Optional Widening of C Library Size Arguments in *9k*

Geoff Collyer

9k is a 64-bit capable version of Plan 9. *9k* as delivered had widened many *ulong*s to *uintptr*, which can hold a pointer, as part of making the kernel 64-bit clean, but it did not address actually exploiting the larger address space, particularly in user mode.

An obvious case of wanting *libc* functions to have their size arguments (and internal integers) widened is *malloc*. If one wants, as with the kernel's *Pages* array, to allocate an array that may be larger than 2 or 4 GB, *malloc* itself (in the kernel), *mallocz* (in user land), or the user will likely invoke *memset* to zero such an array. To avoid making a special case of such arrays, *memset* also needs to have its final argument widened to *uintptr* (and `sizeof`'s result should have the same type, which requires only a very small change to the C compiler common code).

> **N.B.:** This has virtually no effect on 32-bit systems (since *uintptr* is *ulong*), so doesn't break compatibility with existing Plan 9 systems, while allowing programs that wish to allocate vast arrays of buffers (e.g., *fossil* or *venti*) to do so straightforwardly.

If you have no need of large address spaces, you need not adopt the change to *malloc* and *memset*'s final arguments.

> **N.B.:** If you have more than about 8GB of RAM, you need large address spaces (for *Page* structs).

This is an entirely optional change, and not an integral part of *9k*, but be aware that you will need to use an alternate `libc/riscv64/memset.s` (provided) on `riscv64` systems if you choose to not widen arguments.

> **N.B.:** As long as your `libc.h`'s declaration of *memset*, etc. agrees with the implementations of *memset*, etc. in your C library, either choice of argument width will work.