsis

`ioscheduler.elv_next_request`

Values

elevator_name *elevator_name*
The type of I/O elevator currently enabled

Name

`ioscheduler.elv_next_request.return` -- Fires when a request retrieval issues a return signal

Synopsis

```
ioscheduler.elv_next_request.return
```

Values

<i>req_flags</i>	<i>req_flags</i>
Request flags	
<i>req</i>	<i>req</i>
Address of the request	
<i>disk_major</i>	<i>disk_major</i>
Disk major number of the request	
<i>disk_minor</i>	<i>disk_minor</i>
Disk minor number of the request	

Name

`ioscheduler.elv_add_request` -- A request was added to the request queue

Synopsis

`ioscheduler.elv_add_request`

Values

<i>req_flags</i>	<i>req_flags</i>
	Request flags
<i>req</i>	<i>req</i>
	Address of the request
<i>disk_major</i>	<i>disk_major</i>
	Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i>
	The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i>
	Disk minor number of the request

Name

`ioscheduler.elv_completed_request` -- Fires when a request is completed

Synopsis

`ioscheduler.elv_completed_request`

Values

<i>req_flags</i>	<i>req_flags</i>	Request flags
<i>req</i>	<i>req</i>	Address of the request
<i>disk_major</i>	<i>disk_major</i>	Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i>	The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i>	Disk minor number of the request

This family of probe points is used to probe IO scheduler activities. It contains the following probe points:

Name

ioscheduler.elv_next_request -- Fires when a request is retrieved from the request queue

Synopsis

```
ioscheduler.elv_next_request
```

Values

<i>elevator_name</i>	<i>elevator_name</i>
	The type of I/O elevator currently enabled

SystemTap Tapset Reference™

<date>February 2010</date>

ioscheduler.elv_next_requestioscheduler.elv_next_request(3stap)

Name

ioscheduler.elv_next_request -- Fires when a request is retrieved from the request queue

ioscheduler.elv_next_request -- Fires when a request is retrieved from the request queue

Synopsis

```
ioscheduler.elv_next_request
```

```
ioscheduler.elv_next_request
```

Values

<i>elevator_name</i>	<i>elevator_name</i>
	The type of I/O elevator currently enabled

<i>elevator_name</i>	<i>elevator_name</i>
	The type of I/O elevator currently enabled

<varlistentry>*elevator_name*
The type of I/O elevator currently enabled
</varlistentry>

*elevator_name**elevator_name*
The type of I/O elevator currently enabled
The type of I/O elevator currently enabled

Name

`ioscheduler.elv_next_request.return` -- Fires when a request retrieval issues a return signal

Synopsis

```
ioscheduler.elv_next_request.return
```

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

SystemTap Tapset Reference™

<date>February 2010</date>

`ioscheduler.elv_next_request.returnioscheduler.elv_next_request.return(3stap)`

Name

`ioscheduler.elv_next_request.return` -- Fires when a request retrieval issues a return signal

`ioscheduler.elv_next_request.return` -- Fires when a request retrieval issues a return signal

Synopsis

```
ioscheduler.elv_next_request.return
```

```
ioscheduler.elv_next_request.return
```

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

<varlistentry>*req_flags*

Request flags

</varlistentry>

req_flagsreq_flags

Request flags

Request flags

<varlistentry>*req*

Address of the request

</varlistentry>

reqreq

Address of the request

Address of the request

<varlistentry>*disk_major*

Disk major number of the request

</varlistentry>

disk_majordisk_major

Disk major number of the request

Disk major number of the request

<varlistentry>*disk_minor*

Disk minor number of the request

</varlistentry>

disk_minordisk_minor

Disk minor number of the request

Disk minor number of the request

Name

ioscheduler.elv_add_request -- A request was added to the request queue

Synopsis

```
ioscheduler.elv_add_request
```

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

SystemTap Tapset Reference™

<date>February 2010</date>

ioscheduler.elv_add_requestioscheduler.elv_add_request(3stap)

Name

ioscheduler.elv_add_request -- A request was added to the request queue

ioscheduler.elv_add_request -- A request was added to the request queue

Synopsis

```
ioscheduler.elv_add_request
```

```
ioscheduler.elv_add_request
```

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled

<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request
<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

```

<varlistentry>req_flags
Request flags
</varlistentry>
req_flagsreq_flags
Request flags
Request flags
<varlistentry>req
Address of the request
</varlistentry>
reqreq
Address of the request
Address of the request
<varlistentry>disk_major
Disk major number of the request
</varlistentry>
disk_majordisk_major
Disk major number of the request
Disk major number of the request
<varlistentry>elevator_name
The type of I/O elevator currently enabled
</varlistentry>
elevator_nameelevator_name
The type of I/O elevator currently enabled
The type of I/O elevator currently enabled
<varlistentry>disk_minor
Disk minor number of the request
</varlistentry>
disk_minordisk_minor
Disk minor number of the request
Disk minor number of the request

```

Name

ioscheduler.elv_completed_request -- Fires when a request is completed

Synopsis

ioscheduler.elv_completed_request

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

SystemTap Tapset Reference™

<date>February 2010</date>

ioscheduler.elv_completed_requestioscheduler.elv_completed_request(3stap)

Name

ioscheduler.elv_completed_request -- Fires when a request is completed

ioscheduler.elv_completed_request -- Fires when a request is completed

Synopsis

ioscheduler.elv_completed_request

ioscheduler.elv_completed_request

Values

<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled

<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request
<i>req_flags</i>	<i>req_flags</i> Request flags
<i>req</i>	<i>req</i> Address of the request
<i>disk_major</i>	<i>disk_major</i> Disk major number of the request
<i>elevator_name</i>	<i>elevator_name</i> The type of I/O elevator currently enabled
<i>disk_minor</i>	<i>disk_minor</i> Disk minor number of the request

```

<varlistentry>req_flags
Request flags
</varlistentry>
req_flagsreq_flags
Request flags
Request flags
<varlistentry>req
Address of the request
</varlistentry>
reqreq
Address of the request
Address of the request
<varlistentry>disk_major
Disk major number of the request
</varlistentry>
disk_majordisk_major
Disk major number of the request
Disk major number of the request
<varlistentry>elevator_name
The type of I/O elevator currently enabled
</varlistentry>
elevator_nameelevator_name
The type of I/O elevator currently enabled
The type of I/O elevator currently enabled
<varlistentry>disk_minor
Disk minor number of the request
</varlistentry>
disk_minordisk_minor
Disk minor number of the request
Disk minor number of the request

```

Chapter 6. SCSI Tapset

This family of probe points is used to probe SCSI activities. It contains the following probe points:

Name

scsi.ioentry -- Prepares a SCSI mid-layer request

Synopsis

```
scsi.ioentry
```

Values

<i>disk_major</i>	<i>disk_major</i>
The major number of the disk (-1 if no information)	

<i>device_state</i>	<i>device_state</i>
The current state of the device.	

<i>disk_minor</i>	<i>disk_minor</i>
The minor number of the disk (-1 if no information)	

Name

`scsi.iodyspatching` -- SCSI mid-layer dispatched low-level SCSI command

Synopsis

`scsi.iodyspatching`

Values

<i>lun</i>	<i>lun</i>	The lun number
<i>req_bufflen</i>	<i>req_bufflen</i>	The request buffer length
<i>host_no</i>	<i>host_no</i>	The host number
<i>device_state</i>	<i>device_state</i>	The current state of the device.
<i>dev_id</i>	<i>dev_id</i>	The scsi device id
<i>channel</i>	<i>channel</i>	The channel number
<i>data_direction</i>	<i>data_direction</i>	The <i>data_direction</i> specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1 (DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
<i>request_buffer</i>	<i>request_buffer</i>	The request buffer address

Name

`scsi.iodone` -- SCSI command completed by low level driver and enqueued into the done queue.

Synopsis

`scsi.iodone`

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The <code>data_direction</code> specifies whether this command is from/to the device.

Name

`scsi.iocompleted` -- SCSI mid-layer running the completion processing for block device I/O requests

Synopsis

```
scsi.iocompleted
```

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The <code>data_direction</code> specifies whether this command is from/to the device
<i>goodbytes</i>	<i>goodbytes</i> The bytes completed.

This family of probe points is used to probe SCSI activities. It contains the following probe points:

Name

scsi.ioentry -- Prepares a SCSI mid-layer request

Synopsis

scsi.ioentry

Values

<i>disk_major</i>	<i>disk_major</i> The major number of the disk (-1 if no information)
<i>device_state</i>	<i>device_state</i> The current state of the device.
<i>disk_minor</i>	<i>disk_minor</i> The minor number of the disk (-1 if no information)

SystemTap Tapset Reference™

<date>February 2010</date>

scsi.ioentryscsi.ioentry(3stap)

Name

scsi.ioentry -- Prepares a SCSI mid-layer request

scsi.ioentry -- Prepares a SCSI mid-layer request

Synopsis

scsi.ioentry

scsi.ioentry

Values

<i>disk_major</i>	<i>disk_major</i> The major number of the disk (-1 if no information)
<i>device_state</i>	<i>device_state</i> The current state of the device.
<i>disk_minor</i>	<i>disk_minor</i> The minor number of the disk (-1 if no information)

<i>disk_major</i>	<i>disk_major</i> The major number of the disk (-1 if no information)
<i>device_state</i>	<i>device_state</i> The current state of the device.

disk_minor

disk_minor

The minor number of the disk (-1 if no information)

<varlistentry>*disk_major*

The major number of the disk (-1 if no information)

</varlistentry>

disk_majordisk_major

The major number of the disk (-1 if no information)

The major number of the disk (-1 if no information)

<varlistentry>*device_state*

The current state of the device.

</varlistentry>

device_statedevice_state

The current state of the device.

The current state of the device.

<varlistentry>*disk_minor*

The minor number of the disk (-1 if no information)

</varlistentry>

disk_minordisk_minor

The minor number of the disk (-1 if no information)

The minor number of the disk (-1 if no information)

Name

scsi.iodispatching -- SCSI mid-layer dispatched low-level SCSI command

Synopsis

scsi.iodispatching

Values

<i>lun</i>	<i>lun</i> The lun number
<i>req_bufflen</i>	<i>req_bufflen</i> The request buffer length
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device.
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The <i>data_direction</i> specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1 (DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
<i>request_buffer</i>	<i>request_buffer</i> The request buffer address

SystemTap Tapset Reference™

<date>February 2010</date>

scsi.iodispatchingscsi.iodispatching(3stap)

Name

scsi.iodispatching -- SCSI mid-layer dispatched low-level SCSI command

scsi.iodispatching -- SCSI mid-layer dispatched low-level SCSI command

Synopsis

scsi.iodispatching

scsi.iodispatching

Values

<i>lun</i>	<i>lun</i> The lun number
<i>req_bufflen</i>	<i>req_bufflen</i> The request buffer length
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device.
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The <i>data_direction</i> specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1 (DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
<i>request_buffer</i>	<i>request_buffer</i> The request buffer address
<i>lun</i>	<i>lun</i> The lun number
<i>req_bufflen</i>	<i>req_bufflen</i> The request buffer length
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device.
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The <i>data_direction</i> specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1 (DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
<i>request_buffer</i>	<i>request_buffer</i> The request buffer address

```

<varlistentry>lun
The lun number
</varlistentry>
lunlun
The lun number
The lun number
<varlistentry>req_bufflen
The request buffer length
</varlistentry>
req_bufflenreq_bufflen
The request buffer length
The request buffer length
<varlistentry>host_no
The host number
</varlistentry>
host_nohost_no
The host number
The host number
<varlistentry>device_state
The current state of the device.
</varlistentry>
device_statedevice_state
The current state of the device.
The current state of the device.
<varlistentry>dev_id
The scsi device id
</varlistentry>
dev_iddev_id
The scsi device id
The scsi device id
<varlistentry>channel
The channel number
</varlistentry>
channelchannel
The channel number
The channel number
<varlistentry>data_direction
The data_direction specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1
(DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
</varlistentry>
data_directiondata_direction
The data_direction specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1
(DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
The data_direction specifies whether this command is from/to the device. 0 (DMA_BIDIRECTIONAL), 1
(DMA_TO_DEVICE), 2 (DMA_FROM_DEVICE), 3 (DMA_NONE)
<varlistentry>request_buffer
The request buffer address
</varlistentry>
request_bufferrequest_buffer
The request buffer address
The request buffer address

```

Name

scsi.iodone -- SCSI command completed by low level driver and enqueued into the done queue.

Synopsis

scsi.iodone

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device.

SystemTap Tapset Reference™

<date>February 2010</date>

scsi.iodonescsi.iodone(3stap)

Name

scsi.iodone -- SCSI command completed by low level driver and enqueued into the done queue.

scsi.iodone -- SCSI command completed by low level driver and enqueued into the done queue.

Synopsis

scsi.iodone

scsi.iodone

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device

SCSI Tapset

<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device.
<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device.

```
<varlistentry>lun
The lun number
</varlistentry>
lunlun
The lun number
The lun number
<varlistentry>host_no
The host number
</varlistentry>
host_nohost_no
The host number
The host number
<varlistentry>device_state
The current state of the device
</varlistentry>
device_statedevice_state
The current state of the device
The current state of the device
<varlistentry>dev_id
The scsi device id
</varlistentry>
dev_iddev_id
The scsi device id
The scsi device id
<varlistentry>channel
The channel number
</varlistentry>
channelchannel
The channel number
The channel number
<varlistentry>data_direction
```

The `data_direction` specifies whether this command is from/to the device.

</varlistentry>

data_directiondata_direction

The `data_direction` specifies whether this command is from/to the device.

The `data_direction` specifies whether this command is from/to the device.

Name

scsi.iocompleted -- SCSI mid-layer running the completion processing for block device I/O requests

Synopsis

scsi.iocompleted

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device
<i>goodbytes</i>	<i>goodbytes</i> The bytes completed.

SystemTap Tapset Reference™

<date>February 2010</date>

scsi.iocompletedscsi.iocompleted(3stap)

Name

scsi.iocompleted -- SCSI mid-layer running the completion processing for block device I/O requests

scsi.iocompleted -- SCSI mid-layer running the completion processing for block device I/O requests

Synopsis

scsi.iocompleted

scsi.iocompleted

Values

<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number

<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device
<i>goodbytes</i>	<i>goodbytes</i> The bytes completed.
<i>lun</i>	<i>lun</i> The lun number
<i>host_no</i>	<i>host_no</i> The host number
<i>device_state</i>	<i>device_state</i> The current state of the device
<i>dev_id</i>	<i>dev_id</i> The scsi device id
<i>channel</i>	<i>channel</i> The channel number
<i>data_direction</i>	<i>data_direction</i> The data_direction specifies whether this command is from/to the device
<i>goodbytes</i>	<i>goodbytes</i> The bytes completed.

<varlistentry>lun

The lun number

</varlistentry>

lunlun

The lun number

The lun number

<varlistentry>host_no

The host number

</varlistentry>

host_nohost_no

The host number

The host number

<varlistentry>device_state

The current state of the device

</varlistentry>

device_statedevice_state

The current state of the device

The current state of the device

<varlistentry>dev_id

The scsi device id

</varlistentry>

dev_iddev_id

The scsi device id

The scsi device id

<varlistentry>*channel*

The channel number

</varlistentry>

channelchannel

The channel number

The channel number

<varlistentry>*data_direction*

The *data_direction* specifies whether this command is from/to the device

</varlistentry>

data_directiondata_direction

The *data_direction* specifies whether this command is from/to the device

The *data_direction* specifies whether this command is from/to the device

<varlistentry>*goodbytes*

The bytes completed.

</varlistentry>

goodbytesgoodbytes

The bytes completed.

The bytes completed.

Chapter 7. Networking Tapset

This family of probe points is used to probe the activities of the network device and protocol layers.

Name

`netdev.receive` -- Data recieved from network device.

Synopsis

`netdev.receive`

Values

protocol *protocol*
Protocol of recieved packet.

dev_name *dev_name*
The name of the device. e.g: eth0, ath1.

length *length*
The length of the receiving buffer.

Name

`netdev.transmit` -- Network device transmitting buffer

Synopsis

`netdev.transmit`

Values

protocol *protocol*
The protocol of this packet.

dev_name *dev_name*
The name of the device. e.g: eth0, ath1.

length *length*
The length of the transmit buffer.

true_size *true_size*
The size of the the data to be transmitted.

Name

tcp.sendmsg -- Sending a tcp message

Synopsis

tcp.sendmsg

Values

name *name*
Name of this probe

size *size*
Number of bytes to send

sock *sock*
Network socket

Context

The process which sends a tcp message

Name

tcp.sendmsg.return -- Sending TCP message is done

Synopsis

```
tcp.sendmsg.return
```

Values

name *name*
Name of this probe

size *size*
Number of bytes sent or error code if an error occurred.

Context

The process which sends a tcp message

Name

tcp.recvmsg -- Receiving TCP message

Synopsis

tcp.recvmsg

Values

saddr saddr

A string representing the source IP address

daddr daddr

A string representing the destination IP address

name name

Name of this probe

sport sport

TCP source port

dport dport

TCP destination port

size size

Number of bytes to be received

sock sock

Network socket

Context

The process which receives a tcp message

Name

tcp.recvmsg.return -- Receiving TCP message complete

Synopsis

```
tcp.recvmsg.return
```

Values

saddr *saddr*

A string representing the source IP address

daddr *daddr*

A string representing the destination IP address

name *name*

Name of this probe

sport *sport*

TCP source port

dport *dport*

TCP destination port

size *size*

Number of bytes received or error code if an error occurred.

