Postmodern strace

Dmitry Levin

Brussels, 2020

Traditional strace

Printing instruction pointer and timestamps

- print instruction pointer: **-i** option
- print timestamps: **-r**, **-t**, **-tt**, **-ttt**, and **-T** options

Size and format of strings

- string size: **-s** option
- string format: **-x** and **-xx** options

Verbosity of syscall decoding

- abbreviate output: **-e abbrev=*set***, **-v** option
- dereference structures: **-e verbose=*set***
- print raw undecoded syscalls: **-e raw=*set***

Traditional strace

Printing signals

- print signals: -e **signal**=*set*

Dumping

- dump the data read from the specified descriptors: -e **read**=*set*
- dump the data written to the specified descriptors: -e **write**=*set*

Redirecting output to files or pipelines

- write the trace to a file or pipeline: -o ***filename*** option
- write traces of processes to separate files: -ff -o ***filename***

Traditional strace

System call filtering

- trace only the specified set of system calls: **-e trace=*set***

System call statistics

- count time, calls, and errors for each system call: **-c** option
- sort the histogram printed by the **-c** option: **-S *sortby*** option

Tracing control

- attach to existing processes: **-p *pid*** option
- trace child processes: **-f** option

Modern strace

Tracing output format

- pathnames accessed by name or descriptor: **-y** option
- network protocol associated with descriptors: **-yy** option
- stack of function calls: **-k** option

System call filtering

- pathnames accessed by name or descriptor: **-P** option
- regular expressions: -e **trace**=/*regexp*
- optional specifications: -e **trace**=?*spec*
- new syscall classes: %stat, %lstat, %fstat, %statfs, %fstatfs, %%stat, %%statfs

Modern strace

System call statistics

- wall clock time spent in syscalls: **-w** option
- combine statistics with regular output: **-C** option

Tracing control

- attach to multiple processes: **-p *pid_set*** option
- detach on execve: **-b execve** option
- run as a detached grandchild: **-D** option
- interruptibility: **-I** option
- postprocessing: **strace-log-merge**

Modern strace

System call tampering

- fault injection:

  -e **inject**=*set*:**error**=*errno*[:**when**=*expr*][:**syscall**=*syscall*]

- return value injection:

  -e **inject**=*set*:**retval**=*value*[:**when**=*expr*][:**syscall**=*syscall*]

- signal injection:

  -e **inject**=*set*:**signal**=*set*

- delay injection:

  -e **inject**=*set*:**delay_enter**=*usecs*

  -e **inject**=*set*:**delay_exit**=*usecs*

New features since FOSDEM 2018

- PTRACE_GET_SYSCALL_INFO API support
- system call return status filtering:
  -e status=*set*, **-z**, **-Z** options
- seccomp-assisted system call filtering: **--seccomp-bpf** option
- format of named constants and flags: **-X** option
- support of new system calls (≈ 35)
- elaborate syscall parsers
- long options
- copyleft license

x86-64 architecture in Linux kernel

Operating modes

64-bit mode	:	CS register value == 0x33
32-bit mode	:	CS register value == 0x23

Several methods of system call invocation

int 0x80	:	Legacy 32-bit
sysenter	:	Fast 32-bit
syscall	:	64-bit

Surprise: 64-bit processes can invoke both 64-bit and 32-bit system calls.

Linux API provides

- The system call number
- The value of CS register
- The value of RIP register

Linux system call tracers on x86-64 architecture

Legacy method of obtaining system call information

- Fetch the system call number (PTRACE_PEEKUSER ORIG_RAX)
- Fetch the value of CS register (PTRACE_PEEKUSER CS)
- **Guess the system call bitness by the value of CS register**
- Determine the system call by its number and bitness
- Fetch the system call arguments accordingly

Traditional method of obtaining system call information

- Fetch the whole set of registers (PTRACE_GETREGSET NT_PRSTATUS), the return value is decided by the value of CS register
- **Guess the system call bitness by the return value**
- Determine the system call by its number and bitness
- Interpret other registers as the system call arguments accordingly

