Operating Systems
COMS W4118
Lecture 3

Alexnader Roth

2015 – 01 – 29

1 System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accepted by programs via a high-level. **Application Programming Interface (API)** rather than direct system call use.
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX- based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

Note that the system-call names used throught this text are generic.

- System calls are the entry point for kernel code.
- Kernel code only way to interact with raw memory, hardware.
- The kernel code is written already.

2 System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers.
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values.
- The caller need know nothing about how the system call is implemented.
  - Just need to obey API and understand what OS will do as a result call.
  - Most details of OS interface hidden from programmer by API

3 Parameter Passing

- Parameters are stored into registers.
- Registers will be accessed later for function calls.

4 Example: MS-DOS

- Unix operating systems are designed to be multi-user and multi-process.
- The user methods are isolated from other user processes and the operating system itself.
- There are specific privileged operations that the user applications cannot access. Need to ask permission from the operating system.
- The C-wrapper functions are just a user level implementation, the C wrappers are a convenience.
- Because of such small memory size, it would crunch down and reduce size of shell to allow other programs to run.
- There was no concept of virtual memory. Every program worked with actual memory.
- There was no such thing as two programs running at the same time.
- You could write to any register you want; there was no such thing as privileged users.

5 Microkernel System Structure

- Two ways of structing systems
    1. Microkernel - the kernel is very small, it only does memory management and scheduling. It has facilities for other programs to message each other. Separate out all the other subsystems. Very easy to maintain the core parts. The parts that can do privileged operations are very small and well-defined, but subsystems must do message passing between themselves to the kernel. There is a communication overheard.
    2. Monolithic - there is one large file that gets run. Everything is a function call within one program. Disadvantage of having to compile the whole kernel at once. You cannot add modules at run time. All parts of the kernel are in a single process; thus, it is prone to crashing.

6 Kernel Modules

- There is a logical separation between subsystems in a kernel.
- Modules can be loaded into and unloaded from the kernel at runtime.

7 Hybrid Systems

- Most modern operating systems are acutally not one pure modle
  - Hybrid combines multiple approaches to address performance, security, usability needs.

8 Linux System Overiew: From Boot To Panic

8.1 Boot Process

1. When we boot up, the CPU begins with instructions running in ROM. Runs **power-on self-test (POST)**, checks that things are operating well.
2. Jumps to the boot code in the first 440 bytes of the Master Boot Record (MBR). Very first sector of the disk drive.
   - Parition - disk drive can be divide into logical sections and allow file systems in different logical directories.
3. MBR boot code locates and launches boot loader. We used GRUB. GRUB provides the user with the kernel menu. It locates the kernel image and allows the user which kernel to load.
4. The Boot loader will load the kernel and launch it.
5. The kernel has to run `initramfs` (initial RAM filesystem).
   - Initial temporary root filesystem
   - Contains device drivers needed to mount real root filesystem.
   - Mounting means mapping a directory from one subtree to another logical directory tree.
   - Allows for encapsulation between directories.
6. Kernel switches to real root filesystem.

8.2 User Session

1. `init` calls `getty`
    - `getty` is called for each virtual terminal (normally 6 of them)
    - `getty` checks user name and password against `/etc/passwd`
2. `getty` calls `login`, which then runs the user's shell
    - `login` sets the user's environment variables
    - The user's shell is specified in `/etc/passwd`

9 Process Control

9.1 Displaying Process hierarchy

- `ps` command displays what processes are running.
- The always running process is `init`, which is process 1.
- When a parent process dies, the child process is orphaned. This orphaned process will be inherited by `init`.
- A zombie process is a child function that has ended but the parent is waiting on that process to return.