National University of Singapore

CS2106 Operating System

Midterm Summary Notes

Dong Shaocong A0148008J

March 15, 2018

1 Basic Idea

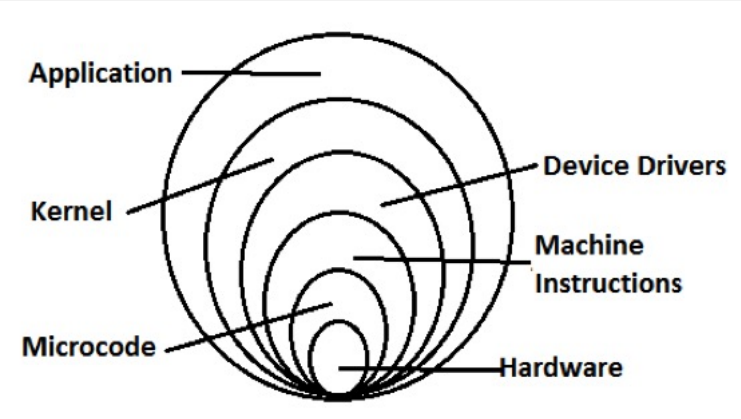
Definition 1.1. Operating System is a suite (i.e. a collection) of specialised software that:

- Gives you access to the hardware devices like disk drives, printers, keyboards and monitors.
- Controls and allocate system resources like memory and processor time.
- Gives you the tools to customise your and tune your system.

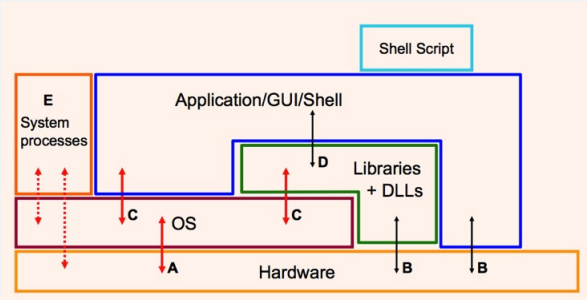
Example 1.1. LINUX, OS X (or MAC OS, a variant of UNIX), Windows 8

What are Operating System? It usually consists of several parts. (Onion Model)

- Bootloader First program run by the system on start-up. Loads remainder of the OS kernel.
  - On Wintel systems this is found in the Master Boot Record (MBR) on the hard disk.
- Kernel The part of the OS that runs almost continuously.
- System Programs Programs provided by the OS to allow:
  - Access to programs.
  - Configuration of the OS.
  - System maintenance, etc.



Abstraction Layer & Opearating System Structure



- **A** : OS executing machine instructions, maybe ***privileged***
- **B** : normal machine instructions executed (program/library code)
- **C** : calling OS using ***system call interface***
- **D** : program calls library code
  - ***statically linked:*** contained in executable
  - ***dynamic link libraries (DLL):*** loaded at runtime
- **E** : system processes
  - usually special
  - sometimes part of the OS

OS takes control in **A** and **C**, maybe **E**.

Definition 1.2. Boostrapping

- The **OS is not present in memory** when a system is cold started.
  - When a system is first started up, memory is completely empty.
- We start first with a **bootloader** to get an operating system into memory.
  - Tiny program in the first (few) sector(s) of the hard-disk.
  - The first sector is generally called the boot sector or master boot record for this reason.
  - Job is to load up the main part of the operating system and start it up.

Definition 1.3. Core CPU units that can execute processes, because we have much more number of processes than the number of cores, we have to do **context switching** to share a core very quickly between different processes.

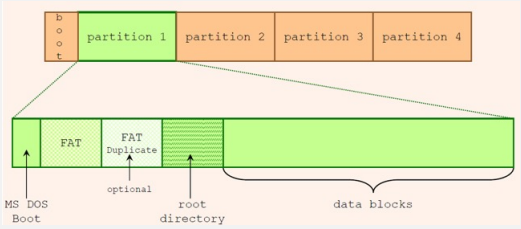
- Entire sharing must be transparent.
- Processes can be suspended and resumed arbitrarily.

Definition 1.4. Context switching

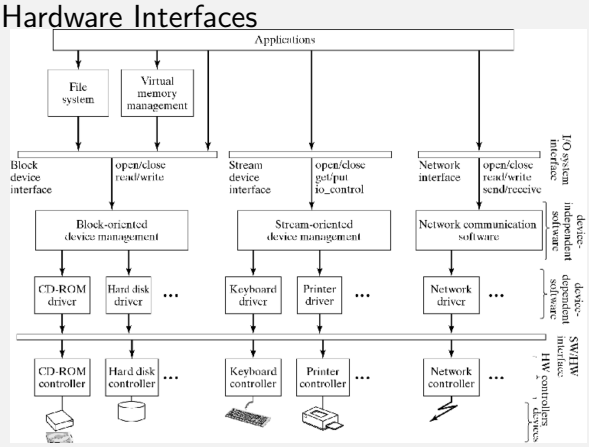
1. Save the context of the process to be suspended.
2. Restore the context of the process to be (re)started.
3. Issues of scheduling to decide which process to run.

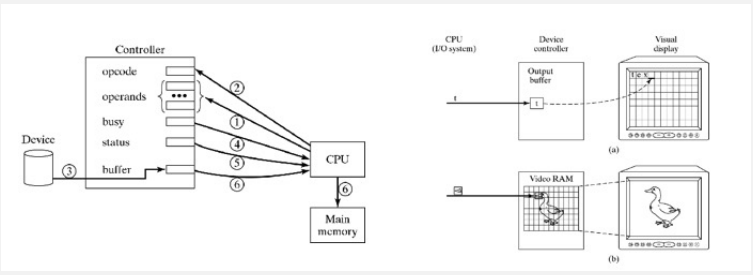
Definition 1.5. File system A set of data structures on disk and within the OS kernel memory to organise persistent data.