Context

The process which receives a tcp message

Name

tcp.disconnect -- TCP socket disconnection

Synopsis

tcp.disconnect

Values

saddr saddr

A string representing the source IP address

daddr daddr

A string representing the destination IP address

flags flags

TCP flags (e.g. FIN, etc)

name name

Name of this probe

sport sport

TCP source port

dport dport

TCP destination port

sock sock

Network socket

Context

The process which disconnects tcp

Name

tcp.disconnect.return -- TCP socket disconnection complete

Synopsis

```
tcp.disconnect.return
```

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

Context

The process which disconnects tcp

Name

tcp.setsockopt -- Call to setsockopt

Synopsis

tcp.setsockopt

Values

<i>optstr</i>	<i>optstr</i>
Resolves optname to a human-readable format	
<i>level</i>	<i>level</i>
The level at which the socket options will be manipulated	
<i>optlen</i>	<i>optlen</i>
Used to access values for setsockopt	
<i>name</i>	<i>name</i>
Name of this probe	
<i>optname</i>	<i>optname</i>
TCP socket options (e.g. TCP_NODELAY, TCP_MAXSEG, etc)	
<i>sock</i>	<i>sock</i>
Network socket	

Context

The process which calls setsockopt

Name

`tcp.setsockopt.return` -- Return from `setsockopt`

Synopsis

`tcp.setsockopt.return`

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

Context

The process which calls `setsockopt`

Name

`tcp.receive` -- Called when a TCP packet is received

Synopsis

`tcp.receive`

Values

urg *urg*
TCP URG flag

psh *psh*
TCP PSH flag

rst *rst*
TCP RST flag

dport *dport*
TCP destination port

saddr *saddr*
A string representing the source IP address

daddr *daddr*
A string representing the destination IP address

ack *ack*
TCP ACK flag

syn *syn*
TCP SYN flag

fin *fin*
TCP FIN flag

sport *sport*
TCP source port

Name

udp.sendmsg -- Fires whenever a process sends a UDP message

Synopsis

```
udp.sendmsg
```

Values

name *name*

The name of this probe

size *size*

Number of bytes sent by the process

sock *sock*

Network socket used by the process

Context

The process which sent a UDP message

Name

`udp.sendmsg.return` -- Fires whenever an attempt to send a UDP message is completed

Synopsis

`udp.sendmsg.return`

Values

name *name*

The name of this probe

size *size*

Number of bytes sent by the process

Context

The process which sent a UDP message

Name

udp.recvmsg -- Fires whenever a UDP message is received

Synopsis

```
udp.recvmsg
```

Values

name *name*

The name of this probe

size *size*

Number of bytes received by the process

sock *sock*

Network socket used by the process

Context

The process which received a UDP message

Name

`udp.recvmsg.return` -- Fires whenever an attempt to receive a UDP message received is completed

Synopsis

`udp.recvmsg.return`

Values

name *name*

The name of this probe

size *size*

Number of bytes received by the process

Context

The process which received a UDP message

Name

udp.disconnect -- Fires when a process requests for a UDP disconnection

Synopsis

```
udp.disconnect
```

Values

flags flags
Flags (e.g. FIN, etc)

name name
The name of this probe

sock sock
Network socket used by the process

Context

The process which requests a UDP disconnection

Name

`udp.disconnect.return` -- UDP has been disconnected successfully

Synopsis

`udp.disconnect.return`

Values

ret *ret*
Error code (0: no error)

name *name*
The name of this probe

Context

The process which requested a UDP disconnection

Name

`ip_ntop` -- returns a string representation from an integer IP number

Synopsis

```
ip_ntop:string(addr:long)
```

Arguments

addr *addr*
the ip represented as an integer

This family of probe points is used to probe the activities of the network device and protocol layers.

Name

netdev.receive -- Data recieved from network device.

Synopsis

netdev.receive

Values

<i>protocol</i>	<i>protocol</i> Protocol of recieved packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.
<i>length</i>	<i>length</i> The length of the receiving buffer.

SystemTap Tapset Reference™

<date>February 2010</date>

netdev.receive
netdev.receive(3tap)

Name

netdev.receive -- Data recieved from network device.

netdev.receive -- Data recieved from network device.

Synopsis

netdev.receive

netdev.receive

Values

<i>protocol</i>	<i>protocol</i> Protocol of recieved packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.
<i>length</i>	<i>length</i> The length of the receiving buffer.

<i>protocol</i>	<i>protocol</i> Protocol of recieved packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.

length *length*
The length of the receiving buffer.

<varlistentry>*protocol*
Protocol of recieved packet.
</varlistentry>
*protocol**protocol*
Protocol of recieved packet.
Protocol of recieved packet.
<varlistentry>*dev_name*
The name of the device. e.g: eth0, ath1.
</varlistentry>
*dev_name**dev_name*
The name of the device. e.g: eth0, ath1.
The name of the device. e.g: eth0, ath1.
<varlistentry>*length*
The length of the receiving buffer.
</varlistentry>
*length**length*
The length of the receiving buffer.
The length of the receiving buffer.

Name

netdev.transmit -- Network device transmitting buffer

Synopsis

netdev.transmit

Values

<i>protocol</i>	<i>protocol</i> The protocol of this packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.
<i>length</i>	<i>length</i> The length of the transmit buffer.
<i>truesize</i>	<i>truesize</i> The size of the the data to be transmitted.

SystemTap Tapset Reference™

<date>February 2010</date>

netdev.transmitnetdev.transmit(3stap)

Name

netdev.transmit -- Network device transmitting buffer

netdev.transmit -- Network device transmitting buffer

Synopsis

netdev.transmit

netdev.transmit

Values

<i>protocol</i>	<i>protocol</i> The protocol of this packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.
<i>length</i>	<i>length</i> The length of the transmit buffer.
<i>truesize</i>	<i>truesize</i> The size of the the data to be transmitted.

<i>protocol</i>	<i>protocol</i> The protocol of this packet.
<i>dev_name</i>	<i>dev_name</i> The name of the device. e.g: eth0, ath1.
<i>length</i>	<i>length</i> The length of the transmit buffer.
<i>true_size</i>	<i>true_size</i> The size of the the data to be transmitted.

<varlistentry>*protocol*
The protocol of this packet.

</varlistentry>

*protocol**protocol*
The protocol of this packet.
The protocol of this packet.

<varlistentry>*dev_name*
The name of the device. e.g: eth0, ath1.

</varlistentry>

*dev_name**dev_name*
The name of the device. e.g: eth0, ath1.
The name of the device. e.g: eth0, ath1.

<varlistentry>*length*
The length of the transmit buffer.

</varlistentry>

*length**length*
The length of the transmit buffer.
The length of the transmit buffer.

<varlistentry>*true_size*
The size of the the data to be transmitted.

</varlistentry>

*true_size**true_size*
The size of the the data to be transmitted.
The size of the the data to be transmitted.

Name

tcp.sendmsg -- Sending a tcp message

Synopsis

tcp.sendmsg

Values

<i>name</i>	<i>name</i> Name of this probe
<i>size</i>	<i>size</i> Number of bytes to send
<i>sock</i>	<i>sock</i> Network socket

Context

The process which sends a tcp message
SystemTap Tapset Reference™
<date>February 2010</date>
tcp.sendmsgtcp.sendmsg(3stap)

Name

tcp.sendmsg -- Sending a tcp message

tcp.sendmsg -- Sending a tcp message

Synopsis

tcp.sendmsg

tcp.sendmsg

Values

<i>name</i>	<i>name</i> Name of this probe
<i>size</i>	<i>size</i> Number of bytes to send
<i>sock</i>	<i>sock</i> Network socket

<i>name</i>	<i>name</i> Name of this probe
-------------	-----------------------------------

size *size*
Number of bytes to send

sock *sock*
Network socket

<varlistentry>*name*
Name of this probe
</varlistentry>
*name**name*
Name of this probe
Name of this probe
<varlistentry>*size*
Number of bytes to send
</varlistentry>
*size**size*
Number of bytes to send
Number of bytes to send
<varlistentry>*sock*
Network socket
</varlistentry>
*sock**sock*
Network socket
Network socket

Context

The process which sends a tcp message

The process which sends a tcp message

Name

tcp.sendmsg.return -- Sending TCP message is done

Synopsis

tcp.sendmsg.return

Values

name *name*
Name of this probe

size *size*
Number of bytes sent or error code if an error occurred.

Context

The process which sends a tcp message

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.sendmsg.returntcp.sendmsg.return(3stap)

Name

tcp.sendmsg.return -- Sending TCP message is done

tcp.sendmsg.return -- Sending TCP message is done

Synopsis

tcp.sendmsg.return

tcp.sendmsg.return

Values

name *name*
Name of this probe

size *size*
Number of bytes sent or error code if an error occurred.

name *name*
Name of this probe

size *size*
Number of bytes sent or error code if an error occurred.

<varlistentry>name

Name of this probe

</varlistentry>

namenname

Name of this probe

Name of this probe

<varlistentry>size

Number of bytes sent or error code if an error occurred.

</varlistentry>

sizesize

Number of bytes sent or error code if an error occurred.

Number of bytes sent or error code if an error occurred.

Context

The process which sends a tcp message

The process which sends a tcp message

Name

tcp.recvmsg -- Receiving TCP message

Synopsis

tcp.recvmsg

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes to be received
<i>sock</i>	<i>sock</i> Network socket

Context

The process which receives a tcp message
SystemTap Tapset Reference™
<date>February 2010</date>
tcp.recvmsgtcp.recvmsg(3stap)

Name

tcp.recvmsg -- Receiving TCP message

tcp.recvmsg -- Receiving TCP message

Synopsis

tcp.recvmsg

tcp.recvmsg

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes to be received
<i>sock</i>	<i>sock</i> Network socket
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes to be received
<i>sock</i>	<i>sock</i> Network socket

```
<varlistentry>saddr
A string representing the source IP address
</varlistentry>
saddrsaddr
A string representing the source IP address
A string representing the source IP address
<varlistentry>daddr
A string representing the destination IP address
</varlistentry>
daddrdaddr
A string representing the destination IP address
A string representing the destination IP address
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>sport
TCP source port
</varlistentry>
sportsport
TCP source port
TCP source port
<varlistentry>dport
TCP destination port
</varlistentry>
dportdport
TCP destination port
TCP destination port
<varlistentry>size
Number of bytes to be received
</varlistentry>
sisesize
Number of bytes to be received
Number of bytes to be received
<varlistentry>sock
Network socket
</varlistentry>
socksock
Network socket
Network socket
```

Context

The process which receives a tcp message

The process which receives a tcp message

Name

tcp.recvmsg.return -- Receiving TCP message complete

Synopsis

tcp.recvmsg.return

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes received or error code if an error occurred.

Context

The process which receives a tcp message

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.recvmsg.returntcp.recvmsg.return(3stap)

Name

tcp.recvmsg.return -- Receiving TCP message complete

tcp.recvmsg.return -- Receiving TCP message complete

Synopsis

tcp.recvmsg.return

tcp.recvmsg.return

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
--------------	---

Networking Tapset

<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes received or error code if an error occurred.
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>size</i>	<i>size</i> Number of bytes received or error code if an error occurred.

<varlistentry>*saddr*
A string representing the source IP address
</varlistentry>
saddrsaddr
A string representing the source IP address
A string representing the source IP address
<varlistentry>*daddr*
A string representing the destination IP address
</varlistentry>
daddr
A string representing the destination IP address
A string representing the destination IP address
<varlistentry>*name*
Name of this probe
</varlistentry>
name
Name of this probe
Name of this probe
<varlistentry>*sport*
TCP source port
</varlistentry>
sport
TCP source port
TCP source port
<varlistentry>*dport*
TCP destination port

```
</varlistentry>  
dportdport  
TCP destination port  
TCP destination port  
<varlistentry>size  
Number of bytes received or error code if an error occurred.  
</varlistentry>  
size  
Number of bytes received or error code if an error occurred.  
Number of bytes received or error code if an error occurred.
```

Context

The process which receives a tcp message

The process which receives a tcp message

Name

tcp.disconnect -- TCP socket disconnection

Synopsis

tcp.disconnect

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>flags</i>	<i>flags</i> TCP flags (e.g. FIN, etc)
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>sock</i>	<i>sock</i> Network socket

Context

The process which disconnects tcp
SystemTap Tapset Reference™
<date>February 2010</date>
tcp.disconnecttcp.disconnect(3stap)

Name

tcp.disconnect -- TCP socket disconnection

tcp.disconnect -- TCP socket disconnection

Synopsis

tcp.disconnect

tcp.disconnect

Values

<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>flags</i>	<i>flags</i> TCP flags (e.g. FIN, etc)
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>sock</i>	<i>sock</i> Network socket
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>flags</i>	<i>flags</i> TCP flags (e.g. FIN, etc)
<i>name</i>	<i>name</i> Name of this probe
<i>sport</i>	<i>sport</i> TCP source port
<i>dport</i>	<i>dport</i> TCP destination port
<i>sock</i>	<i>sock</i> Network socket

<varlistentry>saddr
A string representing the source IP address
</varlistentry>
saddrsaddr
A string representing the source IP address
A string representing the source IP address
<varlistentry>daddr
A string representing the destination IP address
</varlistentry>
daddrdaddr
A string representing the destination IP address
A string representing the destination IP address
<varlistentry>flags
TCP flags (e.g. FIN, etc)
</varlistentry>
flagsflags
TCP flags (e.g. FIN, etc)
TCP flags (e.g. FIN, etc)
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>sport
TCP source port
</varlistentry>
sportsport
TCP source port
TCP source port
<varlistentry>dport
TCP destination port
</varlistentry>
dportdport
TCP destination port
TCP destination port
<varlistentry>sock
Network socket
</varlistentry>
socksock
Network socket
Network socket

Context

The process which disconnects tcp

The process which disconnects tcp

Name

tcp.disconnect.return -- TCP socket disconnection complete

Synopsis

tcp.disconnect.return

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

Context

The process which disconnects tcp

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.disconnect.returntcp.disconnect.return(3stap)

Name

tcp.disconnect.return -- TCP socket disconnection complete

tcp.disconnect.return -- TCP socket disconnection complete

Synopsis

tcp.disconnect.return

tcp.disconnect.return

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

```
<varlistentry>ret
Error code (0: no error)
</varlistentry>
retret
Error code (0: no error)
Error code (0: no error)
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
```

Context

The process which disconnects tcp

The process which disconnects tcp

Name

tcp.setsockopt -- Call to setsockopt

Synopsis

tcp.setsockopt

Values

<i>optstr</i>	<i>optstr</i> Resolves optname to a human-readable format
<i>level</i>	<i>level</i> The level at which the socket options will be manipulated
<i>optlen</i>	<i>optlen</i> Used to access values for setsockopt
<i>name</i>	<i>name</i> Name of this probe
<i>optname</i>	<i>optname</i> TCP socket options (e.g. TCP_NODELAY, TCP_MAXSEG, etc)
<i>sock</i>	<i>sock</i> Network socket

Context

The process which calls setsockopt

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.setsockopttcp.setsockopt(3stap)

Name

tcp.setsockopt -- Call to setsockopt

tcp.setsockopt -- Call to setsockopt setsockopt

Synopsis

tcp.setsockopt

tcp.setsockopt

Values

<i>optstr</i>	<i>optstr</i> Resolves optname to a human-readable format
---------------	--

<i>level</i>	<i>level</i> The level at which the socket options will be manipulated
<i>optlen</i>	<i>optlen</i> Used to access values for <code>setsockopt</code>
<i>name</i>	<i>name</i> Name of this probe
<i>optname</i>	<i>optname</i> TCP socket options (e.g. <code>TCP_NODELAY</code> , <code>TCP_MAXSEG</code> , etc)
<i>sock</i>	<i>sock</i> Network socket
<i>optstr</i>	<i>optstr</i> Resolves <i>optname</i> to a human-readable format
<i>level</i>	<i>level</i> The level at which the socket options will be manipulated
<i>optlen</i>	<i>optlen</i> Used to access values for <code>setsockopt</code>
<i>name</i>	<i>name</i> Name of this probe
<i>optname</i>	<i>optname</i> TCP socket options (e.g. <code>TCP_NODELAY</code> , <code>TCP_MAXSEG</code> , etc)
<i>sock</i>	<i>sock</i> Network socket

<varlistentry>*optstr*
Resolves *optname* to a human-readable format
</varlistentry>
optstroptstr
Resolves *optname* to a human-readable format
Resolves *optname* to a human-readable format
<varlistentry>*level*
The level at which the socket options will be manipulated
</varlistentry>
levellevel
The level at which the socket options will be manipulated
The level at which the socket options will be manipulated
<varlistentry>*optlen*
Used to access values for `setsockopt`
</varlistentry>
optlenoptlen
Used to access values for `setsockopt`
Used to access values for `setsockopt`
`setsockopt`
<varlistentry>*name*
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>*optname*

TCP socket options (e.g. TCP_NODELAY, TCP_MAXSEG, etc)

</varlistentry>

optnameoptname

TCP socket options (e.g. TCP_NODELAY, TCP_MAXSEG, etc)

TCP socket options (e.g. TCP_NODELAY, TCP_MAXSEG, etc)

<varlistentry>*sock*

Network socket

</varlistentry>

socksock

Network socket

Network socket

Context

The process which calls setsockopt

The process which calls setsockopt

Name

tcp.setsockopt.return -- Return from setsockopt

Synopsis

tcp.setsockopt.return

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

Context

The process which calls setsockopt

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.setsockopt.returntcp.setsockopt.return(3stap)

