The current trend in memory sizes lead me to believe that we’ll need 128-bit pointers by 2035. Hardware people are starting to think about it [1a] [1b] [2]. We have a cultural problem in Linux where we believe that all pointers (user or kernel) can be stuffed into an unsigned long and newer C solutions (uintptr_t) are derided as "userspace namespace mess".

The only sane way to set up a C environment for a CPU with 128-bit pointers is sizeof(void *) == 16, sizeof(short) == 2, sizeof(int) == 4, sizeof(long) == 8, sizeof(long long) == 16.

That means that casting a pointer to a long will drop the upper 64 bits, and we’ll have to use long long for the uintptr_t on 128-bit.

Fixing Linux to be 128-bit clean is going to be a big job, and I’m not proposing to do it myself. But we can at least start by not questioning when people use uintptr_t inside the kernel to represent an address.

Getting the userspace API fixed is going to be the most important thing (eg io_uring was just added and is definitely not 128-bit clean). Fortunately, no 128-bit machines exist today, so we have a bit of time to get the UAPI right. But if not today, then we should start soon.

There are two purposes for this session:

- Agree that we do need to start thinking about 128-bit architectures (even if they’re not going to show up in our offices tomorrow)
- Come to terms with needing to use uintptr_t instead of unsigned long

[1b] https://github.com/riscv/riscv-opcodes/blob/master/unratified/rv128_i

I agree to abide by the anti-harassment policy

Yes

Primary author:  WILCOX, Matthew (Oracle)
Presenter:  WILCOX, Matthew (Oracle)
Session Classification:  Kernel Summit
Track Classification:  Kernel Summit Track