System Calls

The interface between an application program and the Operating System is through system calls.

CONTENTS

What is a system call?

The operating system is responsible for

- Process Management (starting, running, stopping processes)
- File Management(creating, opening, closing, reading, writing, renaming files)
- Memory Management (allocating, deallocating memory)
- Other stuff (timing, scheduling, network management)

An application program makes a *system call* to get the operating system to perform a service for it, like reading from a file.

One nice thing about syscalls is that you don't have to link with a C library, so your executables can be much smaller.

System Calls in 32-bit Linux

- To make a system call in 32-bit Linux, place the system call number in `eax`, then its arguments, in order, in `ebx`, `ecx`, `edx`, `esi`, `edi`, and `ebp`, then invoke `int 0x80`.
- Some system calls return information, usually in `eax`.
- All registers are saved across the system call.

System Calls in 64-bit Linux

hello.s

```
# ----------------------------------------------------------------------------------------
# Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.
# To assemble and run:
#
#     gcc -c hello.s && ld hello.o && ./a.out
#
# or
#
#     gcc -nostdlib hello.s && ./a.out
# ----------------------------------------------------------------------------------------

        .global _start

        .text
_start:
        # write(1, message, 13)
        mov     $1, %rax                # system call 1 is write
        mov     $1, %rdi                # file handle 1 is stdout
        mov     $message, %rsi          # address of string to output
        mov     $13, %rdx               # number of bytes
        syscall                         # invoke operating system to do the write

        # exit(0)
        mov     $60, %rax               # system call 60 is exit
        xor     %rdi, %rdi              # we want return code 0
        syscall                         # invoke operating system to exit
message:
        .ascii  "Hello, world\n"
```

- To make a system call in 64-bit Linux, place the system call number in `rax`, then its arguments, in order, in `rdi`, `rsi`, `rdx`, `r10`, `r8`, and `r9`, then invoke `syscall`.
- Some system calls return information, usually in `rax`. A value in the range between -4095 and -1 indicates an error, it is -errno.
- The system call destroys rcx and r11 but others registers are saved across the system call.
- Full details are in Section A.2.1 of the The AMD64 ABI.

Lists of Linux System Calls

There are hundreds of system calls in Linux. A good online source for 32-bit Linux is http://syscalls.kernelgrok.com/ . For 64-bit Linux, check out http://www.acsu.buffalo.edu/~charngda/linux_syscalls_64bit.html

Check those pages, and of course, the Linux source.

macOS System Calls

Windows System Calls