Truly structured output in strace

January 29, 2017

`strace - trace system calls and signals`

— `man 1 strace`

```
$ strace cat
execve("/bin/cat", ["cat"], [/* 40 vars */]) = 0
brk(0)                                  = 0x155e000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691ab4000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=135526, ...}) = 0
mmap(NULL, 135526, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8691a92000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1729984, ...}) = 0
mmap(NULL, 3836448, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f86914ed000
mprotect(0x7f869168c000, 2097152, PROT_NONE) = 0
mmap(0x7f869188c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19f000) = 0x7f869188
mmap(0x7f8691892000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8691892000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a91000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a90000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a8f000
arch_prctl(ARCH_SET_FS, 0x7f8691a90700) = 0
mprotect(0x7f869188c000, 16384, PROT_READ) = 0
mprotect(0x60b000, 4096, PROT_READ)     = 0
mprotect(0x7f8691ab6000, 4096, PROT_READ) = 0
munmap(0x7f8691a92000, 135526)          = 0
brk(0)                                  = 0x155e000
brk(0x157f000)                          = 0x157f000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2855152, ...}) = 0
mmap(NULL, 2855152, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8691233000
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
fadvise64(0, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8691a92000
read(0, "", 131072)                     = 0
munmap(0x7fb604be3000, 139264)          = 0
```

```c
int getitimer(int which, struct itimerval *curr_value);
```

```c
SYS_FUNC(getitimer)
{
    if (entering(tcp)) {
        printxval(itimer_which, tcp->u_arg[0], "ITIMER_???");
        tprints(", ");
    } else {
        print_itimerval(tcp, tcp->u_arg[1]);
    }
    return 0;
}
```

```
case Q_XGETQSTAT:
{
struct xfs_dqstats dq;

if (entering(tcp))
return 0;
if (umove_or_printaddr(tcp, data, &dq))
break;
tprintf("{version=%d, ", dq.qs_version);
if (abbrev(tcp)) {
tprints("...}");
break;
}
tprints("flags=");
printflags(xfs_quota_flags,
   dq.qs_flags, "XFS_QUOTA_???");
tprintf(", incoredqs=%u, ", dq.qs_incoredqs);
tprintf("u_ino=%" PRIu64 ", ", dq.qs_uquota.qfs_ino);
tprintf("u_nblks=%" PRIu64 ", ", dq.qs_uquota.qfs_nblks);
tprintf("u_nextents=%u, ", dq.qs_uquota.qfs_nextents);
tprintf("g_ino=%" PRIu64 ", ", dq.qs_gquota.qfs_ino);
tprintf("g_nblks=%" PRIu64 ", ", dq.qs_gquota.qfs_nblks);
tprintf("g_nextents=%u, ", dq.qs_gquota.qfs_nextents);
tprintf("btimelimit=%d, ", dq.qs_btimelimit);
tprintf("itimelimit=%d, ", dq.qs_itimelimit);
tprintf("rtbtimelimit=%d, ", dq.qs_rtbtimelimit);
tprintf("bwarnlimit=%u, ", dq.qs_bwarnlimit);
tprintf("iwarnlimit=%u}", dq.qs_iwarnlimit);
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5

sendto(4<TCP:[127.0.0.1:34234->127.0.0.1:47663]>, "text", 4,
