Contribution ID: **85** Type: **not specified**

What could be done in the kernel to make strace happy

Wednesday, 14 November 2018 14:00 (45 minutes)

What could be done in the kernel to make strace happy.

Being a traditional tool with a long history, strace has been making every effort to overcome various deficiencies in the kernel API. Unfortunately, some of these workarounds are fragile, and in some cases no workaround is possible. In this talk maintainers of strace will describe these deficiencies and propose extensions to the kernel API so that tools like strace could work in a more reliable way.

1

Problem: there is no kernel API to find out whether the tracee is entering or exiting syscall.

Current workarounds: strace does its best to sort out and track ptrace events, this works in most cases, but in case of strace attaching to a tracee being inside exec when its first syscall stop is syscall-exit-stop instead of syscall-enter-stop, the workaround is fragile, and in infamous case of int 0x80 on x86_64 there is no reliable workaround.

Proposed solution: extend the ptrace API with PTRACE_GET_SYSCALL_INFO request.

2

Problem: there is no kernel API to invoke wait4 syscall with changed signal mask.

Current workarounds: strace does its best to implement a race-free workaround, but it is way too complex and hard to maintain.

Proposed solution: add wait6 syscall which is wait4 with additional signal mask arguments, like pselect vs select and ppoll vs poll.

3

Problem: time precision provided by struct rusage is too low for strace -c nowadays.

Current workarounds: none.

Proposed solution: when adding wait6 syscall, change struct rusage argument to a different structure with fields of type struct timespec instead of struct timeval.

4

Problem: PID namespaces have been introduced without a proper kernel API to translate between tracer and tracee views of pids. This causes confusion among strace users, e.g. https://bugzilla.redhat.com/1035433

Current workarounds: none.

Proposed solution: add translate_pid syscall, e.g. https://lkml.org/lkml/2018/7/3/589

Problem: there are no consistent declarative syscall descriptions, this forces every user to reinvent its own wheel and catch up with the kernel.

Current workarounds: a lot of manual work has been done in strace to implement parsers of all syscalls. Some of these parsers are quite complex and hard to test. Other projects, e.g. syzkaller, implement their own representation of syscall ABI.

Proposed solution: provide declarative descriptions for all syscalls consistently.

I agree to abide by the anti-harassment policy

Yes

Primary author: LEVIN, Dmitry (BaseALT)

Co-authors: KHABIROVA, Elvira (BaseALT); SYROMYATNIKOV, Eugene (RedHat)

Presenters: LEVIN, Dmitry (BaseALT); KHABIROVA, Elvira (BaseALT); SYROMYATNIKOV, Eugene (RedHat)

Session Classification: LPC Main Track

Track Classification: Refereed talk