Process bitness does not match syscall bitness

Example based on Debian bug report #459820 submitted in 2008

```
#include <stdio.h>
#include <unistd.h>
int main() {
    setlinebuf(stdout);
    puts("------------");
    __asm__("movl $2, %eax; int $0x80");
    printf("[I am %d]\n", getpid());
    return 0;
}
```

Regular invocation: ./debbug459820

```
------------
[I am 23450]
[I am 23451]
```

Process bitness does not match syscall bitness

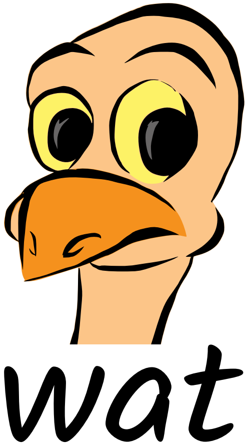
Invocation under strace

```
$ strace -f ./debbug459820 > /dev/null
...
write(1, "------------\n", 13) = 13
strace: Process 23451 attached
open(0x1, O_RDONLY|O_CREAT|O_TRUNC
|O_DSYNC|O_DIRECT|O_NOATIME|O_CLOEXEC
|O_PATH|O_TMPFILE|0x1000020, 0134300)
= 23451
...
```

Process bitness does not match syscall bitness

Invocation under strace

```
$ strace -f ./debbug459820 > /dev/null
...
write(1, "------------\n", 13) = 13
strace: Process 23451 attached
open(0x1, O_RDONLY|O_CREAT|O_TRUNC
|O_DSYNC|O_DIRECT|O_NOATIME|O_CLOEXEC
|O_PATH|O_TMPFILE|0x1000020, 0134300)
= 23451
...
```



Process bitness does not match syscall bitness

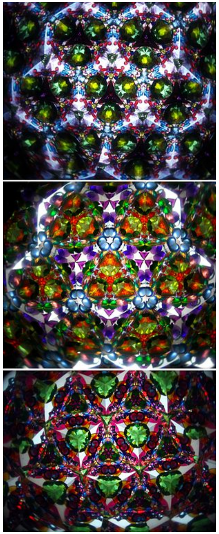
$ strace -f -z ./debbug459820 > /dev/null

```
write(1, "-------------\n", 13)              = 13
strace: Process 23451 attached
open(0x1, O_RDONLY|O_CREAT|O_TRUNC|O_DSYNC|O_DIRECT
|O_NOATIME|O_CLOEXEC|O_PATH|O_TMPFILE|0x1000020,
0134300) = 23451
[pid 23450] getpid()                        = 23450
[pid 23451] getpid()                        = 23451
[pid 23450] write(1, "[I am 23450]\n", 13) = 13
[pid 23451] write(1, "[I am 23451]\n", 13) = 13
[pid 23450] +++ exited with 0 +++
+++ exited with 0 +++
```