MSG_DONTROUTE|MSG_DONTWAIT, NULL, 0) = 4
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5

sendto(4<TCP:[127.0.0.1:34234->127.0.0.1:47663]>, "text", 4,
MSG_DONTROUTE|MSG_DONTWAIT, NULL, 0) = 4

listen(3<TCP:[4490622]>, 1) = 0
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5

sendto(4<TCP:[127.0.0.1:34234->127.0.0.1:47663]>, "text", 4,
MSG_DONTROUTE|MSG_DONTWAIT, NULL, 0) = 4

listen(3<TCP:[4490622]>, 1) = 0

setitimer(ITIMER_REAL, {it_interval={0, 222222}, it_value={0, 111111}},
NULL) = 0
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5

sendto(4<TCP:[127.0.0.1:34234->127.0.0.1:47663]>, "text", 4,
MSG_DONTROUTE|MSG_DONTWAIT, NULL, 0) = 4

listen(3<TCP:[4490622]>, 1) = 0

setitimer(ITIMER_REAL, {it_interval={0, 222222}, it_value={0, 111111}},
NULL) = 0

rt_sigprocmask(SIG_SETMASK, NULL, [HUP INT QUIT ALRM TERM], 8) = 0
```

```
accept4(3, {sa_family=AF_UNIX, sun_path="accept4.socket.connect"},
[110->25], SOCK_NONBLOCK|SOCK_CLOEXEC) = 5

sendto(4<TCP:[127.0.0.1:34234->127.0.0.1:47663]>, "text", 4,
MSG_DONTROUTE|MSG_DONTWAIT, NULL, 0) = 4

listen(3<TCP:[4490622]>, 1) = 0

setitimer(ITIMER_REAL, {it_interval={0, 222222}, it_value={0, 111111}},
NULL) = 0

rt_sigprocmask(SIG_SETMASK, NULL, [HUP INT QUIT ALRM TERM], 8) = 0

execve("./execve", ["./execve"], [/* 41 vars */]) = 0
```

```
global re_extract_unfinished
re_extract_unfinished \
    = re.compile(r"\s*(\d+\.\d+ .*) <unfinished \.\.\.>$")

global re_extract_resumed
re_extract_resumed \
    = re.compile(r"\s*(\d+\.\d+) <\.\.\. [\a-zA-Z\d]+ resumed>(.*)$")

global re_extract_signal
re_extract_signal \
    = re.compile(r"\s*(\d+\.\d+) --- (\w+) \(([\w ]+)\) @ (\d)+ \((\d+)\) ---$")

global re_extract_arguments_and_return_value_none
re_extract_arguments_and_return_value_none \
    = re.compile(r"\((.*)\)[ \t]*= (\?)$")

global re_extract_arguments_and_return_value_ok
re_extract_arguments_and_return_value_ok \
    = re.compile(r"\((.*)\)[ \t]*= (-?\d+)$")

global re_extract_arguments_and_return_value_ok_hex
re_extract_arguments_and_return_value_ok_hex \
    = re.compile(r"\((.*)\)[ \t]*= (-?0[xX][a-fA-F\d]+)$")

global re_extract_arguments_and_return_value_error
re_extract_arguments_and_return_value_error \
    = re.compile(r"\((.*)\)[ \t]*= (-?\d+) (\w+) \([\w ]+\)$")
```

```
# 16:08:17.082102 libpagemanager.so.1->mdb_env_create(0x6469c8, 0x7ff1e7c0d250,
# 0x646a48, 0x7ff1e7e41ea0) = 0 <0.000181>

normal= re.compile ('\s*%s +(?:%s\-\>)?%s\(%s\= %s %s' % (timestamp_parser,
    caller_parser, funcname_parser, params_parser, result_parser, time_parser))

# 16:08:17.192471 libDocumentAccess-Mh.so.1->MF_DeleteCollection(0x6490a8, 55,
# 0x6490a8, 0x7ff1e83c44e0 <unfinished ...>

unfinished= re.compile ('\s*%s +(?:%s\-\>)?%s\(%s\<unfinished \.\.\.\>' %
    (timestamp_parser, caller_parser, funcname_parser, params_parser))

# 16:08:17.203365 <... MF_DeleteCollection resumed> ) = 0x7ea000000000 <0.001550>

resumed= re.compile ('\s*%s +\<\.\.\. %s resumed\>.*\= %s %s' % (timestamp_parser,