Name

tcp.setsockopt.return -- Return from setsockopt

tcp.setsockopt.return -- Return from setsockopt setsockopt

Synopsis

tcp.setsockopt.return

tcp.setsockopt.return

Values

ret *ret*
Error code (0: no error)

name *name*
Name of this probe

ret *ret*
Error code (0: no error)

name *name*
Name of this probe


```
<varlistentry>ret
Error code (0: no error)
</varlistentry>
retret
Error code (0: no error)
Error code (0: no error)
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
```

Context

The process which calls setsockopt

The process which calls setsockopt

Name

tcp.receive -- Called when a TCP packet is received

Synopsis

tcp.receive

Values

<i>urg</i>	<i>urg</i> TCP URG flag
<i>psh</i>	<i>psh</i> TCP PSH flag
<i>rst</i>	<i>rst</i> TCP RST flag
<i>dport</i>	<i>dport</i> TCP destination port
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>ack</i>	<i>ack</i> TCP ACK flag
<i>syn</i>	<i>syn</i> TCP SYN flag
<i>fin</i>	<i>fin</i> TCP FIN flag
<i>sport</i>	<i>sport</i> TCP source port

SystemTap Tapset Reference™

<date>February 2010</date>

tcp.receivetcp.receive(3stap)

Name

tcp.receive -- Called when a TCP packet is received

tcp.receive -- Called when a TCP packet is received

Synopsis

tcp.receive

tcp.receive

Values

<i>urg</i>	<i>urg</i> TCP URG flag
<i>psh</i>	<i>psh</i> TCP PSH flag
<i>rst</i>	<i>rst</i> TCP RST flag
<i>dport</i>	<i>dport</i> TCP destination port
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>ack</i>	<i>ack</i> TCP ACK flag
<i>syn</i>	<i>syn</i> TCP SYN flag
<i>fin</i>	<i>fin</i> TCP FIN flag
<i>sport</i>	<i>sport</i> TCP source port
<i>urg</i>	<i>urg</i> TCP URG flag
<i>psh</i>	<i>psh</i> TCP PSH flag
<i>rst</i>	<i>rst</i> TCP RST flag
<i>dport</i>	<i>dport</i> TCP destination port
<i>saddr</i>	<i>saddr</i> A string representing the source IP address
<i>daddr</i>	<i>daddr</i> A string representing the destination IP address
<i>ack</i>	<i>ack</i> TCP ACK flag
<i>syn</i>	<i>syn</i> TCP SYN flag
<i>fin</i>	<i>fin</i> TCP FIN flag
<i>sport</i>	<i>sport</i> TCP source port

```
<varlistentry>urg
TCP URG flag
</varlistentry>
urgurg
TCP URG flag
TCP URG flag
<varlistentry>psh
TCP PSH flag
</varlistentry>
pshpsh
TCP PSH flag
TCP PSH flag
<varlistentry>rst
TCP RST flag
</varlistentry>
rstrst
TCP RST flag
TCP RST flag
<varlistentry>dport
TCP destination port
</varlistentry>
dportdport
TCP destination port
TCP destination port
<varlistentry>saddr
A string representing the source IP address
</varlistentry>
saddrsaddr
A string representing the source IP address
A string representing the source IP address
<varlistentry>daddr
A string representing the destination IP address
</varlistentry>
daddrdaddr
A string representing the destination IP address
A string representing the destination IP address
<varlistentry>ack
TCP ACK flag
</varlistentry>
ackack
TCP ACK flag
TCP ACK flag
<varlistentry>syn
TCP SYN flag
</varlistentry>
synsyn
TCP SYN flag
TCP SYN flag
<varlistentry>fin
TCP FIN flag
</varlistentry>
finfin
TCP FIN flag
TCP FIN flag
<varlistentry>sport
TCP source port
</varlistentry>
sportsport
```

TCP source port
TCP source port

Name

udp.sendmsg -- Fires whenever a process sends a UDP message

Synopsis

udp.sendmsg

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes sent by the process
<i>sock</i>	<i>sock</i>
	Network socket used by the process

Context

The process which sent a UDP message

SystemTap Tapset Reference™

<date>February 2010</date>

udp.sendmsgudp.sendmsg(3stap)

Name

udp.sendmsg -- Fires whenever a process sends a UDP message

udp.sendmsg -- Fires whenever a process sends a UDP message

Synopsis

udp.sendmsg

udp.sendmsg

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes sent by the process
<i>sock</i>	<i>sock</i>
	Network socket used by the process

<i>name</i>	<i>name</i>
	The name of this probe

size *size*
Number of bytes sent by the process

sock *sock*
Network socket used by the process

<varlistentry>*name*
The name of this probe
</varlistentry>

*name**name*
The name of this probe
The name of this probe

<varlistentry>*size*
Number of bytes sent by the process
</varlistentry>

*size**size*
Number of bytes sent by the process
Number of bytes sent by the process

<varlistentry>*sock*
Network socket used by the process
</varlistentry>

*sock**sock*
Network socket used by the process
Network socket used by the process

Context

The process which sent a UDP message

The process which sent a UDP message

Name

udp.sendmsg.return -- Fires whenever an attempt to send a UDP message is completed

Synopsis

udp.sendmsg.return

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes sent by the process

Context

The process which sent a UDP message
SystemTap Tapset Reference™
<date>February 2010</date>
udp.sendmsg.returnudp.sendmsg.return(3stap)

Name

udp.sendmsg.return -- Fires whenever an attempt to send a UDP message is completed

udp.sendmsg.return -- Fires whenever an attempt to send a UDP message is completed

Synopsis

udp.sendmsg.return

udp.sendmsg.return

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes sent by the process

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes sent by the process

`<varlistentry>name`

The name of this probe

`</varlistentry>`

*name**name*

The name of this probe

The name of this probe

`<varlistentry>size`

Number of bytes sent by the process

`</varlistentry>`

*size**size*

Number of bytes sent by the process

Number of bytes sent by the process

Context

The process which sent a UDP message

The process which sent a UDP message

Name

udp.recvmsg -- Fires whenever a UDP message is received

Synopsis

udp.recvmsg

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes received by the process
<i>sock</i>	<i>sock</i>
	Network socket used by the process

Context

The process which received a UDP message

SystemTap Tapset Reference™

<date>February 2010</date>

udp.recvmsgudp.recvmsg(3stap)

Name

udp.recvmsg -- Fires whenever a UDP message is received

udp.recvmsg -- Fires whenever a UDP message is received

Synopsis

udp.recvmsg

udp.recvmsg

Values

<i>name</i>	<i>name</i>
	The name of this probe
<i>size</i>	<i>size</i>
	Number of bytes received by the process
<i>sock</i>	<i>sock</i>
	Network socket used by the process

<i>name</i>	<i>name</i>
	The name of this probe

size *size*
Number of bytes received by the process

sock *sock*
Network socket used by the process

<varlistentry>*name*
The name of this probe
</varlistentry>

*name**name*
The name of this probe
The name of this probe

<varlistentry>*size*
Number of bytes received by the process
</varlistentry>

*size**size*
Number of bytes received by the process
Number of bytes received by the process

<varlistentry>*sock*
Network socket used by the process
</varlistentry>

*sock**sock*
Network socket used by the process
Network socket used by the process

Context

The process which received a UDP message

The process which received a UDP message

Name

udp.recvmsg.return -- Fires whenever an attempt to receive a UDP message received is completed

Synopsis

```
udp.recvmsg.return
```

Values

<i>name</i>	<i>name</i> The name of this probe
<i>size</i>	<i>size</i> Number of bytes received by the process

Context

The process which received a UDP message
SystemTap Tapset Reference™
<date>February 2010</date>
udp.recvmsg.returnudp.recvmsg.return(3stap)

Name

udp.recvmsg.return -- Fires whenever an attempt to receive a UDP message received is completed

udp.recvmsg.return -- Fires whenever an attempt to receive a UDP message received is completed

Synopsis

```
udp.recvmsg.return
```

```
udp.recvmsg.return
```

Values

<i>name</i>	<i>name</i> The name of this probe
<i>size</i>	<i>size</i> Number of bytes received by the process

<i>name</i>	<i>name</i> The name of this probe
<i>size</i>	<i>size</i> Number of bytes received by the process

<varlistentry>*name*

The name of this probe

</varlistentry>

*name**name*

The name of this probe

The name of this probe

<varlistentry>*size*

Number of bytes received by the process

</varlistentry>

*size**size*

Number of bytes received by the process

Number of bytes received by the process

Context

The process which received a UDP message

The process which received a UDP message

Name

udp.disconnect -- Fires when a process requests for a UDP disconnection

Synopsis

udp.disconnect

Values

<i>flags</i>	<i>flags</i> Flags (e.g. FIN, etc)
<i>name</i>	<i>name</i> The name of this probe
<i>sock</i>	<i>sock</i> Network socket used by the process

Context

The process which requests a UDP disconnection

SystemTap Tapset Reference™

<date>February 2010</date>

udp.disconnectudp.disconnect(3stap)

Name

udp.disconnect -- Fires when a process requests for a UDP disconnection

udp.disconnect -- Fires when a process requests for a UDP disconnection

Synopsis

udp.disconnect

udp.disconnect

Values

<i>flags</i>	<i>flags</i> Flags (e.g. FIN, etc)
<i>name</i>	<i>name</i> The name of this probe
<i>sock</i>	<i>sock</i> Network socket used by the process

<i>flags</i>	<i>flags</i> Flags (e.g. FIN, etc)
--------------	---------------------------------------

<i>name</i>	<i>name</i> The name of this probe
<i>sock</i>	<i>sock</i> Network socket used by the process

```
<varlistentry>flags
Flags (e.g. FIN, etc)
</varlistentry>
flagsflags
Flags (e.g. FIN, etc)
Flags (e.g. FIN, etc)
<varlistentry>name
The name of this probe
</varlistentry>
namenname
The name of this probe
The name of this probe
<varlistentry>sock
Network socket used by the process
</varlistentry>
socksock
Network socket used by the process
Network socket used by the process
```

Context

The process which requests a UDP disconnection

The process which requests a UDP disconnection

Name

udp.disconnect.return -- UDP has been disconnected successfully

Synopsis

udp.disconnect.return

Values

ret *ret*
Error code (0: no error)

name *name*
The name of this probe

Context

The process which requested a UDP disconnection

SystemTap Tapset Reference™

<date>February 2010</date>

udp.disconnect.returnudp.disconnect.return(3stap)

Name

udp.disconnect.return -- UDP has been disconnected successfully

udp.disconnect.return -- UDP has been disconnected successfully

Synopsis

udp.disconnect.return

udp.disconnect.return

Values

ret *ret*
Error code (0: no error)

name *name*
The name of this probe

ret *ret*
Error code (0: no error)

name *name*
The name of this probe


```
<varlistentry>ret
Error code (0: no error)
</varlistentry>
retret
Error code (0: no error)
Error code (0: no error)
<varlistentry>name
The name of this probe
</varlistentry>
namenname
The name of this probe
The name of this probe
```

Context

The process which requested a UDP disconnection

The process which requested a UDP disconnection

Name

`ip_ntop` -- returns a string representation from an integer IP number

Synopsis

```
ip_ntop:string(addr:long)
```

Arguments

addr *addr*
the ip represented as an integer

SystemTap Tapset Reference™

<date>February 2010</date>

`ip_ntop``ip_ntop(3stap)`

Name

`ip_ntop` -- returns a string representation from an integer IP number

`ip_ntop` -- returns a string representation from an integer IP number

Synopsis

```
ip_ntop:string(addr:long)
```

```
ip_ntop:string(addr:long)
```

Arguments

addr *addr*
the ip represented as an integer

addr *addr*
the ip represented as an integer

<varlistentry>*addr*
the ip represented as an integer
</varlistentry>

*addr**addr*
the ip represented as an integer
the ip represented as an integer

Chapter 8. Socket Tapset

This family of probe points is used to probe socket activities. It contains the following probe points:

Name

socket.send -- Message sent on a socket.

Synopsis

```
socket.send
```

Values

<i>success</i>	<i>success</i>
Was send successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message sent (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message sender

Name

socket.receive -- Message received on a socket.

Synopsis

```
socket.receive
```

Values

<i>success</i>	<i>success</i>
Was send successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message received (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver

Name

socket.sendmsg -- Message is currently being sent on a socket.

Synopsis

```
socket.sendmsg
```

Values

protocol *protocol*
Protocol value

flags *flags*
Socket flags value

name *name*
Name of this probe

state *state*
Socket state value

size *size*
Message size in bytes

type *type*
Socket type value

family *family*
Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the the `sock_sendmsg` function

Name

`socket.sendmsg.return` -- Return from `<command>socket.sendmsg</command>`.

Synopsis

```
socket.sendmsg.return
```

Values

<i>success</i>	<i>success</i>
Was send successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message sent (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message sender.

Description

Fires at the conclusion of sending a message on a socket via the `sock_sendmsg` function

Name

socket.recvmsg -- Message being received on socket

Synopsis

```
socket.recvmsg
```

Values

protocol *protocol*
Protocol value

flags *flags*
Socket flags value

name *name*
Name of this probe

state *state*
Socket state value

size *size*
Message size in bytes

type *type*
Socket type value

family *family*
Protocol family value

Context

The message receiver.

Description

Fires at the beginning of receiving a message on a socket via the `sock_recvmsg` function

Name

socket.recvmsg.return -- Return from Message being received on socket

Synopsis

```
socket.recvmsg.return
```

Values

<i>success</i>	<i>success</i>
Was receive successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message received (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_recvmsg` function.

Name

socket.aio_write -- Message send via sock_aio_write

Synopsis

```
socket.aio_write
```

Values

protocol *protocol*
Protocol value

flags *flags*
Socket flags value

name *name*
Name of this probe

state *state*
Socket state value

size *size*
Message size in bytes

type *type*
Socket type value

family *family*
Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the `sock_aio_write` function

Name

socket.aio_write.return -- Conclusion of message send via `sock_aio_write`

Synopsis

```
socket.aio_write.return
```

Values

<i>success</i>	<i>success</i>
Was receive successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message received (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the `sock_aio_write` function

Name

socket.aio_read -- Receiving message via sock_aio_read

Synopsis

```
socket.aio_read
```

Values

<i>protocol</i>	<i>protocol</i>
Protocol value	

<i>flags</i>	<i>flags</i>
Socket flags value	

<i>name</i>	<i>name</i>
Name of this probe	

<i>state</i>	<i>state</i>
Socket state value	

<i>size</i>	<i>size</i>
Message size in bytes	

<i>type</i>	<i>type</i>
Socket type value	

<i>family</i>	<i>family</i>
Protocol family value	

Context

The message sender

Description

Fires at the beginning of receiving a message on a socket via the `sock_aio_read` function

Name

socket.aio_read.return -- Conclusion of message received via `sock_aio_read`

Synopsis

```
socket.aio_read.return
```

Values

<i>success</i>	<i>success</i>
Was receive successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message received (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_aio_read` function

Name

socket.writev -- Message sent via socket_writev

Synopsis

```
socket.writev
```

Values

protocol *protocol*
Protocol value

flags *flags*
Socket flags value

name *name*
Name of this probe

state *state*
Socket state value

size *size*
Message size in bytes

type *type*
Socket type value

family *family*
Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the `sock_writev` function

Name

`socket.writev.return` -- Conclusion of message sent via `socket.writev`

Synopsis

`socket.writev.return`

Values

<i>success</i>	<i>success</i>
Was send successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message sent (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the `sock_writev` function

Name

socket.readv -- Receiving a message via `sock_readv`

Synopsis

```
socket.readv
```

Values

protocol *protocol*
Protocol value

flags *flags*
Socket flags value

name *name*
Name of this probe

state *state*
Socket state value

size *size*
Message size in bytes

type *type*
Socket type value

family *family*
Protocol family value

Context

The message sender

Description

Fires at the beginning of receiving a message on a socket via the `sock_readv` function

Name

socket.readv.return -- Conclusion of receiving a message via `sock_readv`

Synopsis

```
socket.readv.return
```

Values

<i>success</i>	<i>success</i>
Was receive successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>flags</i>	<i>flags</i>
Socket flags value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>state</i>	<i>state</i>
Socket state value	
<i>size</i>	<i>size</i>
Size of message received (in bytes) or error code if success = 0	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_readv` function

Name

socket.create -- Creation of a socket

Synopsis

```
socket.create
```

Values

<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>name</i>	<i>name</i>
Name of this probe	
<i>requester</i>	<i>requester</i>
Requested by user process or the kernel (1 = kernel, 0 = user)	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The requester (see requester variable)

Description

Fires at the beginning of creating a socket.

Name

`socket.create.return` -- Return from Creation of a socket

Synopsis

`socket.create.return`

Values

<i>success</i>	<i>success</i>
Was socket creation successful? (1 = yes, 0 = no)	
<i>protocol</i>	<i>protocol</i>
Protocol value	
<i>err</i>	<i>err</i>
Error code if success == 0	
<i>name</i>	<i>name</i>
Name of this probe	
<i>requester</i>	<i>requester</i>
Requested by user process or the kernel (1 = kernel, 0 = user)	
<i>type</i>	<i>type</i>
Socket type value	
<i>family</i>	<i>family</i>
Protocol family value	

Context

The requester (user process or kernel)

Description

Fires at the conclusion of creating a socket.