Process bitness does not match syscall bitness: problem

for i in ‘seq 0 9‘; do strace -qq -esignal=none -eopen ./debbug459820 >/dev/null; done

```
open(0x1, O_RDONLY|O_CREAT|O_EXCL|O_TRUNC|O_SYNC|O_DIRECT|O_LARGEFILE|
          O_NOFOLLOW|O_CLOEXEC|0x4f000008, 0151330) = 15565
open(0x1, O_RDONLY|O_EXCL|O_NOCTTY|O_APPEND|O_NONBLOCK|O_DSYNC|O_TMPFILE|
          FASYNC|0x57800008, 036630) = 15570
open(0x1, O_RDONLY|O_EXCL|O_NOCTTY|O_APPEND|O_SYNC|O_LARGEFILE|O_NOATIME|
          O_PATH|O_DIRECTORY|FASYNC|0x8e800038) = 15575
open(0x1, O_RDONLY|O_CREAT|O_EXCL|O_APPEND|O_DSYNC|O_DIRECT|O_NOFOLLOW|
          O_PATH|O_DIRECTORY|FASYNC|0xe2800018, 072350) = 15580
open(0x1, O_RDONLY|O_CREAT|O_NOCTTY|O_SYNC|O_NOFOLLOW|O_CLOEXEC|FASYNC|
          0xcf800038, 030610) = 15585
open(0x1, O_RDONLY|O_TRUNC|O_NOFOLLOW|O_CLOEXEC|O_DIRECTORY|FASYNC|
          0x11800008) = 15590
open(0x1, O_RDONLY|O_CREAT|O_EXCL|O_NOCTTY|__O_SYNC|O_LARGEFILE|O_NOATIME|
          O_CLOEXEC|O_PATH|O_TMPFILE|FASYNC|0x43000038, 0121010) = 15595
open(0x1, O_RDONLY|O_EXCL|O_NONBLOCK|__O_SYNC|O_DIRECT|O_CLOEXEC|O_PATH|
          __O_TMPFILE|FASYNC|0x3a800038, 064310) = 15600
open(0x1, O_RDONLY|O_CREAT|O_EXCL|O_NOCTTY|O_NONBLOCK|O_DSYNC|O_DIRECT|
          O_LARGEFILE|O_DIRECTORY|0x47800028, 0154770) = 15610
open(0x1, O_RDONLY|O_NOFOLLOW|O_CLOEXEC|O_PATH|FASYNC|0x5e000008) = 15605
```



PTRACE_GET_SYSCALL_INFO support

Linux >= v5.3-rc1

```
$ git log -i -E --author=altlinux.org \
  --grep='ptrace|syscall_a|elf-em|selftests' \
  v4.20-rc2..v5.3-rc1
```

- 29 commits, 47 files changed, 703 insertions, 125 deletions
- 2 authors: Elvira Khabirova, Dmitry Levin
- 22 persons added their Acked-by/Reviewed-by/Signed-off-by
- 07.11.2018: first RFC patch submitted
- 12.11.2018: first patch committed
- 13.12.2018: API finalized
- 17.07.2019: last patch committed
- Implements PTRACE_GET_SYSCALL_INFO on those 19 architectures that enable CONFIG_HAVE_ARCH_TRACEHOOK

PTRACE_GET_SYSCALL_INFO support

Linux PTRACE_GET_SYSCALL_INFO API

```
struct ptrace_syscall_info {
  __u8 op;                                               /* Type of system call stop */
  __aligned_u32 arch;                          /* AUDIT_ARCH_* value; see seccomp(2) */
  __u64 instruction_pointer;                           /* CPU instruction pointer */
  __u64 stack_pointer;                                       /* CPU stack pointer */
  union {
    struct {                                    /* op == PTRACE_SYSCALL_INFO_ENTRY */
      __u64 nr;                                                 /* Syscall number */
      __u64 args[6];                                         /* Syscall arguments */
    } entry;
    struct {                                     /* op == PTRACE_SYSCALL_INFO_EXIT */
      __s64 rval;                                         /* Syscall return value */
      __u8 is_error;                           /* Does rval contain an error value? */
    } exit;
    struct {                                  /* op == PTRACE_SYSCALL_INFO_SECCOMP */
      __u64 nr;                                                 /* Syscall number */
      __u64 args[6];                                         /* Syscall arguments */
      __u32 ret_data;              /* SECCOMP_RET_DATA portion of SECCOMP_RET_TRACE */
    } seccomp;
  };
};
```

PTRACE_GET_SYSCALL_INFO support

strace >= v4.26, linux >= v5.3-rc1

Invocation under strace: strace -f -z ./debbug459820 > /dev/null

```
...
write(1, "------------\n", 13)            = 13
strace: [ Process PID=23450 runs in 32 bit mode. ]
strace: Process 23451 attached
fork()                                    = 23451
strace: [ Process PID=23450 runs in 64 bit mode. ]
[pid 23450] getpid()                      = 23450
strace: [ Process PID=23451 runs in 64 bit mode. ]
[pid 23451] getpid()                      = 23451
[pid 23450] write(1, "[I am 23450]\n", 13) = 13
[pid 23451] write(1, "[I am 23451]\n", 13) = 13
[pid 23450] +++ exited with 0 +++
+++ exited with 0 +++
```