funcname_parser, result_parser, time_parser))

# 11:44:55.470482  libpagemanager.so.1->mdb_txn_begin(0x646a80, 0, 0,
# 0x7fff6d7769f8 <no return ...>

# the point of...
no_return= re.compile ('\s*%s +(?:%s\-\>)?%s\(%s\<no return \.\.\.\>' %
    (timestamp_parser, caller_parser, funcname_parser, params_parser))

# anything, really, most likely
# 11:46:38.322997 +++ killed by SIGTERM +++
just_time= re.compile ('\s*%s' % timestamp_parser)
```

```
} elsif ($cmd eq $STAT) {
    if ($debug > 0) {
        printf(STRACE_LOG "FOUND A STAT!!!\n");
    }
    $CmdCounter{"stat"}++;
    $temp = floor($sec - $BeginTime) + 1;
    # Time in second referenced to beginning time
    $IOPS_Total[$temp]++;

    # Increment total IO time and command counters
    $IOTimeSum = $IOTimeSum + $elapsed_time;
    $IOTime_count++;
} elsif ($cmd eq $FSTAT) {
    if ($debug > 0) {
        printf(STRACE_LOG "FOUND A FSTAT!!!\n");
    }
    $CmdCounter{"fstat"}++;
    $temp = floor($sec - $BeginTime) + 1;
    # Time in second referenced to beginning time
    $IOPS_Total[$temp]++;

    # Increment total IO time and command counters
    $IOTimeSum = $IOTimeSum + $elapsed_time;
    $IOTime_count++;
} elsif ($cmd eq $STAT64) {
    if ($debug > 0) {
        printf(STRACE_LOG "FOUND A STAT64!!!\n");
    }
```

```
/^[1-9][0-9]*[[:space:]]+.* <unfinished \.\.\.>$/ {
    pid = $1;
    sub(" <unfinished \\.\\.\\.>$", "");
    trace[pid] = $0;
    next;
}

/^[1-9][0-9]*[[:space:]]+<\.\.\. [[:alnum:]_]+ resumed>/ {
    pid = $1;
    if (trace[pid] != "")
        sub("^[1-9][0-9]*[[:space:]]+<\\.\\.\\. [[:alnum:]_]+ resumed>", trace[pid]
}

match($0, /^[1-9][0-9]*[[:space:]]+([[:alnum:]_]+at)\([^",]+, "(\/[^"]*)", .*\) +=
    delete trace[$1];
    output(ary[1], ary[2]);
    next;
}

match($0, /^[1-9][0-9]*[[:space:]]+([[:alnum:]_]+)\("(\/[^"]*)", .*\) += [[:alnum:]
    delete trace[$1];
    output(ary[1], ary[2]);
    next;
}

/^[1-9][0-9]*[[:space:]]+/ { delete trace[$1]; }
```

```
$ strace -etrace=pwritev -ewrite=1 -s2 ./preadv-pwritev
...
pwritev(1, [{iov_base="01"..., iov_len=3}, {iov_base="34"..., iov_len=5}, ...],
3, 0) = 15
 * 3 bytes in buffer 0
 | 00000  30 31 32                                          012             |
 * 5 bytes in buffer 1
 | 00000  33 34 35 36 37                                    34567           |
 * 7 bytes in buffer 2
 | 00000  38 39 61 62 63 64 65                              89abcde         |
```

```
$ strace -c sync
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.012000       12000         1           sync
  0.00    0.000000           0         1           read
  0.00    0.000000           0         3           open
  0.00    0.000000           0         5           close
  0.00    0.000000           0         3           fstat
  0.00    0.000000           0         7           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         3         3 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.012000                    33         3 total
```

