Chapter 13

System Calls

System calls is the way user-mode applications interact with the kernel - to ask for resources, request operations to be performed, etc. The system call API is the part of the kernel that is most exposed to the users, therefore its design requires some thought.

Designing System Calls

It is up to us, the kernel developers, to design the system calls that application developers can use. We can draw inspiration from the POSIX standards or, if they seem like too much work, just look at the ones for Linux, and pick and choose. See the section "Further Reading" at the end of the chapter for references.

Implementing System Calls

System calls are traditionally invoked with software interrupts. The user applications put the appropriate values in registers or on the stack and then initiates a pre-defined interrupt which transfers execution to the kernel. The interrupt number used is dependent on the kernel, Linux uses the number `0x80` to identify that an interrupt is intended as a system call.

When system calls are executed, the current privilege level is typically changed from PL3 to PL0 (if the application is running in user mode). To allow this, the DPL of the entry in the IDT for the system call interrupt needs to allow PL3 access.

Whenever inter-privilege level interrupts occur, the processor pushes a few important registers onto the stack - the same ones we used to enter user mode before, see figure 6-4, section 6.12.1, in the Intel manual [33]. What stack is used? The same section in [33] specifies that if an interrupt leads to code executing at a numerically lower privilege level, a stack switch occurs. The new values for the registers `ss` and `esp` is loaded from the current Task State Segment (TSS). The TSS structure is specified in figure 7-2, section 7.2.1 of the Intel manual [33].

To enable system calls we need to setup a TSS before entering user mode. Setting it up can be done in C by setting the `ss0` and `esp0` fields of a "packed struct" that represents a TSS. Before loading the "packed struct" into the processor, a TSS descriptor has to be added to the GDT. The structure of the TSS descriptor is described in section 7.2.2 in [33].

You specify the current TSS segment selector by loading it into the `tr` register with the `ltr` assembly code instruction. If the TSS segment descriptor has index 5, and thus offset `5 * 8 = 40 = 0x28`, this is the value that should be loaded into the register `tr`.

When we entered user mode before in the chapter "Entering User Mode" we disabled interrupts when executing in PL3. Since system calls are implemented using interrupts, interrupts must be enabled in user mode. Setting the IF flag bit in the `eflags` value on the stack will make `iret` enable interrupts (since the `eflags` value on the stack will be loaded into the `eflags` register by the assembly code instruction `iret`).

Further Reading

- The Wikipedia page on POSIX, with links to the specifications: http://en.wikipedia.org/wiki/POSIX
- A list of system calls used in Linux: http://bluemaster.iu.hio.no/edu/dark/lin-asm/syscalls.html
- The Wikipedia page on system calls: http://en.wikipedia.org/wiki/System_call
- The Intel manual [33] sections on interrupts (chapter 6) and TSS (chapter 7) are where you get all the details you need.