System call return status filtering: **-e status=***set*

Introduced in v5.2 (July 2019)

Print only system calls with the specified return status.
set can include the following elements:

successful	: returned without an error code, alias to **-z**
failed	: returned with an error code, alias to **-Z**
unfinished	: did not return
detached	: detached before return
unavailable	: returned but failed to fetch the error status

The default is **-e status=all**.

System call return status filtering: -z, -Z

env -i LD_LIBRARY_PATH=/lib64 strace -z -e%file /bin/cat < /dev/null

```
execve("/bin/cat", ["/bin/cat"], 0x7ffc2b781ca0 /* 1 var */) = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
+++ exited with 0 +++
```

env -i LD_LIBRARY_PATH=/lib64 strace -Z -e%file /bin/cat </dev/null

```
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No suc
stat("/lib64/tls/x86_64/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
stat("/lib64/tls/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
stat("/lib64/tls/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or dire
stat("/lib64/tls", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such fi
stat("/lib64/x86_64/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or d
stat("/lib64/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or d
stat("/lib64/x86_64", 0x7ffdcc6a0c20) = -1 ENOENT (No such file or directory)
+++ exited with 0 +++
```

System call return status filtering: -z aggregation

```
strace -o log -f -e signal=none -e trace=execve,nanosleep \
sh -c 'sleep 0.1 & sleep 0.2 & sleep 0.3' && cat log
```

```
13475 execve("/bin/sh", ["sh", "-c", "sleep 0.1 & sleep 0.2 & sleep 0."...],
      0x5631be4f87a8 /* 42 vars */) = 0
13476 execve("/bin/sleep", ["sleep", "0.1"], 0xe4c4f0 /* 33 vars */ <unfinished ...>
13477 execve("/bin/sleep", ["sleep", "0.2"], 0xe4c4f0 /* 33 vars */ <unfinished ...>
13478 execve("/bin/sleep", ["sleep", "0.3"], 0xe4c4f0 /* 33 vars */ <unfinished ...>
13476 <... execve resumed>)             = 0
13477 <... execve resumed>)             = 0
13478 <... execve resumed>)             = 0
13476 nanosleep(tv_sec=0, tv_nsec=100000000,  <unfinished ...>
13477 nanosleep(tv_sec=0, tv_nsec=200000000,  <unfinished ...>
13478 nanosleep(tv_sec=0, tv_nsec=300000000,  <unfinished ...>
13476 <... nanosleep resumed>NULL)       = 0
13476 +++ exited with 0 +++
13477 <... nanosleep resumed>NULL)       = 0
13477 +++ exited with 0 +++
13478 <... nanosleep resumed>NULL)       = 0
13478 +++ exited with 0 +++
13475 +++ exited with 0 +++
```

System call return status filtering: -z aggregation

```
strace -o log -z -f -e signal=none -e trace=execve,nanosleep \
sh -c 'sleep 0.1 & sleep 0.2 & sleep 0.3' && cat log
```

```
13475 execve("/bin/sh", ["sh", "-c", "sleep 0.1 & sleep 0.2 & sleep 0."...],
      0x5631be4f87a8 /* 42 vars */) = 0
13476 execve("/bin/sleep", ["sleep", "0.1"], 0xe4c4f0 /* 33 vars */) = 0
13477 execve("/bin/sleep", ["sleep", "0.2"], 0xe4c4f0 /* 33 vars */) = 0
13478 execve("/bin/sleep", ["sleep", "0.3"], 0xe4c4f0 /* 33 vars */) = 0
13476 nanosleep(tv_sec=0, tv_nsec=100000000, NULL) = 0
13476 +++ exited with 0 +++
13477 nanosleep(tv_sec=0, tv_nsec=200000000, NULL) = 0
13477 +++ exited with 0 +++
13478 nanosleep(tv_sec=0, tv_nsec=300000000, NULL) = 0
13478 +++ exited with 0 +++
13475 +++ exited with 0 +++
```