Approach

- Separate parsing and printing (duh)
  - Syscall decoders should store all the information in internal representation
  - Printers should output internal representation to the desired format

Approach

- Separate parsing and printing (duh)
  - Syscall decoders should store all the information in internal representation
  - Printers should output internal representation to the desired format
- Parsers should have only one job and be as simple as possible

Approach

- Separate parsing and printing (duh)
  - Syscall decoders should store all the information in internal representation
  - Printers should output internal representation to the desired format
- Parsers should have only one job and be as simple as possible
- Printers should be as local as possible

Approach

- Separate parsing and printing (duh)
  - Syscall decoders should store all the information in internal representation
  - Printers should output internal representation to the desired format
- Parsers should have only one job and be as simple as possible
- Printers should be as local as possible
- The middle layer is introduced, which backs up parsers in their workings, and calls appropriate formatter methods.

Implementation

- Currently, syscall information is stored in `struct tcb` (for Tracee Control Block) and it already has the information regarding syscall itself (its name, arguments, tc)
- It has been augmented with argument storage
- In addition, it now stores current "insertion state", which enables great simplification of syscall decoders
  - For example, it stores "current" argument index, which enables such constructs as "put current argument to argument storage as unsigned int"
  - It also has a notion of "insertion stack", enabling inserting data without providing insertion context (array or struct the data should be placed in) in every call, just like the old way
  - It frees decoders from taking special measures for handling parameters of `long long` type

Implementation

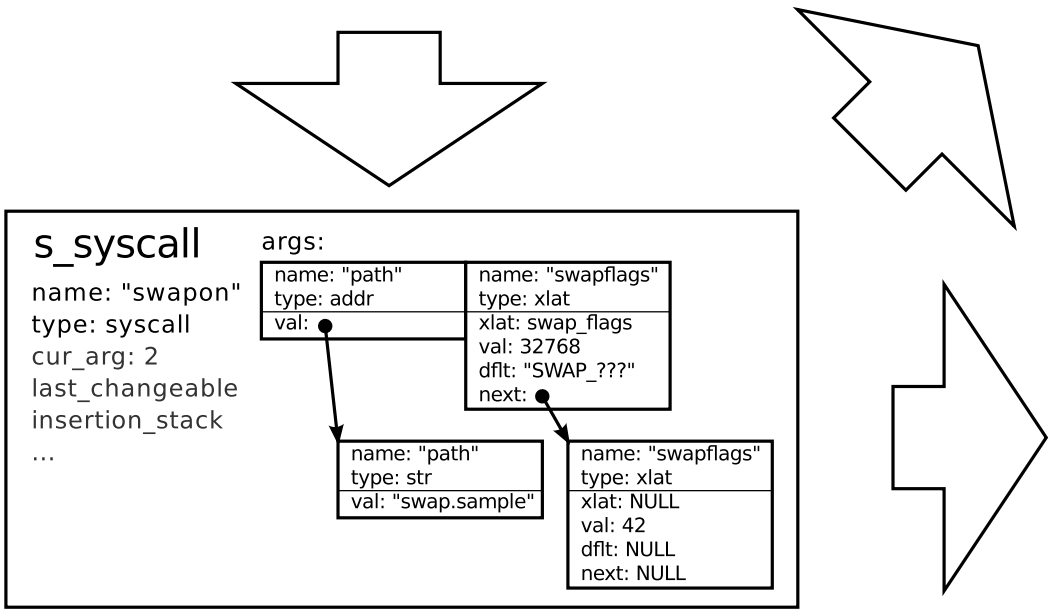
- A separate type is defined for each entity which has its own peculiarities in output formatting
  - Argument type has information about its size (`short`, `kernel long`, ...) and formatting (`uid`, `dirfd`, `pointer`, `signal number`, ...) encoded
  - It also specifies how it should be stored internally (pointers have additional field for storing data they point to, for example)
  - Each type has a separate set of calls for working with it, all calls (mostly) have common naming scheme and similar calling convention, though
- In additional to basic types, calls for printing common types (`struct timespec_t`, `struct siginfo`, `struct statfs`, ...) converted to sets of calls which are similar to calls for basic types
- Additional API calls are added for generalised handling of arrays (as a replacement of `print_array` and some ad-hoc implementations — `mincore`, netlink decoding)