Name

socket.close -- Close a socket

Synopsis

```
socket.close
```

Values

<i>protocol</i>	<i>protocol</i>
Protocol value	

<i>flags</i>	<i>flags</i>
Socket flags value	

<i>name</i>	<i>name</i>
Name of this probe	

<i>state</i>	<i>state</i>
Socket state value	

<i>type</i>	<i>type</i>
Socket type value	

<i>family</i>	<i>family</i>
Protocol family value	

Context

The requester (user process or kernel)

Description

Fires at the beginning of closing a socket.

Name

socket.close.return -- Return from closing a socket

Synopsis

```
socket.close.return
```

Values

name *name*
Name of this probe

Context

The requester (user process or kernel)

Description

Fires at the conclusion of closing a socket.

Name

`sock_prot_num2str` -- Given a protocol number, return a string representation.

Synopsis

```
sock_prot_num2str:string(proto:long)
```

Arguments

proto *proto*
The protocol number.

Name

`sock_prot_str2num` -- Given a protocol name (string), return the corresponding protocol number.

Synopsis

```
sock_prot_str2num:long(proto:string)
```

Arguments

proto *proto*
The protocol name.

Name

`sock_fam_num2str` -- Given a protocol family number, return a string representation.

Synopsis

```
sock_fam_num2str:string(family:long)
```

Arguments

family *family*
The family number.

Name

`sock_fam_str2num` -- Given a protocol family name (string), return the corresponding

Synopsis

```
sock_fam_str2num:long(family:string)
```

Arguments

family *family*
The family name.

Description

protocol family number.

Name

`sock_state_num2str` -- Given a socket state number, return a string representation.

Synopsis

```
sock_state_num2str:string(state:long)
```

Arguments

state *state*
The state number.

Name

`sock_state_str2num` -- Given a socket state string, return the corresponding state number.

Synopsis

```
sock_state_str2num:long(state:string)
```

Arguments

state *state*
The state name.

This family of probe points is used to probe socket activities. It contains the following probe points:

Name

socket.send -- Message sent on a socket.

Synopsis

```
socket.send
```

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender
SystemTap Tapset Reference™
<date>February 2010</date>
socket.sendsocket.send(3stap)

Name

socket.send -- Message sent on a socket.

socket.send -- Message sent on a socket.

Synopsis

```
socket.send
```

```
socket.send
```

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>success
Was send successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was send successful? (1 = yes, 0 = no)
Was send successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message sent (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message sent (in bytes) or error code if success = 0
Size of message sent (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Name

socket.receive -- Message received on a socket.

Synopsis

socket.receive

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver

SystemTap Tapset Reference™

<date>February 2010</date>

socket.receivesocket.receive(3stap)

Name

socket.receive -- Message received on a socket.

socket.receive -- Message received on a socket.

Synopsis

socket.receive

socket.receive

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>success
Was send successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was send successful? (1 = yes, 0 = no)
Was send successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message received (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message received (in bytes) or error code if success = 0
Size of message received (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message receiver

The message receiver

Name

socket.sendmsg -- Message is currently being sent on a socket.

Synopsis

socket.sendmsg

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the the sock_sendmsg function
SystemTap Tapset Reference™

<date>February 2010</date>

socket.sendmsgsocket.sendmsg(3stap)

Name

socket.sendmsg -- Message is currently being sent on a socket.

socket.sendmsg -- Message is currently being sent on a socket.

Synopsis

socket.sendmsg

socket.sendmsg

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Description

Fires at the beginning of sending a message on a socket via the the `sock_sendmsg` function

Fires at the beginning of sending a message on a socket via the the `sock_sendmsg` function
`sock_sendmsg`

Name

socket.sendmsg.return -- Return from <command>socket.sendmsg</command>.

Synopsis

```
socket.sendmsg.return
```

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender.

Description

Fires at the conclusion of sending a message on a socket via the `sock_sendmsg` function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.sendmsg.returnssocket.sendmsg.return(3stap)

Name

socket.sendmsg.return -- Return from <command>socket.sendmsg</command>.

socket.sendmsg.return -- Return from <command>socket.sendmsg</command>.

Synopsis

```
socket.sendmsg.return
```

`socket.sendmsg.return`

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>success
Was send successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was send successful? (1 = yes, 0 = no)
Was send successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message sent (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message sent (in bytes) or error code if success = 0
Size of message sent (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender.

The message sender.

Description

Fires at the conclusion of sending a message on a socket via the `sock_sendmsg` function

Fires at the conclusion of sending a message on a socket via the `sock_sendmsg` function
`sock_sendmsg`

Name

socket.recvmsg -- Message being received on socket

Synopsis

socket.recvmsg

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the beginning of receiving a message on a socket via the `sock_recvmsg` function
SystemTap Tapset Reference™

<date>February 2010</date>

socket.recvmsgsocket.recvmsg(3stap)

Name

socket.recvmsg -- Message being received on socket

socket.recvmsg -- Message being received on socket

Synopsis

socket.recvmsg

socket.recvmsg

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message receiver.

The message receiver.

Description

Fires at the beginning of receiving a message on a socket via the `sock_recvmsg` function

Fires at the beginning of receiving a message on a socket via the `sock_recvmsg` function
`sock_recvmsg`

Name

socket.recvmsg.return -- Return from Message being received on socket

Synopsis

```
socket.recvmsg.return
```

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_recvmsg` function.

SystemTap Tapset Reference™

<date>February 2010</date>

socket.recvmsg.returnssocket.recvmsg.return(3stap)

Name

socket.recvmsg.return -- Return from Message being received on socket

socket.recvmsg.return -- Return from Message being received on socket

Synopsis

```
socket.recvmsg.return
```

`socket.recvmsg.return`

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>success
Was receive successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was receive successful? (1 = yes, 0 = no)
Was receive successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message received (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message received (in bytes) or error code if success = 0
Size of message received (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The message receiver.

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_recvmsg` function.

Fires at the conclusion of receiving a message on a socket via the `sock_recvmsg` function.
`sock_recvmsg`

Name

socket.aio_write -- Message send via sock_aio_write

Synopsis

socket.aio_write

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the sock_aio_write function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.aio_writesocket.aio_write(3stap)

Name

socket.aio_write -- Message send via sock_aio_write

socket.aio_write -- Message send via sock_aio_write sock_aio_write

Synopsis

socket.aio_write

socket.aio_write

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesizesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Description

Fires at the beginning of sending a message on a socket via the `sock_aio_write` function

Fires at the beginning of sending a message on a socket via the `sock_aio_write` function
`sock_aio_write`

Name

socket.aio_write.return -- Conclusion of message send via sock_aio_write

Synopsis

```
socket.aio_write.return
```

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the sock_aio_write function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.aio_write.returnssocket.aio_write.return(3stap)

Name

socket.aio_write.return -- Conclusion of message send via sock_aio_write

socket.aio_write.return -- Conclusion of message send via sock_aio_write sock_aio_write

Synopsis

```
socket.aio_write.return
```

```
socket.aio_write.return
```

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>success
Was receive successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was receive successful? (1 = yes, 0 = no)
Was receive successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message received (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message received (in bytes) or error code if success = 0
Size of message received (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The message receiver.

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the `sock_aio_write` function

Fires at the conclusion of sending a message on a socket via the `sock_aio_write` function
`sock_aio_write`

Name

socket.aio_read -- Receiving message via sock_aio_read

Synopsis

socket.aio_read

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender

Description

Fires at the beginning of receiving a message on a socket via the sock_aio_read function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.aio_readsocket.aio_read(3stap)

Name

socket.aio_read -- Receiving message via sock_aio_read

socket.aio_read -- Receiving message via sock_aio_read sock_aio_read

Synopsis

socket.aio_read

socket.aio_read

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Description

Fires at the beginning of receiving a message on a socket via the `sock_aio_read` function

Fires at the beginning of receiving a message on a socket via the `sock_aio_read` function
`sock_aio_read`

Name

socket.aio_read.return -- Conclusion of message received via sock_aio_read

Synopsis

socket.aio_read.return

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the sock_aio_read function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.aio_read.returnssocket.aio_read.return(3stap)

Name

socket.aio_read.return -- Conclusion of message received via sock_aio_read

socket.aio_read.return -- Conclusion of message received via sock_aio_read sock_aio_read

Synopsis

socket.aio_read.return

```
socket.aio_read.return
```

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>success
Was receive successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was receive successful? (1 = yes, 0 = no)
Was receive successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message received (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message received (in bytes) or error code if success = 0
Size of message received (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The message receiver.

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_aio_read` function

Fires at the conclusion of receiving a message on a socket via the `sock_aio_read` function
`sock_aio_read`

Name

socket.writev -- Message sent via socket_writev

Synopsis

socket.writev

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender

Description

Fires at the beginning of sending a message on a socket via the sock_writev function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.writevsocket.writev(3stap)

Name

socket.writev -- Message sent via socket_writev

socket.writev -- Message sent via socket_writev socket_writev

Synopsis

socket.writev

socket.writev

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value


```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Description

Fires at the beginning of sending a message on a socket via the `sock_writev` function

Fires at the beginning of sending a message on a socket via the `sock_writev` function
`sock_writev`

Name

socket.writev.return -- Conclusion of message sent via socket_writev

Synopsis

socket.writev.return

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the sock_writev function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.writev.returnssocket.writev.return(3stap)

Name

socket.writev.return -- Conclusion of message sent via socket_writev

socket.writev.return -- Conclusion of message sent via socket_writev socket_writev

Synopsis

socket.writev.return

`socket.writev.return`

Values

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was send successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message sent (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>success
Was send successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was send successful? (1 = yes, 0 = no)
Was send successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message sent (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message sent (in bytes) or error code if success = 0
Size of message sent (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message receiver.

The message receiver.

Description

Fires at the conclusion of sending a message on a socket via the `sock_writev` function

Fires at the conclusion of sending a message on a socket via the `sock_writev` function
`sock_writev`

Name

socket.readv -- Receiving a message via `sock_readv`

Synopsis

`socket.readv`

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message sender

Description

Fires at the beginning of receiving a message on a socket via the `sock_readv` function
SystemTap Tapset Reference™

<date>February 2010</date>

socket.readvsocket.readv(3stap)

Name

socket.readv -- Receiving a message via `sock_readv`

socket.readv -- Receiving a message via `sock_readv` `sock_readv`

Synopsis

`socket.readv`

`socket.readv`

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Message size in bytes
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenamename
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Message size in bytes
</varlistentry>
sisesize
Message size in bytes
Message size in bytes
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value
```

Context

The message sender

The message sender

Description

Fires at the beginning of receiving a message on a socket via the `sock_readv` function

Fires at the beginning of receiving a message on a socket via the `sock_readv` function
`sock_readv`

Name

socket.readv.return -- Conclusion of receiving a message via sock_readv

Synopsis

socket.readv.return

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the sock_readv function

SystemTap Tapset Reference™

<date>February 2010</date>

socket.readv.returnssocket.readv.return(3stap)

Name

socket.readv.return -- Conclusion of receiving a message via sock_readv

socket.readv.return -- Conclusion of receiving a message via sock_readv sock_readv

Synopsis

socket.readv.return

`socket.readv.return`

Values

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>success</i>	<i>success</i> Was receive successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>size</i>	<i>size</i> Size of message received (in bytes) or error code if success = 0
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>success
Was receive successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was receive successful? (1 = yes, 0 = no)
Was receive successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>flags
Socket flags value
</varlistentry>
flagsflags
Socket flags value
Socket flags value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>state
Socket state value
</varlistentry>
statestate
Socket state value
Socket state value
<varlistentry>size
Size of message received (in bytes) or error code if success = 0
</varlistentry>
sisesize
Size of message received (in bytes) or error code if success = 0
Size of message received (in bytes) or error code if success = 0
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The message receiver.

The message receiver.

Description

Fires at the conclusion of receiving a message on a socket via the `sock_readv` function

Fires at the conclusion of receiving a message on a socket via the `sock_readv` function
`sock_readv`

Name

socket.create -- Creation of a socket

Synopsis

socket.create

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The requester (see requester variable)

Description

Fires at the beginning of creating a socket.

SystemTap Tapset Reference™

<date>February 2010</date>

socket.createsocket.create(3stap)

Name

socket.create -- Creation of a socket

socket.create -- Creation of a socket

Synopsis

socket.create

socket.create

Values

<i>protocol</i>	<i>protocol</i> Protocol value
-----------------	-----------------------------------

<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>protocol</i>	<i>protocol</i> Protocol value
<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>requester
Requested by user process or the kernel (1 = kernel, 0 = user)
</varlistentry>
requesterrequester
Requested by user process or the kernel (1 = kernel, 0 = user)
Requested by user process or the kernel (1 = kernel, 0 = user)
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The requester (see requester variable)

The requester (see requester variable)

Description

Fires at the beginning of creating a socket.

Fires at the beginning of creating a socket.

Name

socket.create.return -- Return from Creation of a socket

Synopsis

socket.create.return

Values

<i>success</i>	<i>success</i> Was socket creation successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>err</i>	<i>err</i> Error code if success == 0
<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The requester (user process or kernel)

Description

Fires at the conclusion of creating a socket.

SystemTap Tapset Reference™

<date>February 2010</date>

socket.create.returnsocket.create.return(3stap)

Name

socket.create.return -- Return from Creation of a socket

socket.create.return -- Return from Creation of a socket

Synopsis

socket.create.return

socket.create.return

Values

<i>success</i>	<i>success</i> Was socket creation successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>err</i>	<i>err</i> Error code if success == 0
<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value
<i>success</i>	<i>success</i> Was socket creation successful? (1 = yes, 0 = no)
<i>protocol</i>	<i>protocol</i> Protocol value
<i>err</i>	<i>err</i> Error code if success == 0
<i>name</i>	<i>name</i> Name of this probe
<i>requester</i>	<i>requester</i> Requested by user process or the kernel (1 = kernel, 0 = user)
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

```

<varlistentry>success
Was socket creation successful? (1 = yes, 0 = no)
</varlistentry>
successsuccess
Was socket creation successful? (1 = yes, 0 = no)
Was socket creation successful? (1 = yes, 0 = no)
<varlistentry>protocol
Protocol value
</varlistentry>
protocolprotocol
Protocol value
Protocol value
<varlistentry>err
Error code if success == 0
</varlistentry>
errerr
Error code if success == 0
Error code if success == 0
<varlistentry>name
Name of this probe
</varlistentry>
namenname
Name of this probe
Name of this probe
<varlistentry>requester
Requested by user process or the kernel (1 = kernel, 0 = user)
</varlistentry>
requesterrequester
Requested by user process or the kernel (1 = kernel, 0 = user)
Requested by user process or the kernel (1 = kernel, 0 = user)
<varlistentry>type
Socket type value
</varlistentry>
typetype
Socket type value
Socket type value
<varlistentry>family
Protocol family value
</varlistentry>
familyfamily
Protocol family value
Protocol family value

```

Context

The requester (user process or kernel)

The requester (user process or kernel)

Description

Fires at the conclusion of creating a socket.

Fires at the conclusion of creating a socket.

Name

socket.close -- Close a socket

Synopsis

socket.close

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

Context

The requester (user process or kernel)

Description

Fires at the beginning of closing a socket.

SystemTap Tapset Reference™

<date>February 2010</date>

socket.closesocket.close(3stap)

Name

socket.close -- Close a socket

socket.close -- Close a socket

Synopsis

socket.close

socket.close

Values

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<i>protocol</i>	<i>protocol</i> Protocol value
<i>flags</i>	<i>flags</i> Socket flags value
<i>name</i>	<i>name</i> Name of this probe
<i>state</i>	<i>state</i> Socket state value
<i>type</i>	<i>type</i> Socket type value
<i>family</i>	<i>family</i> Protocol family value

<varlistentry>*protocol*

Protocol value

</varlistentry>

protocolprotocol

Protocol value

Protocol value

<varlistentry>*flags*

Socket flags value

</varlistentry>

flagsflags

Socket flags value

Socket flags value

<varlistentry>*name*

Name of this probe

</varlistentry>

namenamename

Name of this probe

Name of this probe

<varlistentry>*state*

Socket state value

</varlistentry>

statestate

Socket state value

Socket state value

<varlistentry>type

Socket type value

</varlistentry>

*type**type*

Socket type value

Socket type value

<varlistentry>family

Protocol family value

</varlistentry>

*family**family*

Protocol family value

Protocol family value

Context

The requester (user process or kernel)

The requester (user process or kernel)

Description

Fires at the beginning of closing a socket.

Fires at the beginning of closing a socket.

Name

socket.close.return -- Return from closing a socket

Synopsis

socket.close.return

Values

name *name*
 Name of this probe

Context

The requester (user process or kernel)

Description

Fires at the conclusion of closing a socket.

SystemTap Tapset Reference™

<date>February 2010</date>

socket.close.returnssocket.close.return(3stap)

Name

socket.close.return -- Return from closing a socket

socket.close.return -- Return from closing a socket

Synopsis

socket.close.return

socket.close.return

Values

name *name*
 Name of this probe

name *name*
 Name of this probe

```
<varlistentry>name  
Name of this probe  
</varlistentry>  
namename  
Name of this probe  
Name of this probe
```

Context

The requester (user process or kernel)

The requester (user process or kernel)

Description

Fires at the conclusion of closing a socket.

Fires at the conclusion of closing a socket.

Name

sock_prot_num2str -- Given a protocol number, return a string representation.

Synopsis

```
sock_prot_num2str:string(proto:long)
```

Arguments

proto *proto*
 The protocol number.