System call return status filtering: -Z

Introduced in v5.2 (July 2019)

Trace only those system calls that returned with an error code:
-Z, **-e status=failed**

eu-elflint no-such-file1 no-such-file2

```
eu-elflint: cannot open input file: No such file or directory
eu-elflint: cannot open input file: No such file or directory
```

strace -qq -Z -efile eu-elflint no-such-file1 no-such-file2

```
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "no-such-file1", O_RDONLY) = -1 ENOENT (No such file or directory)
eu-elflint: cannot open input file: No such file or directory
openat(AT_FDCWD, "no-such-file2", O_RDONLY) = -1 ENOENT (No such file or directory)
eu-elflint: cannot open input file: No such file or directory
```

Fixed in elfutils-0.178.

Seccomp-assisted system call filtering

Introduced in v5.3 (September 2019)

- Automatically generates and attaches a BPF program to filter system calls
- Makes execution of untraced system calls two orders of magnitude faster

The infamous dd example

```
$ dd if=/dev/zero of=/dev/null bs=1 count=1M 2>&1 | grep -v records
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.668768 s, 1.6 MB/s

$ strace --seccomp-bpf -f -qq -e signal=none -e trace=fchdir \
  dd if=/dev/zero of=/dev/null bs=1 count=1M 2>&1 | grep -v records
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.736511 s, 1.4 MB/s

$ strace -f -qq -e signal=none -e trace=fchdir \
  dd if=/dev/zero of=/dev/null bs=1 count=1M 2>&1 | grep -v records
1048576 bytes (1.0 MB, 1.0 MiB) copied, 28.0445 s, 37.4 kB/s
```

Comparative slowdown

$0.736511/0.668768 \approx 1.10$, $28.0445/0.736511 \approx 38$, $28.0445/0.668768 \approx 42$

Long options

1991 . . . 2019: short options only

a:Ab:cCdDe:E:fFhiI:ko:O:p:P:qrs:S:tTu:vVwxX:yzZ

Introduced in v5.3 (September 2019)

- `--help`: alias to `-h`
- `--version`: alias to `-V`
- `--seccomp-bpf`: seccomp-assisted system call filtering

Introduced in v5.5 (February 2020)

- `--debug`: alias to `-d`
- . . .

Stack of function calls: -**k** option

strace -qq -P /dev/full cat /dev/null > /dev/full

```
fstat(1, st_mode=S_IFCHR|0666, st_rdev=makedev(1, 7), ...) = 0
close(1)                                       = 0
```

strace -**k** -qq -P /dev/full cat /dev/null > /dev/full

```
fstat(1, st_mode=S_IFCHR|0666, st_rdev=makedev(1, 7), ...) = 0
 > /lib64/libc-2.27.so(__fxstat64+0x13) [0xe79c3]
 > /bin/cat(main+0x1b3) [0x4017e3]
 > /lib64/libc-2.27.so(__libc_start_main+0xe6) [0x21bd6]
 > /bin/cat(_start+0x29) [0x402179]
close(1)                                       = 0
 > /lib64/libc-2.27.so(__close_nocancel+0x7) [0xe8b47]
 > /lib64/libc-2.27.so(_IO_file_close_it@@GLIBC_2.2.5+0x67) [0x79fd7]
 > /lib64/libc-2.27.so(fclose@@GLIBC_2.2.5+0x136) [0x6d376]
 > /bin/cat(close_stream+0x19) [0x404ce9]
 > /bin/cat(close_stdout+0x11) [0x402691]
 > /lib64/libc-2.27.so(__run_exit_handlers+0x170) [0x379c0]
 > /lib64/libc-2.27.so(exit+0x19) [0x37ab9]
 > /lib64/libc-2.27.so(__libc_start_main+0xed) [0x21bdd]
 > /bin/cat(_start+0x29) [0x402179]
```