Formatters

- Formatters are implemented as a structure containing pointers to functions which are called at various stages of tracee handling (syscall entering, syscall exiting, ptrace error, switching contexts, tracee termination, ...)
- Currently, only two formatters are implemented: legacy ("text") and JSON.

```
SYS_FUNC(swapon)
{
    s_push_path("path");
    s_push_xlat_flags_int("swapflags", swap_flags,
        NULL, SWAP_FLAG_PRIO_MASK,
        "SWAP_FLAG_???", NULL, false, 0);

    return RVAL_DECODED;
}
```



Legacy:

```
swapon("swap.sample", SWAP_FLAG_PREFER|42) =
    -1 EPERM (Operation not permitted)
```

JSON:

```
{
    "name": "swapon", "type": "syscall",
    "return": -1, "errno": 1, "errnostr": "EPERM",
    "retstring": "Operation not permitted",
    "args": [ {
            "name": "path", "type": "address",
            "addr": 4199168,
            "value": { "name": "path", "type": "str",
                    "value": "swap.sample" }
        }, {
            "name": "swapflags", "type": "xlat",
            "value": [
                { "default": false, "value": 32768,
                        "str": "SWAP_FLAG_PREFER" },
                { "default": false, "value": 42 }
] } ] }
```

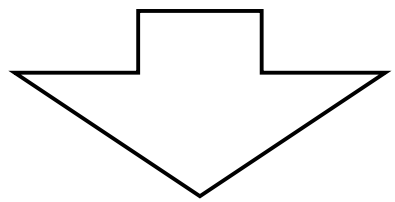
API overview

- `s_insert_<smth>` — insert something at current insertion point, `s_push_<smth>` — add something as a new argument, discard current insertion stack
  - Integer values: `s_insert_kld`, `s_push_hhu`
  - Basic types: `s_insert_uid`, `s_insert_signo`, `s_push_rlim64`
  - Strings: `s_insert_str`, `s_push_path`
  - Named values (xlat): `s_insert_xlat_int`, `s_push_flags_klong`, `s_insert_xlat_flags_64`, `s_append_xlat_long`
- Arrays and structs: `s_struct_enter`
- Handling of item by pointer/array of items of the specific type: `s_insert_addr_type`, `s_push_array_type`
- In/out arguments: `s_changeable`
- Misc: `s_push_empty`, `s_insert_ellipsis`

```
SYS_FUNC(swapon)
{
    unsigned int flags = tcp->u_arg[1];
    unsigned int prio = flags & SWAP_FLAG_PRIO_MASK;
    flags &= ~SWAP_FLAG_PRIO_MASK;

    printpath(tcp, tcp->u_arg[0]);
    tprints(", ");
    if (flags) {
        printflags(swap_flags, flags, "SWAP_FLAG_???");
        if (prio)
            tprintf("|%u", prio);
       } else {
           tprintf("%u", prio);
       }

    return RVAL_DECODED;
}
```



```
SYS_FUNC(swapon)
{
    s_push_path("path");
    s_push_xlat_flags_int("swapflags", swap_flags,
        NULL, SWAP_FLAG_PRIO_MASK,
        "SWAP_FLAG_???", NULL, false, 0);

    return RVAL_DECODED;
}
```

Full documentation: `https://github.com/lineprinter/strace/blob/structured/README-structured.md`

```
{
    "name": "getrandom",
    "type": "syscall",
    "args": [
        {
            "name": "buf",
            "type": "changeable",
            "entering_value": null,
            "exiting_value": {
                "name": "buf",
                "type": "address",
                "addr": 140722827922048,
                "value": {
                    "name": "buf",
                    "type": "str",
                    "value": "\\x26\\x4d\\x4e",
                    "size": 3,
                    "truncated": true
                }
            }
        },
        {
            "name": "count",
            "value": 3
        },
        {
            "name": "flags",
            "type": "xlat",
            "value": [
                {
                    "default": true,
                    "value": 0
                }
            ]
        }
    ],
    "return": 3
}
```

- less redundancy

- less redundancy
- arg name everywhere
- arg comments

- less redundancy
- arg name everywhere
- arg comments
- `-z`

- less redundancy
- arg name everywhere
- arg comments
- `-z`
- out & in-out args

