Contribution ID: **166** Type: **not specified**

Wrapping system calls in glibc

Tuesday 10 September 2019 10:30 (30 minutes)

The glibc project decided a while back that it wants to add wrappers for system calls which are useful for general application usage. However, that doesn't mean that all those missing system calls are added immediately.

System call wrappers still need documentation in the manual, which can be difficult in areas where there is no consensus how to describe the desired semantics (e.g., in the area of concurrency). Copyright assignment to the FSF is needed for both the code and the manual update, but can usually be performed electronically these days, and is reasonably straightforward. On top of that, the glibc project is seriously constrained by available reviewer bandwidth.

Some more specific notes:

Emulation of the system call is not required. It has been historically very problematic. The only thing that has not come back to bite us is checking if a new flag argument is zero and call the old, equivalent system call instead in this case.

Wrapper names should be architecture-independent if at all possible. Sharing system call names as much as possible between architectures in the UAPI headers helps with that.

Mutiplexing system calls are difficult to wrap, particularly if the types and number of arguments vary. Previous attempts to use varargs for this have led to bugs. For example, open/openat would not pass down the mode flag for O_TMPFILE initially, or cannot be called with a non-variadic prototype/function pointer on some architectures. We wouldn't want to wrap ipc or socketcall (even if they had not been superseded), and may wrap futex as separate functions.

We strongly prefer if a system call that is not inherently architecture-specific (e.g., some new VFS functionality) is enabled for all architectures in the same kernel release.

When it comes to exposing the system call, we prefer to use ssize_t or size_t for buffer sizes (even if the kernel uses int or unsigned int), purely for documentation purposes. Flag arguments should not be long int because it is unclear whether in the future more than 32 flags will be added on 64-bit architectures. Except for pthread_* functions, error reporting is based on errno and special return values.

Passing file offsets through off64_t * arguments is fine with us. Otherwise, off64_t parameter passing tends to vary too much.

If constants and types related to a particular system call are defined in a separate header which does not contain much else, we can include

that from the glibc headers if available. As result, new kernel flags will become available to application developers immediately once they install newer kernel headers. This may not work for multiplexing system calls, of course, even if we wrap the multiplexer.

I agree to abide by the anti-harassment policy

Yes

I confirm that I am already registered for LPC 2019

Primary author: LEVIN, Dmitry (BaseALT)

Co-author: WEIMER, Florian

Presenters: LEVIN, Dmitry (BaseALT); WEIMER, Florian; ROZYCKI, Maciej W.

Session Classification: Toolchains MC