Running as a detached grandchild: -D option

foreground strace

```
$ echo $$ && strace -e none sh -c 'echo $PPID'
1234
23456
+++ exited with 0 +++
$ echo $$ && strace -e none sh -c 'echo $PPID'
1234
23459
+++ exited with 0 +++
```

background strace

```
$ echo $$ && strace -D -e none sh -c 'echo $PPID'
1234
1234
+++ exited with 0 +++
```

Format of named constants and flags: -X option

strace -e /open cat /dev/null

```
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/dev/null", O_RDONLY) = 3
+++ exited with 0 +++
```

strace -**X** **verbose** -e /open cat /dev/null

```
openat(-100 /* AT_FDCWD */, "/etc/ld.so.cache",
       0x80000 /* O_RDONLY|O_CLOEXEC */) = 3
openat(-100 /* AT_FDCWD */, "/lib64/libc.so.6",
       0x80000 /* O_RDONLY|O_CLOEXEC */) = 3
openat(-100 /* AT_FDCWD */, "/dev/null", 0 /* O_RDONLY */) = 3
+++ exited with 0 +++
```

strace -**X** **raw** -e /open cat /dev/null

```
openat(-100, "/etc/ld.so.cache", 0x80000) = 3
openat(-100, "/lib64/libc.so.6", 0x80000) = 3
openat(-100, "/dev/null", 0)              = 3
+++ exited with 0 +++
```

New system calls

Introduced in v5.3 (September 2019)

pidfd_open, clone3

Introduced in v5.2 (July 2019)

open_tree, move_mount, fsopen, fsconfig, fsmount, fspick

Introduced in v5.1 (May 2019)

clock_gettime64, clock_settime64, clock_adjtime64, clock_getres_time64, clock_nanosleep_time64, timer_gettime64, timer_settime64, timerfd_gettime64, timerfd_settime64, utimensat_time64, pselect6_time64, ppoll_time64, io_pgetevents_time64, recvmmsg_time64, mq_timedsend_time64, mq_timedreceive_time64, semtimedop_time64, rt_sigtimedwait_time64, futex_time64, sched_rr_get_interval_time64, pidfd_send_signal, io_uring_setup, io_uring_enter, io_uring_register

Netlink socket parsers

Currently supported netlink protocols

- NETLINK_AUDIT
- NETLINK_CRYPTO
- NETLINK_KOBJECT_UEVENT
- NETLINK_NETFILTER
- NETLINK_ROUTE
- NETLINK_SELINUX
- NETLINK_SOCK_DIAG
- NETLINK_XFRM
- NETLINK_GENERIC

NETLINK_ROUTE: ip route list table all

```
broadcast 127.0.0.0 dev lo table local proto kernel scope link src 127.0.0.1
local 127.0.0.0/8 dev lo table local proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo table local proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo table local proto kernel scope link src 127.0.0.1
```

