Contribution ID: **38** Type: **not specified**

Extensible Syscalls

Thursday, 27 August 2020 07:00 (45 minutes)

Most Linux syscall design conventions have been established through trial and error. One well-known example is the missing flag argument in a range of syscalls that triggered the addition of a revised version of theses syscalls. Nowadays, adding a flag argument to keep syscalls extensible is an accepted convention recorded in our kernel docs.

In this session we'd like to propose and discuss a few simple conventions that have proven useful over time and a few new ones that were just established recently with the addition of new in-kernel apis. Ideally these conventions would be added to the kernel docs and maintainers encouraged to use them as guidance when new syscalls are added.
We believe that these conventions can lead to a more consistent (and possibly more pleasant) uapi going forward making programming on Linux easier for userspace. They hopefully also prevent new syscalls running into various design pitfalls that have lead to quirky or cumbersome apis and (security) bugs.

Topics we'd like to discuss include the use of structs versioned by size in syscalls such as openat2(), sched_{set,get}_attr(), and clone3() and the associated api that we added last year, whether new syscalls should be allowed to use nested pointers in general and specifically with an eye on being conveniently filterable by seccomp, the convention to always use unsigned int as the type for register-based flag arguments intstead of the current potpourri of types, naming conventions when revised versions of syscalls are added, and - ideally a uniform way - how to test whether a syscall supports a given feature.

I agree to abide by the anti-harassment policy

I agree

Primary authors: BRAUNER, Christian (Canonical); SARAI, Aleksa (SUSE LLC)

Presenters: BRAUNER, Christian (Canonical); SARAI, Aleksa (SUSE LLC)

Session Classification: Kernel Summit

Track Classification: Kernel Summit