SystemTap Tapset Reference™

<date>February 2010</date>

sock_prot_num2strsock_prot_num2str(3stap)

Name

sock_prot_num2str -- Given a protocol number, return a string representation.

sock_prot_num2str -- Given a protocol number, return a string representation.

Synopsis

```
sock_prot_num2str:string(proto:long)
```

```
sock_prot_num2str:string(proto:long)
```

Arguments

proto *proto*
 The protocol number.

proto *proto*
 The protocol number.

<varlistentry>*proto*
The protocol number.
</varlistentry>

*proto**proto*
The protocol number.
The protocol number.

Name

sock_prot_str2num -- Given a protocol name (string), return the corresponding protocol number.

Synopsis

```
sock_prot_str2num:long(proto:string)
```

Arguments

proto *proto*
 The protocol name.

SystemTap Tapset Reference™

<date>February 2010</date>

sock_prot_str2numsock_prot_str2num(3stap)

Name

sock_prot_str2num -- Given a protocol name (string), return the corresponding protocol number.

sock_prot_str2num -- Given a protocol name (string), return the corresponding protocol number.

Synopsis

```
sock_prot_str2num:long(proto:string)
```

```
sock_prot_str2num:long(proto:string)
```

Arguments

proto *proto*
 The protocol name.

proto *proto*
 The protocol name.

<varlistentry>*proto*

The protocol name.

</varlistentry>

*proto**proto*

The protocol name.

The protocol name.

Name

sock_fam_num2str -- Given a protocol family number, return a string representation.

Synopsis

```
sock_fam_num2str:string(family:long)
```

Arguments

<i>family</i>	<i>family</i>
	The family number.

SystemTap Tapset Reference™

<date>February 2010</date>

sock_fam_num2strsock_fam_num2str(3stap)

Name

sock_fam_num2str -- Given a protocol family number, return a string representation.

sock_fam_num2str -- Given a protocol family number, return a string representation.

Synopsis

```
sock_fam_num2str:string(family:long)
```

```
sock_fam_num2str:string(family:long)
```

Arguments

<i>family</i>	<i>family</i>
	The family number.

<i>family</i>	<i>family</i>
	The family number.

<varlistentry>*family*

The family number.

</varlistentry>

familyfamily

The family number.

The family number.

Name

sock_fam_str2num -- Given a protocol family name (string), return the corresponding

Synopsis

```
sock_fam_str2num:long(family:string)
```

Arguments

<i>family</i>	<i>family</i>
	The family name.

Description

protocol family number.

SystemTap Tapset Reference™

<date>February 2010</date>

sock_fam_str2numsock_fam_str2num(3stap)

Name

sock_fam_str2num -- Given a protocol family name (string), return the corresponding

sock_fam_str2num -- Given a protocol family name (string), return the corresponding

Synopsis

```
sock_fam_str2num:long(family:string)
```

```
sock_fam_str2num:long(family:string)
```

Arguments

<i>family</i>	<i>family</i>
	The family name.

<i>family</i>	<i>family</i>
	The family name.

<varlistentry>*family*

The family name.

</varlistentry>

familyfamily

The family name.

The family name.

Description

protocol family number.

protocol family number.

Name

sock_state_num2str -- Given a socket state number, return a string representation.

Synopsis

```
sock_state_num2str:string(state:long)
```

Arguments

state *state*
The state number.

SystemTap Tapset Reference™

<date>February 2010</date>

sock_state_num2strsock_state_num2str(3stap)

Name

sock_state_num2str -- Given a socket state number, return a string representation.

sock_state_num2str -- Given a socket state number, return a string representation.

Synopsis

```
sock_state_num2str:string(state:long)
```

```
sock_state_num2str:string(state:long)
```

Arguments

state *state*
The state number.

state *state*
The state number.

<varlistentry>*state*

The state number.

</varlistentry>

statestate

The state number.

The state number.

Name

sock_state_str2num -- Given a socket state string, return the corresponding state number.

Synopsis

```
sock_state_str2num:long(state:string)
```

Arguments

```
state      state
           The state name.
```

SystemTap Tapset Reference™

<date>February 2010</date>

sock_state_str2numsock_state_str2num(3stap)

Name

sock_state_str2num -- Given a socket state string, return the corresponding state number.

sock_state_str2num -- Given a socket state string, return the corresponding state number.

Synopsis

```
sock_state_str2num:long(state:string)
```

```
sock_state_str2num:long(state:string)
```

Arguments

```
state      state
           The state name.
```

```
state      state
           The state name.
```

<varlistentry>state

The state name.

</varlistentry>

```
statestate
```

The state name.

The state name.

Chapter 9. Kernel Process Tapset

This family of probe points is used to probe process-related activities. It contains the following probe points:

Name

kprocess.create -- Fires whenever a new process is successfully created

Synopsis

```
kprocess.create
```

Values

```
new_pid    new_pid
```

The PID of the newly created process

Context

Parent of the created process.

Description

Fires whenever a new process is successfully created, either as a result of <command>fork</command> (or one of its syscall variants), or a new kernel thread.

Name

kprocess.start -- Starting new process

Synopsis

```
kprocess.start
```

Values

None

Context

Newly created process.

Description

Fires immediately before a new process begins execution.

Name

kprocess.exec -- Attempt to exec to a new program

Synopsis

kprocess.exec

Values

filename *filename*
The path to the new executable

Context

The caller of exec.

Description

Fires whenever a process attempts to exec to a new program.

Name

kprocess.exec_complete -- Return from exec to a new program

Synopsis

```
kprocess.exec_complete
```

Values

success *success*

A boolean indicating whether the exec was successful

errno *errno*

The error number resulting from the exec

Context

On success, the context of the new executable. On failure, remains in the context of the caller.

Description

Fires at the completion of an exec call.

Name

kprocess.exit -- Exit from process

Synopsis

```
kprocess.exit
```

Values

code *code*

The exit code of the process

Context

The process which is terminating.

Description

Fires when a process terminates. This will always be followed by a kprocess.release, though the latter may be delayed if the process waits in a zombie state.

Name

kprocess.release -- Process released

Synopsis

```
kprocess.release
```

Values

```
pid    pid
PID of the process being released
```

```
task    task
A task handle to the process being released
```

Context

The context of the parent, if it wanted notification of this process' termination, else the context of the process itself.

Description

Fires when a process is released from the kernel. This always follows a kprocess.exit, though it may be delayed somewhat if the process waits in a zombie state.

This family of probe points is used to probe process-related activities. It contains the following probe points:

Name

kprocess.create -- Fires whenever a new process is successfully created

Synopsis

kprocess.create

Values

<i>new_pid</i>	<i>new_pid</i>
	The PID of the newly created process

Context

Parent of the created process.

Description

Fires whenever a new process is successfully created, either as a result of `<command>fork</command>` (or one of its syscall variants), or a new kernel thread.

SystemTap Tapset Reference™

<date>February 2010</date>

kprocess.createkprocess.create(3stap)

Name

kprocess.create -- Fires whenever a new process is successfully created

kprocess.create -- Fires whenever a new process is successfully created

Synopsis

kprocess.create

kprocess.create

Values

<i>new_pid</i>	<i>new_pid</i>
	The PID of the newly created process

<i>new_pid</i>	<i>new_pid</i>
	The PID of the newly created process

<varlistentry>*new_pid*

The PID of the newly created process

</varlistentry>

new_pidnew_pid

The PID of the newly created process

The PID of the newly created process

Context

Parent of the created process.

Parent of the created process.

Description

Fires whenever a new process is successfully created, either as a result of <command>fork</command> (or one of its syscall variants), or a new kernel thread.

Fires whenever a new process is successfully created, either as a result of <command>fork</command> (or one of its syscall variants), or a new kernel thread.

Name

kprocess.start -- Starting new process

Synopsis

```
kprocess.start
```

Values

None

Context

Newly created process.

Description

Fires immediately before a new process begins execution.

SystemTap Tapset Reference™

<date>February 2010</date>

```
kprocess.startkprocess.start(3stap)
```

Name

kprocess.start -- Starting new process

kprocess.start -- Starting new process

Synopsis

```
kprocess.start
```

```
kprocess.start
```

Values

None

None

Context

Newly created process.

Newly created process.

Description

Fires immediately before a new process begins execution.

Fires immediately before a new process begins execution.

Name

kprocess.exec -- Attempt to exec to a new program

Synopsis

kprocess.exec

Values

<i>filename</i>	<i>filename</i>
	The path to the new executable

Context

The caller of exec.

Description

Fires whenever a process attempts to exec to a new program.

SystemTap Tapset Reference™

<date>February 2010</date>

kprocess.execkprocess.exec(3stap)

Name

kprocess.exec -- Attempt to exec to a new program

kprocess.exec -- Attempt to exec to a new program

Synopsis

kprocess.exec

kprocess.exec

Values

<i>filename</i>	<i>filename</i>
	The path to the new executable

<i>filename</i>	<i>filename</i>
	The path to the new executable

```
<varlistentry>filename  
The path to the new executable  
</varlistentry>  
filenamefilename  
The path to the new executable  
The path to the new executable
```

Context

The caller of exec.

The caller of exec.

Description

Fires whenever a process attempts to exec to a new program.

Fires whenever a process attempts to exec to a new program.

Name

kprocess.exec_complete -- Return from exec to a new program

Synopsis

kprocess.exec_complete

Values

<i>success</i>	<i>success</i> A boolean indicating whether the exec was successful
<i>errno</i>	<i>errno</i> The error number resulting from the exec

Context

On success, the context of the new executable. On failure, remains in the context of the caller.

Description

Fires at the completion of an exec call.

SystemTap Tapset Reference™

<date>February 2010</date>

kprocess.exec_completekprocess.exec_complete(3stap)

Name

kprocess.exec_complete -- Return from exec to a new program

kprocess.exec_complete -- Return from exec to a new program

Synopsis

kprocess.exec_complete

kprocess.exec_complete

Values

<i>success</i>	<i>success</i> A boolean indicating whether the exec was successful
<i>errno</i>	<i>errno</i> The error number resulting from the exec
<i>success</i>	<i>success</i> A boolean indicating whether the exec was successful

errno *errno*
The error number resulting from the exec

<varlistentry>*success*
A boolean indicating whether the exec was successful
</varlistentry>

successsuccess
A boolean indicating whether the exec was successful
A boolean indicating whether the exec was successful

<varlistentry>*errno*
The error number resulting from the exec
</varlistentry>

errnoerrno
The error number resulting from the exec
The error number resulting from the exec

Context

On success, the context of the new executable. On failure, remains in the context of the caller.

On success, the context of the new executable. On failure, remains in the context of the caller.

Description

Fires at the completion of an exec call.

Fires at the completion of an exec call.

Name

kprocess.exit -- Exit from process

Synopsis

```
kprocess.exit
```

Values

code *code*
The exit code of the process

Context

The process which is terminating.

Description

Fires when a process terminates. This will always be followed by a kprocess.release, though the latter may be delayed if the process waits in a zombie state.

SystemTap Tapset Reference™

<date>February 2010</date>

kprocess.exitekprocess.exit(3stap)

Name

kprocess.exit -- Exit from process

kprocess.exit -- Exit from process

Synopsis

```
kprocess.exit
```

```
kprocess.exit
```

Values

code *code*
The exit code of the process

code *code*
The exit code of the process

<varlistentry>*code*

The exit code of the process

</varlistentry>

*code**code*

The exit code of the process

The exit code of the process

Context

The process which is terminating.

The process which is terminating.

Description

Fires when a process terminates. This will always be followed by a `kprocess.release`, though the latter may be delayed if the process waits in a zombie state.

Fires when a process terminates. This will always be followed by a `kprocess.release`, though the latter may be delayed if the process waits in a zombie state.

Name

kprocess.release -- Process released

Synopsis

kprocess.release

Values

pid *pid*
 PID of the process being released

task *task*
 A task handle to the process being released

Context

The context of the parent, if it wanted notification of this process' termination, else the context of the process itself.

Description

Fires when a process is released from the kernel. This always follows a kprocess.exit, though it may be delayed somewhat if the process waits in a zombie state.

SystemTap Tapset Reference™

<date>February 2010</date>

kprocess.releasekprocess.release(3stap)

Name

kprocess.release -- Process released

kprocess.release -- Process released

Synopsis

kprocess.release

kprocess.release

Values

pid *pid*
 PID of the process being released

task *task*
 A task handle to the process being released

pid *pid*
PID of the process being released

task *task*
A task handle to the process being released

<varlistentry>*pid*
PID of the process being released
</varlistentry>

pidpid
PID of the process being released
PID of the process being released
<varlistentry>*task*
A task handle to the process being released
</varlistentry>

tasktask
A task handle to the process being released
A task handle to the process being released

Context

The context of the parent, if it wanted notification of this process' termination, else the context of the process itself.

The context of the parent, if it wanted notification of this process' termination, else the context of the process itself.

Description

Fires when a process is released from the kernel. This always follows a `kprocess.exit`, though it may be delayed somewhat if the process waits in a zombie state.

Fires when a process is released from the kernel. This always follows a `kprocess.exit`, though it may be delayed somewhat if the process waits in a zombie state.

Chapter 10. Signal Tapset

This family of probe points is used to probe signal activities. It contains the following probe points:

Name

signal.send -- Signal being sent to a process

Synopsis

```
signal.send
```

Values

<i>send2queue</i>	<i>send2queue</i>	Indicates whether the signal is sent to an existing <command>sigqueue</command>
<i>name</i>	<i>name</i>	The name of the function used to send out the signal
<i>task</i>	<i>task</i>	A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i>	The address of <command>sinfo</command> struct
<i>si_code</i>	<i>si_code</i>	Indicates the signal type
<i>sig_name</i>	<i>sig_name</i>	A string representation of the signal
<i>sig</i>	<i>sig</i>	The number of the signal
<i>shared</i>	<i>shared</i>	Indicates whether the signal is shared by the thread group
<i>sig_pid</i>	<i>sig_pid</i>	The PID of the process receiving the signal
<i>pid_name</i>	<i>pid_name</i>	The name of the signal recipient

Context

The signal's sender.

Name

signal.send.return -- Signal being sent to a process completed

Synopsis

signal.send.return

Values

<i>retstr</i>	<i>retstr</i>	The return value to either <command>__group_send_sig_info</command>, <command>specific_send_sig_info</command>, or <command>send_sigqueue</command>
<i>send2queue</i>	<i>send2queue</i>	Indicates whether the sent signal was sent to an existing <command>sigqueue</command>
<i>name</i>	<i>name</i>	The name of the function used to send out the signal
<i>shared</i>	<i>shared</i>	Indicates whether the sent signal is shared by the thread group.

Context

The signal's sender. <remark>(correct?)</remark>

Description

Possible <command>__group_send_sig_info</command> and <command>specific_send_sig_info</command> return values are as follows;

<command>0</command> -- The signal is successfully sent to a process, which means that <1> the signal was ignored by the receiving process, <2> this is a non-RT signal and the system already has one queued, and <3> the signal was successfully added to the <command>sigqueue</command> of the receiving process.

<command>-EAGAIN</command> -- The <command>sigqueue</command> of the receiving process is overflowing, the signal was RT, and the signal was sent by a user using something other than <command>kill</command>.

Possible <command>send_group_sigqueue</command> and <command>send_sigqueue</command> return values are as follows;

<command>0</command> -- The signal was either successfully added into the <command>sigqueue</command> of the receiving process, or a <command>SI_TIMER</command> entry is already queued (in which case, the overrun count will be simply incremented).

<command>1</command> -- The signal was ignored by the receiving process.

<command>-1</command> -- (<command>send_sigqueue</command> only) The task was marked <command>exiting</command>, allowing * <command>posix_timer_event</command> to redirect it to the group leader.

Name

signal.checkperm -- Check being performed on a sent signal

Synopsis

signal.checkperm

Values

<i>name</i>	<i>name</i>	Name of the probe point; default value is <command>signal.checkperm</command>
<i>task</i>	<i>task</i>	A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i>	The address of the <command>siginfo</command> structure
<i>si_code</i>	<i>si_code</i>	Indicates the signal type
<i>sig_name</i>	<i>sig_name</i>	A string representation of the signal
<i>sig</i>	<i>sig</i>	The number of the signal
<i>pid_name</i>	<i>pid_name</i>	Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i>	The PID of the process receiving the signal

Name

signal.checkperm.return -- Check performed on a sent signal completed

Synopsis

```
signal.checkperm.return
```

Values

```
retstr      retstr  
Return value as a string
```

```
name        name  
Name of the probe point; default value is <command>signal.checkperm</command>
```

Name

signal.wakeup -- Sleeping process being wakened for signal

Synopsis

signal.wakeup

Values

<i>resume</i>	<i>resume</i>	Indicates whether to wake up a task in a <command>STOPPED</command> or <command>TRACED</command> state
<i>state_mask</i>	<i>state_mask</i>	A string representation indicating the mask of task states to wake. Possible values are <command>TASK_INTERRUPTIBLE</command>, <command>TASK_STOPPED</command>, <command>TASK_TRACED</command>, and <command>TASK_INTERRUPTIBLE</command>.
<i>pid_name</i>	<i>pid_name</i>	Name of the process to wake
<i>sig_pid</i>	<i>sig_pid</i>	The PID of the process to wake

Name

signal.check_ignored -- Checking to see signal is ignored

Synopsis

```
signal.check_ignored
```

Values

<i>sig_name</i>	<i>sig_name</i>
A string representation of the signal	
<i>sig</i>	<i>sig</i>
The number of the signal	
<i>pid_name</i>	<i>pid_name</i>
Name of the process receiving the signal	
<i>sig_pid</i>	<i>sig_pid</i>
The PID of the process receiving the signal	

Name

signal.check_ignored.return -- Check to see signal is ignored completed

Synopsis

```
signal.check_ignored.return
```

Values

```
retstr      retstr  
Return value as a string
```

```
name        name  
Name of the probe point; default value is <command>signal.checkperm</command>
```

Name

signal.force_segv -- Forcing send of <command>SIGSEGV</command>

Synopsis

signal.force_segv

Values

<i>sig_name</i>	<i>sig_name</i>	A string representation of the signal
<i>sig</i>	<i>sig</i>	The number of the signal
<i>pid_name</i>	<i>pid_name</i>	Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i>	The PID of the process receiving the signal

Name

signal.force_segv.return -- Forcing send of <command>SIGSEGV</command> complete

Synopsis

```
signal.force_segv.return
```

Values

retstr *retstr*
Return value as a string

name *name*
Name of the probe point; default value is <command>force_sigsegv</command>

Name

signal.syskill -- Sending kill signal to a process

Synopsis

```
signal.syskill
```

Values

sig sig

The specific signal sent to the process

pid pid

The PID of the process receiving the signal

Name

signal.syskill.return -- Sending kill signal completed

Synopsis

```
signal.syskill.return
```

Values

None

Name

signal.sys_tkill -- Sending a kill signal to a thread

Synopsis

```
signal.sys_tkill
```

Values

<i>sig_name</i>	<i>sig_name</i>	The specific signal sent to the process
<i>sig</i>	<i>sig</i>	The specific signal sent to the process
<i>pid</i>	<i>pid</i>	The PID of the process receiving the kill signal

Description

The `<command>tkill</command>` call is analogous to `<command>kill(2)</command>`, except that it also allows a process within a specific thread group to be targetted. Such processes are targetted through their unique thread IDs (TID).