Netlink socket parsers: NETLINK_ROUTE

strace -e trace=sendto,recvmsg ip route list

```
sendto(3, {{len=40, type=RTM_GETROUTE, flags=NLM_F_REQUEST|NLM_F_DUMP, seq=1357924680, pid=0}, {rtm_family=AF_UNSPEC,
rtm_dst_len=0, rtm_src_len=0, rtm_tos=0, rtm_table=RT_TABLE_UNSPEC, rtm_protocol=RTPROT_UNSPEC, rtm_scope=RT_SCOPE_UNIVERSE,
rtm_type=RTN_UNSPEC, rtm_flags=0}, {nla_len=0, nla_type=RTA_UNSPEC}}, 40, 0, NULL, 0) = 40

recvmsg(3, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12, msg_iov=[{iov_base=[ {{len=60,
type=RTM_NEWROUTE, flags=NLM_F_MULTI, seq=1357924680, pid=12345}, {rtm_family=AF_INET, rtm_dst_len=32, rtm_src_len=0,
rtm_tos=0, rtm_table=RT_TABLE_LOCAL, rtm_protocol=RTPROT_KERNEL, rtm_scope=RT_SCOPE_LINK, rtm_type=RTN_BROADCAST,
rtm_flags=0}, [{{nla_len=8, nla_type=RTA_TABLE}, RT_TABLE_LOCAL}, {{nla_len=8, nla_type=RTA_DST}, inet_addr("127.0.0.0")},
{{nla_len=8, nla_type=RTA_PREFSRC}, inet_addr("127.0.0.1")}, {{nla_len=8, nla_type=RTA_OIF}, if_nametoindex("lo")}]},
{{len=60, type=RTM_NEWROUTE, flags=NLM_F_MULTI, seq=1357924680, pid=12345}, {rtm_family=AF_INET, rtm_dst_len=8,
rtm_src_len=0, rtm_tos=0, rtm_table=RT_TABLE_LOCAL, rtm_protocol=RTPROT_KERNEL, rtm_scope=RT_SCOPE_HOST,
rtm_type=RTN_LOCAL, rtm_flags=0}, [{{nla_len=8, nla_type=RTA_TABLE}, RT_TABLE_LOCAL}, {{nla_len=8, nla_type=RTA_DST},
inet_addr("127.0.0.0")}, {{nla_len=8, nla_type=RTA_PREFSRC}, inet_addr("127.0.0.1")}, {{nla_len=8, nla_type=RTA_OIF},
if_nametoindex("lo")}]}, {{len=60, type=RTM_NEWROUTE, flags=NLM_F_MULTI, seq=1357924680, pid=12345}, {rtm_family=AF_INET,
rtm_dst_len=32, rtm_src_len=0, rtm_tos=0, rtm_table=RT_TABLE_LOCAL, rtm_protocol=RTPROT_KERNEL, rtm_scope=RT_SCOPE_HOST,
rtm_type=RTN_LOCAL, rtm_flags=0}, [{{nla_len=8, nla_type=RTA_TABLE}, RT_TABLE_LOCAL}, {{nla_len=8, nla_type=RTA_DST},
inet_addr("127.0.0.1")}, {{nla_len=8, nla_type=RTA_PREFSRC}, inet_addr("127.0.0.1")}, {{nla_len=8, nla_type=RTA_OIF},
if_nametoindex("lo")}]}, {{len=60, type=RTM_NEWROUTE, flags=NLM_F_MULTI, seq=1357924680, pid=12345}, {rtm_family=AF_INET,
rtm_dst_len=32, rtm_src_len=0, rtm_tos=0, rtm_table=RT_TABLE_LOCAL, rtm_protocol=RTPROT_KERNEL, rtm_scope=RT_SCOPE_LINK,
rtm_type=RTN_BROADCAST, rtm_flags=0}, [{{nla_len=8, nla_type=RTA_TABLE}, RT_TABLE_LOCAL}, {{nla_len=8, nla_type=RTA_DST},
inet_addr("127.255.255.255")}, {{nla_len=8, nla_type=RTA_PREFSRC}, inet_addr("127.0.0.1")}, {{nla_len=8, nla_type=RTA_OIF},
if_nametoindex("lo")}]} ], iov_len=32768}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 240
...
```

Copyleft license

1991 ... 2018: permissive license

strace was released under a Berkeley-style license
at the request of Paul Kranenburg.

Since v4.26∼52 (December 2018): copyleft license

- test suite: GNU General Public License v2+
- the rest of strace: GNU Lesser General Public License v2.1+

The first major strace feature implemented after this change is PTRACE_GET_SYSCALL_INFO API support.

Questions?

homepage

https://strace.io

strace.git, strace-talks.git

https://github.com/strace/strace.git
https://gitlab.com/strace/strace.git

https://github.com/strace/strace-talks.git
https://gitlab.com/strace/strace-talks.git

mailing list

strace-devel@lists.strace.io

IRC channel

#strace@freenode