- less redundancy
- arg name everywhere
- arg comments
- `-z`
- out & in-out args
- the struggle finally ends

Status and Plans

- About 40—50% done (API is pretty stable, about 70% of syscalls and 35% of printing routines are converted to structured)
- Effort is currently in mainline, mostly (additional tests, porting of output changes, fixes of errors in existing parsers)
- Once test coverage becomes pretty full, switch to structured would be considered pretty safe
- Open question is testing of JSON formatter and the JSON schema itself (first iteration is just to document existing one)

https://github.com/lineprinter/strace/tree/structured

Examples of comments in existing output

```
ioctl(-1, BTRFS_IOC_DEFRAG_RANGE, {start=0, len=18446744073709551615 /* UINT64_MAX */,
flags=BTRFS_DEFRAG_RANGE_COMPRESS|BTRFS_DEFRAG_RANGE_START_IO|0x4, extent_thresh=131072,
compress_type=0x3 /* BTRFS_COMPRESS_??? */}) = -1 EBADF (Bad file descriptor)

ioctl(-1, BTRFS_IOC_TREE_SEARCH_V2, {key={tree_id=5 /* BTRFS_FS_TREE_OBJECTID */,
min_objectid=6 /* BTRFS_ROOT_TREE_DIR_OBJECTID */, max_objectid=7 /* BTRFS_CSUM_TREE_OBJECTID */,
min_offset=1, max_offset=18446744073709551614, min_transid=1, max_transid=18446744073709551614,
min_type=0, max_type=4294967295 /* UINT32_MAX */, nr_items=10}, buf_size=4096}) =
-1 EBADF (Bad file descriptor)

ioctl(-1, DM_VERSION, {version=4.1.2, data_size=14, /* Incorrect data_size */ ...}) =
-1 EBADF (Bad file descriptor)

ioctl(-1, DM_TABLE_LOAD, {version=4.1.2, data_size=312, data_start=4294967288, dev=makedev(18, 52),
name="nnn", uuid="uuu", target_count=1234, flags=0, /* misplaced struct dm_target_spec */ ...}) =
-1 EBADF (Bad file descriptor)

perf_event_open({type=0x6 /* PERF_TYPE_??? */, size=PERF_ATTR_SIZE_VER3, config=0,
sample_freq=11357123448625731478, sample_type=0xdeadc0deda780000 /* PERF_SAMPLE_??? */,
read_format=PERF_FORMAT_TOTAL_TIME_ENABLED, disabled=0, inherit=1, pinned=1, exclusive=1,
exclusive_user=0, exclude_kernel=1, exclude_hv=0, exclude_idle=1, mmap=1, comm=1, freq=1,
inherit_stat=1, enable_on_exec=0, task=1, watermark=0, precise_ip=1 /* constant skid */, mmap_data=0,
sample_id_all=0, exclude_host=0, exclude_guest=1, exclude_callchain_kernel=1, exclude_callchain_user=0,
mmap2=1, comm_exec=1, use_clockid=0, context_switch=0, write_backward=0, __reserved_1=0xb5b4b3b2b
/* Bits 63..28 */, wakeup_events=3115890614, config1=0xc5c4c3c2c1c0bfbe, config2=0xcdcccbcac9c8c7c6,
sample_regs_user=0xdddcdbdad9d8d7d6}, -1, -1, 1,
PERF_FLAG_FD_NO_GROUP|PERF_FLAG_FD_OUTPUT|PERF_FLAG_PID_CGROUP|PERF_FLAG_FD_CLOEXEC) =
-1 EINVAL (Invalid argument)

execve("test.execve\nfilename", ["test.execve\nfilename", "first", "second", 0xffffffffffffffff,
0xfffffffffffffffe, 0xfffffffffffffffd, ???], [/* 5 vars, unterminated */]) =
-1 ENOENT (No such file or directory)
```