CS510 Languages and Low Level Programming: Portfolio submission, Topics 7 and 8

Due on May 20, 2016 at 11:59pm

Mark P. Jones Spring 2016

Konstantin Macarenco

Topic 7. Explain the use and implementation of capabilities in access control and resource management.

Capabilities a way to control access to a resource. It is often compared to lock, i.e. if an application can access a resource only if it possesses a required lock. Even though this metaphor is similar to actual definition of capabilities it does not reflect some of the details like recursive keys, or keys with the same or fewer restrictions, and “principle of least privilege” - each program should have minimal set of capabilities. Some other desired properties of capabilities:

- Capabilities don’t care who owns them.
- Capabilities can be delegated.
- Copied, passed to anyone.
- Should define set of restrictions (the same file can be accessed by two different capabilities with different level of access).

Linux file descriptor is very close to this definition.

One of the capability based system is seL4 implements capabilities in the following way.
Brief description: All capabilities are stored in protected area (within kernel), and not visible from the outside. Each process has a capability space (CSpace), and as system starts up set of capabilities installed to each process, according to some policies. Each capability has assigned metadata to store some vital information, for example Window Capability might store set of permissions, Untyped Capability has to store pointer to the next available address, etc.
Capabilities management is done by using capability derivation tree, which for performance purposes implemented as doubly link list (efficient sequential traverse and quick insert/delete operations). This way capabilities can be easily shared with other processes without system need to track each one of them.
When a process requests some action, kernel checks if the process has appropriate capability installed in it’s CSpace, if not found action is rejected.

Topic 8. Develop programs using the capability abstraction provided by the seL4 microkernel, or capabilities lab.

To meet this objective I Completed capability lab, except for task 8 (not enough time), which is included along in this submission. During the lab I learned basic principles of working with capabilities, although real seL4 experience probably would be more valuable.
For part 7 (Improving existing operation) I picked the second approach, even though a bit more troublesome to implement. *printString*() system call, provides address to the region of memory to be printed, however this is not safe and should be checked for:

1. The desired address is not in any way interfere kernel address space. (Can be a pointer to kernel space, or non null terminated region of memory, that overlaps with address space).
2. The Array must be within existing memory, it should not point or overlap with any untyped regions.

During system call kernel finds memory page, where the string belongs and find last page with contiguous memory that is started by this page, check for null in this region.