How OS file system works?



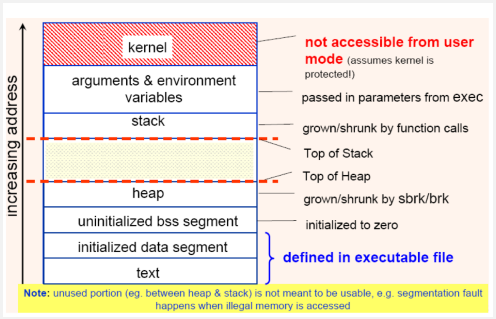
Hardware Interfaces





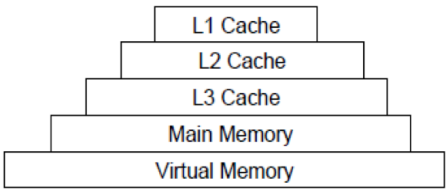
Definition 1.6. Memory static/dynamic (new, delete, malloc, free). Memory to store instructions Memory to store data.

Memory Management



Definition 1.7. Virtual Memory management

- For cost/speed reasons memory is organized in a hierarchy:



- The lowest level is called "virtual memory" and is the slowest but cheapest memory.
  - Actually made using hard-disk space!
  - Allows us to fit much more instructions and data than memory allows!

Definition 1.8. OS security

- Data (files): Encryption techniques, Access control lists
- Resources: Access to the hardware (biometric, passwords, etc), Memory access, File access, etc.

Writing an OS (BSD Unix)

Machine independent

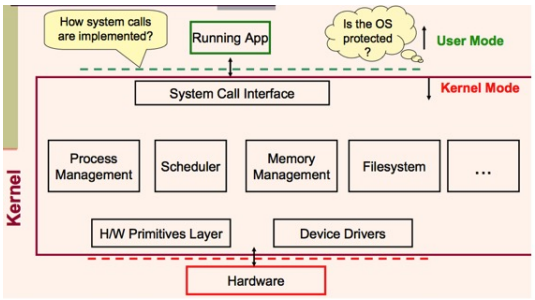
- 162 KLOC
- 80% of kernel
- headers, init, generic interfaces, virtual memory, filesystem, networking+protocols, terminal handling

Machine dependent

- 39 KLOC
- 20% of kernel
- 3 KLOC in asm
- machine dependent headers, device drivers, VM

Definition 1.9. Kernel

- Monolithic Kernel (Linux, MS Windows)
  - All major parts of the OS-devices drivers, file systems, IPC, etc, running in "kernel space" (an elevated execution mode where certain privileged operations are allowed).
  - Bits and pieces of the kernel can be loaded and unloaded at runtime (e.g. using "modprobe" in Linux)



- MicroKernel (Mac OS)
  - Only the "main" part of the kernel is in "kernel space" (Contains the important stuff like the scheduler, process management, memory management, etc.)
  - The other parts of the kernel operate in "user space" as system services: The file systems, USB device drivers, Other device drivers.

External View of an OS

- The kernel itself is not very useful. (Provides key functionality, but need a way to access all this functionality.)
- We need other components:
  - System libraries (e.g. stdio, unistd, etc.)
  - System services (creat, read, write, ioctl, sbrk, etc.)
  - OS Configuration (task manager, setup, etc.)
  - System programs (Xcode, vim, etc.)
  - Shells (bash, X-Win, Windows GUI, etc.)
  - Admin tools (User management, disk optimization, etc.)
  - User applications (Word, Chrome, etc).

Definition 1.10. System Calls calls made to the Application Program Interface or API of the OS.

- UNIX and similar OS mostly follow the POSIX standard. (Based on C. Programs become more portable.) *POSIX: portable operating system interface for UNIX, minimal set of system calls for application portability between variants of UNIX.*
- Windows follows the WinAPI standard. (Windows 7 and earlier provide Win32/Win64, based on C. Windows 8 provide Win32/Win64 (based on C) and WinRT (based on C++).)

Example 1.2. User mode + Kernel mode

- Programs (process) run in user mode.
- During system calls, running kernel code in kernel mode.
- After system call, back to user mode.

How to switch mode? Use privilege mode to switching instructions:

- syscall instruction
- software interrupt - instruction which raises specific interrupt from software.

Example 1.3. LINUX system call

- User mode: (outside kernel)
  - C function wrapper (eg. **getpid()**) for every system call in C library.
  - assembler code to setup the system call no, arguments
  - trap to kernel
- Kernel mode: (inside kernel)
  - dispatch to correct routine
  - check arguments for errors (eg. invalid argument, invalid address, security violation)
  - do requested service
  - return from kernel trap to user mode
- User mode: (Outside kernel)
  - returns to C wrapper - check for error return values

2 Process Management

- 123

red sftext `tttext`

> Penghao is the best LaTeX writer.

```
def func():
    print("Penghao is cool!")
```

℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘℘

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1. 123
2. 123
3. 123

	M1	M2	M3	M4	M5
M1	0	108	180	228	396
M2		0	72	168	288
M3			0	48	144
M4				0	128
M5					0

Algorithm 1 title

1: 123

References

[1] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies* `www.google.com.sg`]. Annalen der Physik, 322(10):891921, 1905.