Botching Up IOCTLs

Daniel Vetter, Intel OTC
LCA Auckland 2015

overview

- basics: testcases, interface type, ...
- technicalities for proper ABI design
- special topics like resource handling, signals, time, ...

basics: interface type

- (generic) IOCTL or syscall?
- read/write/poll on an FD
- sysfs, configfs, debugfs, …
- extend existing subsystesm like perf

basics: real-world userspace

- tested, reviewed, ready for merging
- production code (corner cases, errors all handled)
- BUT: always merge kernel patches first

basics: testcases

- for everything
- FOR EVERYTHING
- focus on evil corner-cases

technicalities: struct ABI

- goal: no compat layer
- only use `__s32`, `__u32`, `__s64` and `__u64`
- if you have any 64 bit members: align/pad everything to 64 bit
- pointers are `__u64`
- `__attribute__((packed))` plus explicit padding when you screwed up

technicalities: input validation

- unchecked stack garbage breaks extendability
- unchecked evil input scores CVEs
- overflows (careful with variable-sized arrays)
- invalid combinations&values
- have testcases for everything

technicalities: flags

- have a flags parameter
- reject invalid flags with `-EINVAL`
- have a testcase
- specifically check for: invalid flag combinations, unused values in bitfields and the next available flag

technicalities: compatibility

- hide big things for 1-2 kernel releases
- flags, driver caps, userspace caps for opt-in, interface revisions
- remember: it's only a regression when you get a bug report

technicalities: endianess

- it's horrible
- but the world is mostly little-endian

resources

- attach everything to a `struct file`
- consider standard file types like dma-buf, fences, ...
- support `O_CLOEXEC`

resources: sharing

- private namespace ok when there's tons of objects
- but don't reinvent resource passing/sharing
- consider uniqueness requirements
- proper `fstat()` unfortunately needs a full virtual fs

resources: access & revoke

- consider revoke support for global&unshareable resources
- required for proper session switching
- priviledged operation
- properly isolate other objects (e.g. gpu buffers)

signals

- it's UNIX, no way to avoid them
- man (7) signal: „slow“ devices can return -EINTR, others restart by default
- „slow“ devices unclear disdinction and autorestart are fragile

signals: solutions

- userspace simply handles -EINTR correctly in all cases
- or don't support signals when blocking

signals: killable waits

- nice, but
- process exit doesn't necessarily close file
- E.g. logind has dup'ed FD for revoke
- hard to test `-EINTR` code in the kernel

signals: „Stop worrying and ...“

- restarting makes testing error paths trivial
- the more interruptible waits you have the better
- duplicate all your functional tests with one where the main thread gets interrupted all the time
- inject `-EINTR` for testing

signals: summary

- support full restarting
- shared `fooIoctl()` in userspace to enforce proper restarting even for `-EINTR`
- exploit `-EINTR` for testing error paths
- or only do blocking on pollable FDs

time: sampling

- make the clocksource clear to userspace, different clocks *will* mismatch
- prefer `CLOCK_MONOTONIC`
- allow userspace to sample hw clocks
- `__s64` seconds + `__u64` nanoseconds for structs (to match `ktime`), enforce normalization

time: waiting

- seriously consider pollable FDs
- support absolute timeouts
- convert relative to absolute for restarting

documentation

- prefer executable specs
- manpages for generic interfaces
- forget about `Documentation/ABI` ... maybe

summary

- real world user
- testcases, testcases, testcases
- don't screw up technicalities too badly, see http://blog.ffwll.ch/2013/11/botching-up-ioctls.html
- think about documentation