Name

signal.systkill.return -- Sending kill signal to a thread completed

Synopsis

```
signal.systkill.return
```

Values

None

Name

signal.sys_tgkill -- Sending kill signal to a thread group

Synopsis

```
signal.sys_tgkill
```

Values

<i>sig_name</i>	<i>sig_name</i>	A string representation of the signal
<i>sig</i>	<i>sig</i>	The specific kill signal sent to the process
<i>pid</i>	<i>pid</i>	The PID of the thread receiving the kill signal
<i>tgid</i>	<i>tgid</i>	The thread group ID of the thread receiving the kill signal

Description

The <command>tgkill</command> call is similar to <command>tkill</command>, except that it also allows the caller to specify the thread group ID of the thread to be signalled. This protects against TID reuse.

Sig-
nal
_taps

Name

signal.sys_tgkill.return -- Sending kill signal to a thread group completed

Synopsis

```
signal.sys_tgkill.return
```

Values

None

Name

signal.send_sig_queue -- Queuing a signal to a process

Synopsis

```
signal.send_sig_queue
```

Values

<i>sigqueue_addr</i>	<i>sigqueue_addr</i>	The address of the signal queue
<i>sig_name</i>	<i>sig_name</i>	A string representation of the signal
<i>sig</i>	<i>sig</i>	The queued signal
<i>pid_name</i>	<i>pid_name</i>	Name of the process to which the signal is queued
<i>sig_pid</i>	<i>sig_pid</i>	The PID of the process to which the signal is queued

Name

signal.send_sig_queue.return -- Queuing a signal to a process completed

Synopsis

```
signal.send_sig_queue.return
```

Values

```
retstr    retstr  
Return value as a string
```

Name

signal.pending -- Examining pending signal

Synopsis

signal.pending

Values

sigset_size *sigset_size*
The size of the user-space signal set

sigset_add *sigset_add*
The address of the user-space signal set (<command>sigset_t</command>)

Description

This probe is used to examine a set of signals pending for delivery to a specific thread. This normally occurs when the <command>do_sigpending</command> kernel function is executed.

Name

signal.pending.return -- Examination of pending signal completed

Synopsis

```
signal.pending.return
```

Values

```
retstr    retstr  
Return value as a string
```

Name

signal.handle -- Signal handler being invoked

Synopsis

signal.handle

Values

<i>regs</i>	<i>regs</i>	The address of the kernel-mode stack area
<i>sig_code</i>	<i>sig_code</i>	The <command>si_code</command> value of the <command>siginfo</command> signal
<i>sig_mode</i>	<i>sig_mode</i>	Indicates whether the signal was a user-mode or kernel-mode signal
<i>sinfo</i>	<i>sinfo</i>	The address of the <command>siginfo</command> table
<i>oldset_addr</i>	<i>oldset_addr</i>	The address of the bitmask array of blocked signals
<i>sig</i>	<i>sig</i>	The signal number that invoked the signal handler
<i>ka_addr</i>	<i>ka_addr</i>	The address of the <command>k_sigaction</command> table associated with the signal

Name

signal.handle.return -- Signal handler invocation completed

Synopsis

```
signal.handle.return
```

Values

```
retstr    retstr  
Return value as a string
```

Name

signal.do_action -- Examining or changing a signal action

Synopsis

signal.do_action

Values

<i>sa_mask</i>	<i>sa_mask</i> The new mask of the signal
<i>oldsigact_addr</i>	<i>oldsigact_addr</i> The address of the old <command>sigaction</command> struct associated with the signal
<i>sig</i>	<i>sig</i> The signal to be examined/changed
<i>sa_handler</i>	<i>sa_handler</i> The new handler of the signal
<i>sigact_addr</i>	<i>sigact_addr</i> The address of the new <command>sigaction</command> struct associated with the signal

Name

signal.do_action.return -- Examining or changing a signal action completed

Synopsis

```
signal.do_action.return
```

Values

```
retstr    retstr  
Return value as a string
```

Name

signal.procmask -- Examining or changing blocked signals

Synopsis

```
signal.procmask
```

Values

how

how

Indicates how to change the blocked signals; possible values are `<command>SIG_BLOCK=0</command>` (for blocking signals), `<command>SIG_UNBLOCK=1</command>` (for unblocking signals), and `<command>SIG_SETMASK=2</command>` for setting the signal mask.

oldsigset_addr

oldsigset_addr

The old address of the signal set (`<command>sigset_t</command>`)

sigset

sigset

The actual value to be set for `<command>sigset_t</command>` `<remark>(correct?)</remark>`

sigset_addr

sigset_addr

The address of the signal set (`<command>sigset_t</command>`) to be implemented

Name

signal.flush -- Flusing all pending signals for a task

Synopsis

```
signal.flush
```

Values

<i>task</i>	<i>task</i>
The task handler of the process performing the flush	
<i>pid_name</i>	<i>pid_name</i>
The name of the process associated with the task performing the flush	
<i>sig_pid</i>	<i>sig_pid</i>
The PID of the process associated with the task performing the flush	

This family of probe points is used to probe signal activities. It contains the following probe points:

Name

signal.send -- Signal being sent to a process

Synopsis

signal.send

Values

<i>send2queue</i>	<i>send2queue</i> Indicates whether the signal is sent to an existing <command>sigqueue</command>
<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of <command>sinfo</command> struct
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>shared</i>	<i>shared</i> Indicates whether the signal is shared by the thread group
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal
<i>pid_name</i>	<i>pid_name</i> The name of the signal recipient

Context

The signal's sender.

SystemTap Tapset Reference™

<date>February 2010</date>

signal.sendsignal.send(3stap)

Name

signal.send -- Signal being sent to a process

signal.send -- Signal being sent to a process

Synopsis

```
signal.send
```

```
signal.send
```

Values

<i>send2queue</i>	<i>send2queue</i> Indicates whether the signal is sent to an existing <command>sigqueue</command>
<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of <command>sinfo</command> struct
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>shared</i>	<i>shared</i> Indicates whether the signal is shared by the thread group
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal
<i>pid_name</i>	<i>pid_name</i> The name of the signal recipient
<i>send2queue</i>	<i>send2queue</i> Indicates whether the signal is sent to an existing <command>sigqueue</command>

Signal Tapset

<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of <command>siginfo</command> struct
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>shared</i>	<i>shared</i> Indicates whether the signal is shared by the thread group
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal
<i>pid_name</i>	<i>pid_name</i> The name of the signal recipient

<varlistentry>*send2queue*
Indicates whether the signal is sent to an existing <command>sigqueue</command>
</varlistentry>

send2queue
Indicates whether the signal is sent to an existing <command>sigqueue</command>
Indicates whether the signal is sent to an existing <command>sigqueue</command>

<varlistentry>*name*
The name of the function used to send out the signal
</varlistentry>

name
The name of the function used to send out the signal
The name of the function used to send out the signal

<varlistentry>*task*
A task handle to the signal recipient
</varlistentry>

task
A task handle to the signal recipient
A task handle to the signal recipient

<varlistentry>*sinfo*
The address of <command>siginfo</command> struct
</varlistentry>

sinfo
The address of <command>siginfo</command> struct
The address of <command>siginfo</command> struct

<varlistentry>*si_code*
Indicates the signal type
</varlistentry>

si_code
Indicates the signal type
Indicates the signal type

<varlistentry>*sig_name*
A string representation of the signal
</varlistentry>

*sig_name**sig_name*

A string representation of the signal

A string representation of the signal

<varlistentry>*sig*

The number of the signal

</varlistentry>

*sig**sig*

The number of the signal

The number of the signal

<varlistentry>*shared*

Indicates whether the signal is shared by the thread group

</varlistentry>

*shared**shared*

Indicates whether the signal is shared by the thread group

Indicates whether the signal is shared by the thread group

<varlistentry>*sig_pid*

The PID of the process receiving the signal

</varlistentry>

*sig_pid**sig_pid*

The PID of the process receiving the signal

The PID of the process receiving the signal

<varlistentry>*pid_name*

The name of the signal recipient

</varlistentry>

*pid_name**pid_name*

The name of the signal recipient

The name of the signal recipient

Context

The signal's sender.

The signal's sender.

Name

signal.send.return -- Signal being sent to a process completed

Synopsis

signal.send.return

Values

<i>retstr</i>	<i>retstr</i> The return value to either <code><command>__group_send_sig_info</command></code> , <code><command>specific_send_sig_info</command></code> , or <code><command>send_sigqueue</command></code>
<i>send2queue</i>	<i>send2queue</i> Indicates whether the sent signal was sent to an existing <code><command>sigqueue</command></code>
<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>shared</i>	<i>shared</i> Indicates whether the sent signal is shared by the thread group.

Context

The signal's sender. `<remark>(correct?)</remark>`

Description

Possible `<command>__group_send_sig_info</command>` and `<command>specific_send_sig_info</command>` return values are as follows;

`<command>0</command>` -- The signal is successfully sent to a process, which means that `<1>` the signal was ignored by the receiving process, `<2>` this is a non-RT signal and the system already has one queued, and `<3>` the signal was successfully added to the `<command>sigqueue</command>` of the receiving process.

`<command>-EAGAIN</command>` -- The `<command>sigqueue</command>` of the receiving process is overflowing, the signal was RT, and the signal was sent by a user using something other than `<command>kill</command>`.

Possible `<command>send_group_sigqueue</command>` and `<command>send_sigqueue</command>` return values are as follows;

`<command>0</command>` -- The signal was either successfully added into the `<command>sigqueue</command>` of the receiving process, or a `<command>SI_TIMER</command>` entry is already queued (in which case, the overrun count will be simply incremented).

`<command>1</command>` -- The signal was ignored by the receiving process.

`<command>-1</command>` -- (`<command>send_sigqueue</command>` only) The task was marked `<command>exiting</command>`, allowing * `<command>posix_timer_event</command>` to redirect it to the group leader.

SystemTap Tapset Reference™

`<date>February 2010</date>`

signal.send.returnsignal.send.return(3stap)

Name

signal.send.return -- Signal being sent to a process completed

signal.send.return -- Signal being sent to a process completed

Synopsis

signal.send.return

signal.send.return

Values

<i>retstr</i>	<i>retstr</i> The return value to either <code><command>__group_send_sig_info</command></code> , <code><command>specific_send_sig_info</command></code> , or <code><command>send_sigqueue</command></code>
<i>send2queue</i>	<i>send2queue</i> Indicates whether the sent signal was sent to an existing <code><command>sigqueue</command></code>
<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>shared</i>	<i>shared</i> Indicates whether the sent signal is shared by the thread group.
<i>retstr</i>	<i>retstr</i> The return value to either <code><command>__group_send_sig_info</command></code> , <code><command>specific_send_sig_info</command></code> , or <code><command>send_sigqueue</command></code>
<i>send2queue</i>	<i>send2queue</i> Indicates whether the sent signal was sent to an existing <code><command>sigqueue</command></code>
<i>name</i>	<i>name</i> The name of the function used to send out the signal
<i>shared</i>	<i>shared</i> Indicates whether the sent signal is shared by the thread group.

<varlistentry>*retstr*

The return value to either <command>__group_send_sig_info</command>, <command>specific_send_sig_info</command>, or <command>send_sigqueue</command>

</varlistentry>

retstrretstr

The return value to either <command>__group_send_sig_info</command>, <command>specific_send_sig_info</command>, or <command>send_sigqueue</command>

The return value to either <command>__group_send_sig_info</command>, <command>specific_send_sig_info</command>, or <command>send_sigqueue</command>

<varlistentry>*send2queue*

Indicates whether the sent signal was sent to an existing <command>sigqueue</command>

</varlistentry>

*send2queue**send2queue*

Indicates whether the sent signal was sent to an existing <command>sigqueue</command>

Indicates whether the sent signal was sent to an existing <command>sigqueue</command>

<varlistentry>*name*

The name of the function used to send out the signal

</varlistentry>

*name**name*

The name of the function used to send out the signal

The name of the function used to send out the signal

<varlistentry>*shared*

Indicates whether the sent signal is shared by the thread group.

</varlistentry>

*shared**shared*

Indicates whether the sent signal is shared by the thread group.

Indicates whether the sent signal is shared by the thread group.

Context

The signal's sender. <remark>(correct?)</remark>

The signal's sender. <remark>(correct?)</remark>

Description

Possible <command>__group_send_sig_info</command> and <command>specific_send_sig_info</command> return values are as follows;

<command>0</command> -- The signal is successfully sent to a process, which means that <1> the signal was ignored by the receiving process, <2> this is a non-RT signal and the system already has one queued, and <3> the signal was successfully added to the <command>sigqueue</command> of the receiving process.

<command>-EAGAIN</command> -- The <command>sigqueue</command> of the receiving process is overflowing, the signal was RT, and the signal was sent by a user using something other than <command>kill</command>.

Possible <command>send_group_sigqueue</command> and <command>send_sigqueue</command> return values are as follows;

<command>0</command> -- The signal was either successfully added into the <command>sigqueue</command> of the receiving process, or a <command>SI_TIMER</command> entry is already queued (in which case, the overrun count will be simply incremented).

<command>1</command> -- The signal was ignored by the receiving process.

<command>-1</command> -- (<command>send_sigqueue</command> only) The task was marked <command>exiting</command>, allowing * <command>posix_timer_event</command> to redirect it to the group leader.

Possible `<command>__group_send_sig_info</command>` and `<command>specific_send_sig_info</command>` return values are as follows;

`<command>0</command>` -- The signal is successfully sent to a process, which means that <1> the signal was ignored by the receiving process, <2> this is a non-RT signal and the system already has one queued, and <3> the signal was successfully added to the `<command>sigqueue</command>` of the receiving process.

`<command>-EAGAIN</command>` -- The `<command>sigqueue</command>` of the receiving process is overflowing, the signal was RT, and the signal was sent by a user using something other than `<command>kill</command>`.

`kill`

Possible `<command>send_group_sigqueue</command>` and `<command>send_sigqueue</command>` return values are as follows;

`<command>0</command>` -- The signal was either successfully added into the `<command>sigqueue</command>` of the receiving process, or a `<command>SI_TIMER</command>` entry is already queued (in which case, the overrun count will be simply incremented).

`<command>1</command>` -- The signal was ignored by the receiving process.

`<command>-1</command>` -- (`<command>send_sigqueue</command>` only) The task was marked `<command>exiting</command>`, allowing * `<command>posix_timer_event</command>` to redirect it to the group leader.

