Operating Systems: Structures

Alex Chi

Update: 2020-03-02

Contents

1 Service Structure **1**

1.1 User Interfaces 2

1.2 System Calls 2

1.2.1 API 2

2 Component Structure **2**

2.1 MS-DOS 3

2.2 Traditional UNIX 3

2.3 macOS 3

2.4 Solaris 4

2.5 What is System Program? 4

2.6 Virtual Machine 4

2.7 Java Virtual Machine 4

2.8 Questions about VM 4

1 Service Structure

Operating system provides services for programs.

1.1 User Interfaces

macOS, Windows, KDE, Gnome

Touchscreen, Mouse (pointing devices)

1.2 System Calls

System calls are services provided from OS.

Examples:

- write, read files
- list files under folder

Syscalls separate user programs and kernel programs. [Figure on Page 24]

After user program initiate a syscall, the CPU will switch into kernel mode, and then comes at the syscall entry. Kernel looks up corresponding function in syscall table, calls that function in kernel space, and then returns control to user space.

1.2.1 API

But doing a simple task might be hard only with syscall. 3 most common APIs include:

- Win32 API
- POSIX API
- Java API

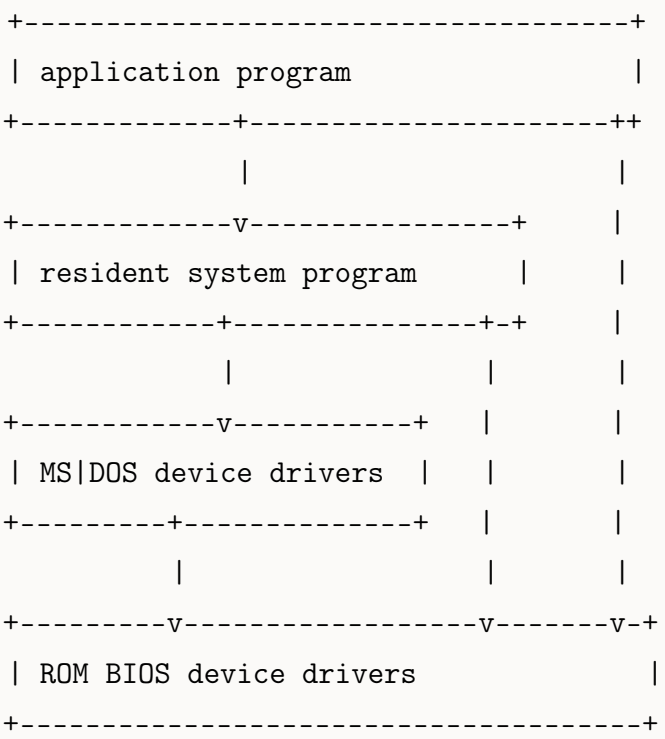
Why using API instead of syscalls?

- **Simplicity** For example, we use `FILE*` pointers instead of file descriptor in the C programming language. This is one abstraction provided by OS.
- **Portability** Different platforms provide different syscall models. For example, `printf` function can be used on all platforms with C runtime library. But its implementation differs. On *nix systems, it calls `write` syscall. On Windows, I guess it calls `WriteConsole` functions and alike.

2 Component Structure

2.1 MS-DOS

- very small, just a few MB
- not divided into modules
- not well modulized
- user program has direct access to hardware



2.2 Traditional UNIX

- layered structure
- hardware (layer inner) > kernel > user program (layer n)‘
- secure, but not efficient

2.3 macOS

- micro-kernel (Mach)
- reliable, secure
- overhead of switching into and from kernel space
- portability
- use message passing for inter-process communication

2.4 Solaris

- modular approach
- core component separated, talking to each other over known interfaces

2.5 What is System Program?

- file system, drivers, etc.
- embedded in kernel, or come from suppliers

2.6 Virtual Machine

- type 0 hypervisor: require hardware support
- type 2 hypervisor: VMware / VirtualBox or alike

1. Benefits
   - multiple OS on a single machine
   - hardware-independence
   - better utilization
   - encapsulating, portability (docker)

2.7 Java Virtual Machine

In fact, JVM is different from VM mentioned before. VMware or alike

2.8 Questions about VM

1. Can I control VM remotely, or move them to elsewhere?
   Surely you can SSH or VNC to your remote VM. You can control AWS EC2 by SSH. Most cloud providers support moving VM to another data center. You can also move VMware VMs to another machine.
2. Is VM safe?
   Unless there's bug with VMM, it is safe.
   But there's still possibility that: host OS and guest OS shares the same network, and the virus affects the host OS by leak in network stack.
3. Can OS inside VM identifying it's running in a VM?

VM software tries to emulate as if OS inside owns the hardware, but there's still clues for detecting that. For example, driver name. For computer without nested virtualization support, VM cannot be run in VM, this can also be used as a clue.