Chris Blappert <blappert@cs>
Cooper Johnson <cooperdj@cs>

Lab X: Rust port with userspace heap allocation

For our project, we chose to extend JOS to support dynamic memory allocation in userspace Rust programs, with an allocation code path in Rust all the way through to the implementations of the necessary system calls. We heavily modified an existing Rust allocator library to provide a smaller-than-page-granularity layer on top of sys_page_alloc and sys_page_unmap. We also ported enough kernel-space types and helper functions to implement the core of system call handling, from trap_dispatch all the way through sys_page_alloc and sys_page_unmap, in Rust, as idiomatically as we could. We chose this project because we were interested both in userspace dynamic memory management and in working with Rust.

We encountered many, many issues related to building and linking Rust code with JOS. The Rust toolchain is not yet stabilized for applications not linking against the standard library, and in order to cross-compile, we had to include the source of multiple “built-in” Rust libraries in our repository. The Rust build tool Cargo would ordinarily handle at least part of this, but because JOS involves primarily existing C code, we had to figure out manually how to compile the necessary Rust code with rustc as static libraries and link those into the JOS image. This took a while to get working. We could not figure out how to get rust to correctly look for rlibs either, so we just put all files that got linked together into the same directory (hence why slabmalloc is in user/). There were also some issues with getting the allocator we used working with rust allocation functions (__rust_allocate and __rust_deallocate), which required some wrangling of the type system as creating a static allocator was somewhat difficult. We intended to use the allocation to try to compile some parts of the standard library (e.g. allocation) but those ended up having other dependencies which we had not implemented.

The most interesting code to look at is probably the new implementations of sys_page_alloc and sys_page_unmap. These use multiple new abstractions built on top of Rust’s unique language features and its core library to provide cleaner, safer intra-kernel interfaces. Many of these interfaces are still thin wrappers around unsafe C calls, but they are more difficult to misuse, and they suggest what the kernel might look like if more of it were in Rust.

To build, you need rustc nightly installed. To verify that the Rust system calls still work, and then see evidence that userspace dynamic allocation works, run:

```
git submodule init
git submodule update # this may take time
make grade
make run-rustshim
```