Name

signal.checkperm -- Check being performed on a sent signal

Synopsis

signal.checkperm

Values

<i>name</i>	<i>name</i> Name of the probe point; default value is <command>signal.checkperm</command>
<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> structure
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

SystemTap Tapset Reference™

<date>February 2010</date>

signal.checkpermsignal.checkperm(3stap)

Name

signal.checkperm -- Check being performed on a sent signal

signal.checkperm -- Check being performed on a sent signal

Synopsis

signal.checkperm

signal.checkperm

Values

<i>name</i>	<i>name</i> Name of the probe point; default value is <command>signal.checkperm</command>
-------------	--

<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> structure
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal
<i>name</i>	<i>name</i> Name of the probe point; default value is <command>signal.checkperm</command>
<i>task</i>	<i>task</i> A task handle to the signal recipient
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> structure
<i>si_code</i>	<i>si_code</i> Indicates the signal type
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

<varlistentry>*name*

Name of the probe point; default value is <command>signal.checkperm</command>

</varlistentry>

namename

Name of the probe point; default value is <command>signal.checkperm</command>

Name of the probe point; default value is <command>signal.checkperm</command>

<varlistentry>*task*

A task handle to the signal recipient

</varlistentry>

tasktask

A task handle to the signal recipient

A task handle to the signal recipient

<varlistentry>*sinfo*

The address of the <command>siginfo</command> structure

</varlistentry>

sinfosinfo

The address of the <command>siginfo</command> structure

The address of the <command>siginfo</command> structure

<varlistentry>*si_code*

Indicates the signal type

</varlistentry>

si_codesi_code

Indicates the signal type

Indicates the signal type

<varlistentry>*sig_name*

A string representation of the signal

</varlistentry>

sig_namesig_name

A string representation of the signal

A string representation of the signal

<varlistentry>*sig*

The number of the signal

</varlistentry>

sigsig

The number of the signal

The number of the signal

<varlistentry>*pid_name*

Name of the process receiving the signal

</varlistentry>

pid_namepid_name

Name of the process receiving the signal

Name of the process receiving the signal

<varlistentry>*sig_pid*

The PID of the process receiving the signal

</varlistentry>

sig_pidsig_pid

The PID of the process receiving the signal

The PID of the process receiving the signal

Name

signal.checkperm.return -- Check performed on a sent signal completed

Synopsis

```
signal.checkperm.return
```

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

SystemTap Tapset Reference™

<date>February 2010</date>

signal.checkperm.returnsignal.checkperm.return(3stap)

Name

signal.checkperm.return -- Check performed on a sent signal completed

signal.checkperm.return -- Check performed on a sent signal completed

Synopsis

```
signal.checkperm.return
```

```
signal.checkperm.return
```

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

<varlistentry>*retstr*

Return value as a string

</varlistentry>

retstrretstr

Return value as a string

Return value as a string

<varlistentry>*name*

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

</varlistentry>

namenname

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

Name

signal.wakeup -- Sleeping process being wakened for signal

Synopsis

signal.wakeup

Values

<i>resume</i>	<i>resume</i> Indicates whether to wake up a task in a <code><command>STOPPED</command></code> or <code><command>TRACED</command></code> state
<i>state_mask</i>	<i>state_mask</i> A string representation indicating the mask of task states to wake. Possible values are <code><command>TASK_INTERRUPTIBLE</command></code> , <code><command>TASK_STOPPED</command></code> , <code><command>TASK_TRACED</command></code> , and <code><command>TASK_INTERRUPTIBLE</command></code> .
<i>pid_name</i>	<i>pid_name</i> Name of the process to wake
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to wake

SystemTap Tapset Reference™

[<date>February 2010</date>](#)

signal.wakeupsignal.wakeup(3stap)

Name

signal.wakeup -- Sleeping process being wakened for signal

signal.wakeup -- Sleeping process being wakened for signal

Synopsis

signal.wakeup

signal.wakeup

Values

<i>resume</i>	<i>resume</i> Indicates whether to wake up a task in a <code><command>STOPPED</command></code> or <code><command>TRACED</command></code> state
<i>state_mask</i>	<i>state_mask</i> A string representation indicating the mask of task states to wake. Possible values are <code><command>TASK_INTERRUPTIBLE</command></code> , <code><command>TASK_STOPPED</command></code> , <code><command>TASK_TRACED</command></code> , and <code><command>TASK_INTERRUPTIBLE</command></code> .

Signal Tapset

<i>pid_name</i>	<i>pid_name</i> Name of the process to wake
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to wake
<i>resume</i>	<i>resume</i> Indicates whether to wake up a task in a <code><command>STOPPED</command></code> or <code><command>TRACED</command></code> state
<i>state_mask</i>	<i>state_mask</i> A string representation indicating the mask of task states to wake. Possible values are <code><command>TASK_INTERRUPTIBLE</command></code> , <code><command>TASK_STOPPED</command></code> , <code><command>TASK_TRACED</command></code> , and <code><command>TASK_INTERRUPTIBLE</command></code> .
<i>pid_name</i>	<i>pid_name</i> Name of the process to wake
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to wake

`<varlistentry>resume`
Indicates whether to wake up a task in a `<command>STOPPED</command>` or `<command>TRACED</command>` state
`</varlistentry>`

resumeresume
Indicates whether to wake up a task in a `<command>STOPPED</command>` or `<command>TRACED</command>` state
Indicates whether to wake up a task in a `<command>STOPPED</command>` or `<command>TRACED</command>` state

`<varlistentry>state_mask`
A string representation indicating the mask of task states to wake. Possible values are `<command>TASK_INTERRUPTIBLE</command>`, `<command>TASK_STOPPED</command>`, `<command>TASK_TRACED</command>`, and `<command>TASK_INTERRUPTIBLE</command>`.
`</varlistentry>`

state_maskstate_mask
A string representation indicating the mask of task states to wake. Possible values are `<command>TASK_INTERRUPTIBLE</command>`, `<command>TASK_STOPPED</command>`, `<command>TASK_TRACED</command>`, and `<command>TASK_INTERRUPTIBLE</command>`.
A string representation indicating the mask of task states to wake. Possible values are `<command>TASK_INTERRUPTIBLE</command>`, `<command>TASK_STOPPED</command>`, `<command>TASK_TRACED</command>`, and `<command>TASK_INTERRUPTIBLE</command>`.

`<varlistentry>pid_name`
Name of the process to wake
`</varlistentry>`

pid_namepid_name
Name of the process to wake
Name of the process to wake

`<varlistentry>sig_pid`
The PID of the process to wake
`</varlistentry>`

sig_pidsig_pid
The PID of the process to wake
The PID of the process to wake

Name

signal.check_ignored -- Checking to see signal is ignored

Synopsis

signal.check_ignored

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

SystemTap Tapset Reference™

<date>February 2010</date>

signal.check_ignoredsignal.check_ignored(3stap)

Name

signal.check_ignored -- Checking to see signal is ignored

signal.check_ignored -- Checking to see signal is ignored

Synopsis

signal.check_ignored

signal.check_ignored

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

Signal Tapset

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

<varlistentry>*sig_name*
A string representation of the signal

</varlistentry>

*sig_name**sig_name*
A string representation of the signal
A string representation of the signal

<varlistentry>*sig*
The number of the signal

</varlistentry>

*sig**sig*
The number of the signal
The number of the signal

<varlistentry>*pid_name*
Name of the process receiving the signal

</varlistentry>

*pid_name**pid_name*
Name of the process receiving the signal
Name of the process receiving the signal

<varlistentry>*sig_pid*
The PID of the process receiving the signal

</varlistentry>

*sig_pid**sig_pid*
The PID of the process receiving the signal
The PID of the process receiving the signal

Name

signal.check_ignored.return -- Check to see signal is ignored completed

Synopsis

signal.check_ignored.return

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

SystemTap Tapset Reference™

<date>February 2010</date>

signal.check_ignored.returnsignal.check_ignored.return(3stap)

Name

signal.check_ignored.return -- Check to see signal is ignored completed

signal.check_ignored.return -- Check to see signal is ignored completed

Synopsis

signal.check_ignored.return

signal.check_ignored.return

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>signal.checkperm</command>

<varlistentry>*retstr*

Return value as a string

</varlistentry>

retstrretstr

Return value as a string

Return value as a string

<varlistentry>*name*

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

</varlistentry>

namenname

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

Name of the probe point; default value is **<command>**signal.checkperm**</command>**

Name

signal.force_segv -- Forcing send of <command>SIGSEGV</command>

Synopsis

signal.force_segv

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

SystemTap Tapset Reference™

<date>February 2010</date>

signal.force_segvsignal.force_segv(3stap)

Name

signal.force_segv -- Forcing send of <command>SIGSEGV</command>

signal.force_segv -- Forcing send of <command>SIGSEGV</command>

Synopsis

signal.force_segv

signal.force_segv

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

Signal Tapset

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The number of the signal
<i>pid_name</i>	<i>pid_name</i> Name of the process receiving the signal
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process receiving the signal

<varlistentry>*sig_name*
A string representation of the signal

</varlistentry>

*sig_name**sig_name*
A string representation of the signal
A string representation of the signal

<varlistentry>*sig*
The number of the signal

</varlistentry>

*sig**sig*
The number of the signal
The number of the signal

<varlistentry>*pid_name*
Name of the process receiving the signal

</varlistentry>

*pid_name**pid_name*
Name of the process receiving the signal
Name of the process receiving the signal

<varlistentry>*sig_pid*
The PID of the process receiving the signal

</varlistentry>

*sig_pid**sig_pid*
The PID of the process receiving the signal
The PID of the process receiving the signal

Name

signal.force_segv.return -- Forcing send of <command>SIGSEGV</command> complete

Synopsis

```
signal.force_segv.return
```

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>force_sigsegv</command>

SystemTap Tapset Reference™

<date>February 2010</date>

signal.force_segv.returnsignal.force_segv.return(3stap)

Name

signal.force_segv.return -- Forcing send of <command>SIGSEGV</command> complete

signal.force_segv.return -- Forcing send of <command>SIGSEGV</command> complete

Synopsis

```
signal.force_segv.return
```

```
signal.force_segv.return
```

Values

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>force_sigsegv</command>

<i>retstr</i>	<i>retstr</i>
	Return value as a string

<i>name</i>	<i>name</i>
	Name of the probe point; default value is <command>force_sigsegv</command>

<varlistentry>*retstr*

Return value as a string

</varlistentry>

retstrretstr

Return value as a string

Return value as a string

<varlistentry>*name*

Name of the probe point; default value is <command>force_sigsegv</command>

</varlistentry>

namenname

Name of the probe point; default value is <command>force_sigsegv</command>

Name of the probe point; default value is <command>force_sigsegv</command>

Name

signal.syskill -- Sending kill signal to a process

Synopsis

signal.syskill

Values

sig sig
The specific signal sent to the process

pid pid
The PID of the process receiving the signal

SystemTap Tapset Reference™
<date>February 2010</date>
signal.syskillsignal.syskill(3stap)

Name

signal.syskill -- Sending kill signal to a process

signal.syskill -- Sending kill signal to a process

Synopsis

signal.syskill

signal.syskill

Values

sig sig
The specific signal sent to the process

pid pid
The PID of the process receiving the signal

sig sig
The specific signal sent to the process

pid pid
The PID of the process receiving the signal

<varlistentry>sig

The specific signal sent to the process

</varlistentry>

sigsig

The specific signal sent to the process

The specific signal sent to the process

<varlistentry>pid

The PID of the process receiving the signal

</varlistentry>

pidpid

The PID of the process receiving the signal

The PID of the process receiving the signal

Name

signal.syskill.return -- Sending kill signal completed

Synopsis

signal.syskill.return

Values

None

SystemTap Tapset Reference™

<date>February 2010</date>

signal.syskill.returnsignal.syskill.return(3stap)

Name

signal.syskill.return -- Sending kill signal completed

signal.syskill.return -- Sending kill signal completed

Synopsis

signal.syskill.return

signal.syskill.return

Values

None

None

Name

signal.sys_tkill -- Sending a kill signal to a thread

Synopsis

```
signal.sys_tkill
```

Values

<i>sig_name</i>	<i>sig_name</i> The specific signal sent to the process
<i>sig</i>	<i>sig</i> The specific signal sent to the process
<i>pid</i>	<i>pid</i> The PID of the process receiving the kill signal

Description

The `<command>tkill</command>` call is analogous to `<command>kill(2)</command>`, except that it also allows a process within a specific thread group to be targetted. Such processes are targetted through their unique thread IDs (TID).

SystemTap Tapset Reference™

<date>February 2010</date>

signal.sys_tkillsignal.sys_tkill(3stap)

Name

signal.sys_tkill -- Sending a kill signal to a thread

signal.sys_tkill -- Sending a kill signal to a thread

Synopsis

```
signal.sys_tkill
```

```
signal.sys_tkill
```

Values

<i>sig_name</i>	<i>sig_name</i> The specific signal sent to the process
<i>sig</i>	<i>sig</i> The specific signal sent to the process
<i>pid</i>	<i>pid</i> The PID of the process receiving the kill signal

Signal Tapset

<i>sig_name</i>	<i>sig_name</i> The specific signal sent to the process
<i>sig</i>	<i>sig</i> The specific signal sent to the process
<i>pid</i>	<i>pid</i> The PID of the process receiving the kill signal

<varlistentry>*sig_name*
The specific signal sent to the process
</varlistentry>

*sig_name**sig_name*
The specific signal sent to the process
The specific signal sent to the process
<varlistentry>*sig*
The specific signal sent to the process
</varlistentry>

*sig**sig*
The specific signal sent to the process
The specific signal sent to the process
<varlistentry>*pid*
The PID of the process receiving the kill signal
</varlistentry>

*pid**pid*
The PID of the process receiving the kill signal
The PID of the process receiving the kill signal

Description

The **<command>tkill</command>** call is analogous to **<command>kill(2)</command>**, except that it also allows a process within a specific thread group to be targetted. Such processes are targetted through their unique thread IDs (TID).

The **<command>tkill</command>** call is analogous to **<command>kill(2)</command>**, except that it also allows a process within a specific thread group to be targetted. Such processes are targetted through their unique thread IDs (TID).

Name

signal.systkill.return -- Sending kill signal to a thread completed

Synopsis

```
signal.systkill.return
```

Values

None

SystemTap Tapset Reference™

<date>February 2010</date>

signal.systkill.returnsignal.systkill.return(3stap)

Name

signal.systkill.return -- Sending kill signal to a thread completed

signal.systkill.return -- Sending kill signal to a thread completed

Synopsis

```
signal.systkill.return
```

```
signal.systkill.return
```

Values

None

None

Name

signal.sys_tgkill -- Sending kill signal to a thread group

Synopsis

```
signal.sys_tgkill
```

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The specific kill signal sent to the process
<i>pid</i>	<i>pid</i> The PID of the thread receiving the kill signal
<i>tgid</i>	<i>tgid</i> The thread group ID of the thread receiving the kill signal

Description

The `<command>tgkill</command>` call is similar to `<command>tkill</command>`, except that it also allows the caller to specify the thread group ID of the thread to be signalled. This protects against TID reuse.

SystemTap Tapset Reference™

<date>February 2010</date>

signal.sys_tgkillsignal.sys_tgkill(3stap)

Name

signal.sys_tgkill -- Sending kill signal to a thread group

signal.sys_tgkill -- Sending kill signal to a thread group

Synopsis

```
signal.sys_tgkill
```

```
signal.sys_tgkill
```

Values

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The specific kill signal sent to the process
<i>pid</i>	<i>pid</i> The PID of the thread receiving the kill signal

tgid *tgid*
The thread group ID of the thread receiving the kill signal

sig_name *sig_name*
A string representation of the signal

sig *sig*
The specific kill signal sent to the process

pid *pid*
The PID of the thread receiving the kill signal

tgid *tgid*
The thread group ID of the thread receiving the kill signal

<varlistentry>*sig_name*
A string representation of the signal
</varlistentry>

sig_namesig_name
A string representation of the signal
A string representation of the signal
<varlistentry>*sig*
The specific kill signal sent to the process
</varlistentry>

sigsig
The specific kill signal sent to the process
The specific kill signal sent to the process
<varlistentry>*pid*
The PID of the thread receiving the kill signal
</varlistentry>

pidpid
The PID of the thread receiving the kill signal
The PID of the thread receiving the kill signal
<varlistentry>*tgid*
The thread group ID of the thread receiving the kill signal
</varlistentry>

tgidtgid
The thread group ID of the thread receiving the kill signal
The thread group ID of the thread receiving the kill signal

Description

The <command>tkill</command> call is similar to <command>kill</command>, except that it also allows the caller to specify the thread group ID of the thread to be signalled. This protects against TID reuse.

The <command>tckill</command> call is similar to <command>kill</command>, except that it also allows the caller to specify the thread group ID of the thread to be signalled. This protects against TID reuse.

Name

signal.sys_tgkill.return -- Sending kill signal to a thread group completed

Synopsis

```
signal.sys_tgkill.return
```

Values

None

SystemTap Tapset Reference™

<date>February 2010</date>

signal.sys_tgkill.returnsignal.sys_tgkill.return(3stap)

Name

signal.sys_tgkill.return -- Sending kill signal to a thread group completed

signal.sys_tgkill.return -- Sending kill signal to a thread group completed

Synopsis

```
signal.sys_tgkill.return
```

```
signal.sys_tgkill.return
```

Values

None

None

Name

signal.send_sig_queue -- Queuing a signal to a process

Synopsis

```
signal.send_sig_queue
```

Values

<i>sigqueue_addr</i>	<i>sigqueue_addr</i> The address of the signal queue
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The queued signal
<i>pid_name</i>	<i>pid_name</i> Name of the process to which the signal is queued
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to which the signal is queued

SystemTap Tapset Reference™

<date>February 2010</date>

signal.send_sig_queuesignal.send_sig_queue(3stap)

Name

signal.send_sig_queue -- Queuing a signal to a process

signal.send_sig_queue -- Queuing a signal to a process

Synopsis

```
signal.send_sig_queue
```

```
signal.send_sig_queue
```

Values

<i>sigqueue_addr</i>	<i>sigqueue_addr</i> The address of the signal queue
<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
<i>sig</i>	<i>sig</i> The queued signal
<i>pid_name</i>	<i>pid_name</i> Name of the process to which the signal is queued

<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to which the signal is queued
----------------	--

<i>sigqueue_addr</i>	<i>sigqueue_addr</i> The address of the signal queue
----------------------	---

<i>sig_name</i>	<i>sig_name</i> A string representation of the signal
-----------------	--

<i>sig</i>	<i>sig</i> The queued signal
------------	---------------------------------

<i>pid_name</i>	<i>pid_name</i> Name of the process to which the signal is queued
-----------------	--

<i>sig_pid</i>	<i>sig_pid</i> The PID of the process to which the signal is queued
----------------	--

<varlistentry>*sigqueue_addr*
The address of the signal queue
</varlistentry>

*sigqueue_addr**sigqueue_addr*
The address of the signal queue
The address of the signal queue

<varlistentry>*sig_name*
A string representation of the signal
</varlistentry>

*sig_name**sig_name*
A string representation of the signal
A string representation of the signal

<varlistentry>*sig*
The queued signal
</varlistentry>

*sig**sig*
The queued signal
The queued signal

<varlistentry>*pid_name*
Name of the process to which the signal is queued
</varlistentry>

*pid_name**pid_name*
Name of the process to which the signal is queued
Name of the process to which the signal is queued

<varlistentry>*sig_pid*
The PID of the process to which the signal is queued
</varlistentry>

*sig_pid**sig_pid*
The PID of the process to which the signal is queued
The PID of the process to which the signal is queued

Name

signal.send_sig_queue.return -- Queuing a signal to a process completed

Synopsis

```
signal.send_sig_queue.return
```

Values

```
retstr    retstr
          Return value as a string
```

SystemTap Tapset Reference™

<date>February 2010</date>

```
signal.send_sig_queue.returnsignal.send_sig_queue.return(3stap)
```

Name

signal.send_sig_queue.return -- Queuing a signal to a process completed

signal.send_sig_queue.return -- Queuing a signal to a process completed

Synopsis

```
signal.send_sig_queue.return
```

```
signal.send_sig_queue.return
```

Values

```
retstr    retstr
          Return value as a string
```

```
retstr    retstr
          Return value as a string
```

```
<varlistentry>retstr
Return value as a string
</varlistentry>
retstrretstr
Return value as a string
Return value as a string
```

Name

signal.pending -- Examining pending signal

Synopsis

signal.pending

Values

<i>sigset_size</i>	<i>sigset_size</i> The size of the user-space signal set
<i>sigset_add</i>	<i>sigset_add</i> The address of the user-space signal set (<command>sigset_t</command>)

Description

This probe is used to examine a set of signals pending for delivery to a specific thread. This normally occurs when the <command>do_sigpending</command> kernel function is executed.

SystemTap Tapset Reference™

<date>February 2010</date>

signal.pendingsignal.pending(3stap)

Name

signal.pending -- Examining pending signal

signal.pending -- Examining pending signal

Synopsis

signal.pending

signal.pending

Values

<i>sigset_size</i>	<i>sigset_size</i> The size of the user-space signal set
<i>sigset_add</i>	<i>sigset_add</i> The address of the user-space signal set (<command>sigset_t</command>)
<i>sigset_size</i>	<i>sigset_size</i> The size of the user-space signal set
<i>sigset_add</i>	<i>sigset_add</i> The address of the user-space signal set (<command>sigset_t</command>)

<varlistentry>*sigset_size*

The size of the user-space signal set

</varlistentry>

*sigset_size**sigset_size*

The size of the user-space signal set

The size of the user-space signal set

<varlistentry>*sigset_add*

The address of the user-space signal set (<command>*sigset_t*</command>)

</varlistentry>

*sigset_add**sigset_add*

The address of the user-space signal set (<command>*sigset_t*</command>)

The address of the user-space signal set (<command>*sigset_t*</command>)

Description

This probe is used to examine a set of signals pending for delivery to a specific thread. This normally occurs when the <command>*do_sigpending*</command> kernel function is executed.

This probe is used to examine a set of signals pending for delivery to a specific thread. This normally occurs when the <command>*do_sigpending*</command> kernel function is executed.

Name

signal.pending.return -- Examination of pending signal completed

Synopsis

signal.pending.return

Values

retstr *retstr*
Return value as a string

SystemTap Tapset Reference™

<date>February 2010</date>

signal.pending.returnsignal.pending.return(3stap)

Name

signal.pending.return -- Examination of pending signal completed

signal.pending.return -- Examination of pending signal completed

Synopsis

signal.pending.return

signal.pending.return

Values

retstr *retstr*
Return value as a string

retstr *retstr*
Return value as a string

<varlistentry>*retstr*
Return value as a string
</varlistentry>

retstrretstr
Return value as a string
Return value as a string

Name

signal.handle -- Signal handler being invoked

Synopsis

signal.handle

Values

<i>regs</i>	<i>regs</i> The address of the kernel-mode stack area
<i>sig_code</i>	<i>sig_code</i> The <command>si_code</command> value of the <command>siginfo</command> signal
<i>sig_mode</i>	<i>sig_mode</i> Indicates whether the signal was a user-mode or kernel-mode signal
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> table
<i>oldset_addr</i>	<i>oldset_addr</i> The address of the bitmask array of blocked signals
<i>sig</i>	<i>sig</i> The signal number that invoked the signal handler
<i>ka_addr</i>	<i>ka_addr</i> The address of the <command>k_sigaction</command> table associated with the signal

SystemTap Tapset Reference™

<date>February 2010</date>

signal.handlesignal.handle(3stap)

Name

signal.handle -- Signal handler being invoked

signal.handle -- Signal handler being invoked

Synopsis

signal.handle

signal.handle

Values

<i>regs</i>	<i>regs</i> The address of the kernel-mode stack area
-------------	--

<i>sig_code</i>	<i>sig_code</i> The <command>si_code</command> value of the <command>siginfo</command> signal
<i>sig_mode</i>	<i>sig_mode</i> Indicates whether the signal was a user-mode or kernel-mode signal
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> table
<i>oldset_addr</i>	<i>oldset_addr</i> The address of the bitmask array of blocked signals
<i>sig</i>	<i>sig</i> The signal number that invoked the signal handler
<i>ka_addr</i>	<i>ka_addr</i> The address of the <command>k_sigaction</command> table associated with the signal
<i>regs</i>	<i>regs</i> The address of the kernel-mode stack area
<i>sig_code</i>	<i>sig_code</i> The <command>si_code</command> value of the <command>siginfo</command> signal
<i>sig_mode</i>	<i>sig_mode</i> Indicates whether the signal was a user-mode or kernel-mode signal
<i>sinfo</i>	<i>sinfo</i> The address of the <command>siginfo</command> table
<i>oldset_addr</i>	<i>oldset_addr</i> The address of the bitmask array of blocked signals
<i>sig</i>	<i>sig</i> The signal number that invoked the signal handler
<i>ka_addr</i>	<i>ka_addr</i> The address of the <command>k_sigaction</command> table associated with the signal

<varlistentry>*regs*

The address of the kernel-mode stack area

</varlistentry>

regsregs

The address of the kernel-mode stack area

The address of the kernel-mode stack area

<varlistentry>*sig_code*

The <command>*si_code*</command> value of the <command>*siginfo*</command> signal

</varlistentry>

sig_codesig_code

The <command>*si_code*</command> value of the <command>*siginfo*</command> signal

The <command>*si_code*</command> value of the <command>*siginfo*</command> signal

<varlistentry>*sig_mode*

Indicates whether the signal was a user-mode or kernel-mode signal

</varlistentry>

sig_modesig_mode

Indicates whether the signal was a user-mode or kernel-mode signal

Indicates whether the signal was a user-mode or kernel-mode signal

<varlistentry>*sinfo*

The address of the <command>*siginfo*</command> table

</varlistentry>

sinfosinfo

The address of the <command>*siginfo*</command> table

The address of the <command>*siginfo*</command> table

<varlistentry>*oldset_addr*

The address of the bitmask array of blocked signals

</varlistentry>

oldset_addroldset_addr

The address of the bitmask array of blocked signals

The address of the bitmask array of blocked signals

<varlistentry>*sig*

The signal number that invoked the signal handler

</varlistentry>

sigsig

The signal number that invoked the signal handler

The signal number that invoked the signal handler

<varlistentry>*ka_addr*

The address of the <command>*k_sigaction*</command> table associated with the signal

</varlistentry>

ka_addrka_addr

The address of the <command>*k_sigaction*</command> table associated with the signal

The address of the <command>*k_sigaction*</command> table associated with the signal

Name

signal.handle.return -- Signal handler invocation completed

Synopsis

signal.handle.return

Values

retstr *retstr*
Return value as a string

SystemTap Tapset Reference™

<date>February 2010</date>

signal.handle.returnsignal.handle.return(3stap)

Name

signal.handle.return -- Signal handler invocation completed

signal.handle.return -- Signal handler invocation completed

Synopsis

signal.handle.return

signal.handle.return

Values

retstr *retstr*
Return value as a string

retstr *retstr*
Return value as a string

<varlistentry>*retstr*
Return value as a string
</varlistentry>

retstrretstr
Return value as a string
Return value as a string

Name

signal.do_action -- Examining or changing a signal action

Synopsis

signal.do_action

Values

<i>sa_mask</i>	<i>sa_mask</i> The new mask of the signal
<i>oldsigact_addr</i>	<i>oldsigact_addr</i> The address of the old <command>sigaction</command> struct associated with the signal
<i>sig</i>	<i>sig</i> The signal to be examined/changed
<i>sa_handler</i>	<i>sa_handler</i> The new handler of the signal
<i>sigact_addr</i>	<i>sigact_addr</i> The address of the new <command>sigaction</command> struct associated with the signal

SystemTap Tapset Reference™

<date>February 2010</date>

signal.do_actionsignal.do_action(3stap)

Name

signal.do_action -- Examining or changing a signal action

signal.do_action -- Examining or changing a signal action

Synopsis

signal.do_action

signal.do_action

Values

<i>sa_mask</i>	<i>sa_mask</i> The new mask of the signal
<i>oldsigact_addr</i>	<i>oldsigact_addr</i> The address of the old <command>sigaction</command> struct associated with the signal
<i>sig</i>	<i>sig</i> The signal to be examined/changed

<i>sa_handler</i>	<i>sa_handler</i> The new handler of the signal
<i>sigact_addr</i>	<i>sigact_addr</i> The address of the new <command>sigaction</command> struct associated with the signal
<i>sa_mask</i>	<i>sa_mask</i> The new mask of the signal
<i>oldsigact_addr</i>	<i>oldsigact_addr</i> The address of the old <command>sigaction</command> struct associated with the signal
<i>sig</i>	<i>sig</i> The signal to be examined/changed
<i>sa_handler</i>	<i>sa_handler</i> The new handler of the signal
<i>sigact_addr</i>	<i>sigact_addr</i> The address of the new <command>sigaction</command> struct associated with the signal

<varlistentry>*sa_mask*
The new mask of the signal
</varlistentry>
sa_masksa_mask
The new mask of the signal
The new mask of the signal
<varlistentry>*oldsigact_addr*
The address of the old <command>sigaction</command> struct associated with the signal
</varlistentry>
oldsigact_addr
The address of the old <command>sigaction</command> struct associated with the signal
The address of the old <command>sigaction</command> struct associated with the signal
<varlistentry>*sig*
The signal to be examined/changed
</varlistentry>
sigsig
The signal to be examined/changed
The signal to be examined/changed
<varlistentry>*sa_handler*
The new handler of the signal
</varlistentry>
sa_handlersa_handler
The new handler of the signal
The new handler of the signal
<varlistentry>*sigact_addr*
The address of the new <command>sigaction</command> struct associated with the signal
</varlistentry>
sigact_addrsigact_addr
The address of the new <command>sigaction</command> struct associated with the signal
The address of the new <command>sigaction</command> struct associated with the signal

Name

signal.do_action.return -- Examining or changing a signal action completed

Synopsis

```
signal.do_action.return
```

Values

```
retstr    retstr
          Return value as a string
```

SystemTap Tapset Reference™

<date>February 2010</date>

```
signal.do_action.returnsignal.do_action.return(3stap)
```

Name

signal.do_action.return -- Examining or changing a signal action completed

signal.do_action.return -- Examining or changing a signal action completed

Synopsis

```
signal.do_action.return
```

```
signal.do_action.return
```

Values

```
retstr    retstr
          Return value as a string
```

```
retstr    retstr
          Return value as a string
```

```
<varlistentry>retstr
Return value as a string
</varlistentry>
```

```
retstrretstr
Return value as a string
Return value as a string
```

Name

signal.procmask -- Examining or changing blocked signals

Synopsis

signal.procmask

Values

how

how

Indicates how to change the blocked signals; possible values are `<command>SIG_BLOCK=0</command>` (for blocking signals), `<command>SIG_UNBLOCK=1</command>` (for unblocking signals), and `<command>SIG_SETMASK=2</command>` for setting the signal mask.

oldsigset_addr

oldsigset_addr

The old address of the signal set (`<command>sigset_t</command>`)

sigset

sigset

The actual value to be set for `<command>sigset_t</command>`
<remark>(correct?)</remark>

sigset_addr

sigset_addr

The address of the signal set (`<command>sigset_t</command>`) to be implemented

SystemTap Tapset Reference™

<date>February 2010</date>

signal.procmasksignal.procmask(3stap)

Name

signal.procmask -- Examining or changing blocked signals

signal.procmask -- Examining or changing blocked signals

Synopsis

signal.procmask

signal.procmask

Values

how

how

Indicates how to change the blocked signals; possible values are `<command>SIG_BLOCK=0</command>` (for blocking signals), `<command>SIG_UNBLOCK=1</command>` (for unblocking signals), and `<command>SIG_SETMASK=2</command>` for setting the signal mask.

oldsigset_addr

oldsigset_addr

The old address of the signal set (`<command>sigset_t</command>`)

<i>sigset</i>	<i>sigset</i> The actual value to be set for <command>sigset_t</command> <remark>(correct?)</remark>
<i>sigset_addr</i>	<i>sigset_addr</i> The address of the signal set (<command>sigset_t</command>) to be implemented
<i>how</i>	<i>how</i> Indicates how to change the blocked signals; possible values are <command>SIG_BLOCK=0</command> (for blocking signals), <command>SIG_UNBLOCK=1</command> (for unblocking signals), and <command>SIG_SETMASK=2</command> for setting the signal mask.
<i>oldsigset_addr</i>	<i>oldsigset_addr</i> The old address of the signal set (<command>sigset_t</command>)
<i>sigset</i>	<i>sigset</i> The actual value to be set for <command>sigset_t</command> <remark>(correct?)</remark>
<i>sigset_addr</i>	<i>sigset_addr</i> The address of the signal set (<command>sigset_t</command>) to be implemented

<varlistentry>*how*

Indicates how to change the blocked signals; possible values are <command>SIG_BLOCK=0</command> (for blocking signals), <command>SIG_UNBLOCK=1</command> (for unblocking signals), and <command>SIG_SETMASK=2</command> for setting the signal mask.

</varlistentry>

howhow

Indicates how to change the blocked signals; possible values are <command>SIG_BLOCK=0</command> (for blocking signals), <command>SIG_UNBLOCK=1</command> (for unblocking signals), and <command>SIG_SETMASK=2</command> for setting the signal mask.

Indicates how to change the blocked signals; possible values are <command>SIG_BLOCK=0</command> (for blocking signals), <command>SIG_UNBLOCK=1</command> (for unblocking signals), and <command>SIG_SETMASK=2</command> for setting the signal mask.

<varlistentry>*oldsigset_addr*

The old address of the signal set (<command>sigset_t</command>)

</varlistentry>

*oldsigset_addr**oldsigset_addr*

The old address of the signal set (<command>sigset_t</command>)

The old address of the signal set (<command>sigset_t</command>)

<varlistentry>*sigset*

The actual value to be set for <command>sigset_t</command> <remark>(correct?)</remark>

</varlistentry>

sigsetsigset

The actual value to be set for <command>sigset_t</command> <remark>(correct?)</remark>

The actual value to be set for <command>sigset_t</command> <remark>(correct?)</remark>

<varlistentry>*sigset_addr*

The address of the signal set (<command>sigset_t</command>) to be implemented

</varlistentry>

*sigset_addr**sigset_addr*

The address of the signal set (<command>sigset_t</command>) to be implemented

The address of the signal set (<command>sigset_t</command>) to be implemented

Name

signal.flush -- Flusing all pending signals for a task

Synopsis

```
signal.flush
```

Values

<i>task</i>	<i>task</i> The task handler of the process performing the flush
<i>pid_name</i>	<i>pid_name</i> The name of the process associated with the task performing the flush
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process associated with the task performing the flush

SystemTap Tapset Reference™
<date>February 2010</date>
signal.flushsignal.flush(3stap)

Name

signal.flush -- Flusing all pending signals for a task

signal.flush -- Flusing all pending signals for a task

Synopsis

```
signal.flush
```

```
signal.flush
```

Values

<i>task</i>	<i>task</i> The task handler of the process performing the flush
<i>pid_name</i>	<i>pid_name</i> The name of the process associated with the task performing the flush
<i>sig_pid</i>	<i>sig_pid</i> The PID of the process associated with the task performing the flush

<i>task</i>	<i>task</i> The task handler of the process performing the flush
<i>pid_name</i>	<i>pid_name</i> The name of the process associated with the task performing the flush

sig_pid

sig_pid

The PID of the process associated with the task performing the flush

<varlistentry>*task*

The task handler of the process performing the flush

</varlistentry>

tasktask

The task handler of the process performing the flush

The task handler of the process performing the flush

<varlistentry>*pid_name*

The name of the process associated with the task performing the flush

</varlistentry>

pid_namepid_name

The name of the process associated with the task performing the flush

The name of the process associated with the task performing the flush

<varlistentry>*sig_pid*

The PID of the process associated with the task performing the flush

</varlistentry>

sig_pidsig_pid

The PID of the process associated with the task performing the flush

The PID of the process associated with the task performing the flush