CSE 323 / L-11 / 16.03.2024 /

Linux System Structure:

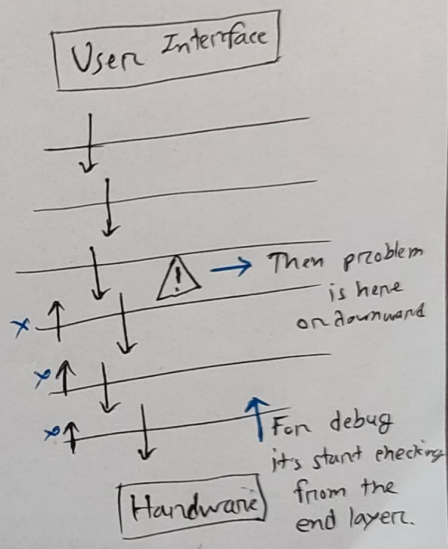
- based on UNIX
- application use the glibc standard C library.
- monolithic, run entirely in kernel mode in a single address space.

Slide-22

- modular design.

Layered Approach:

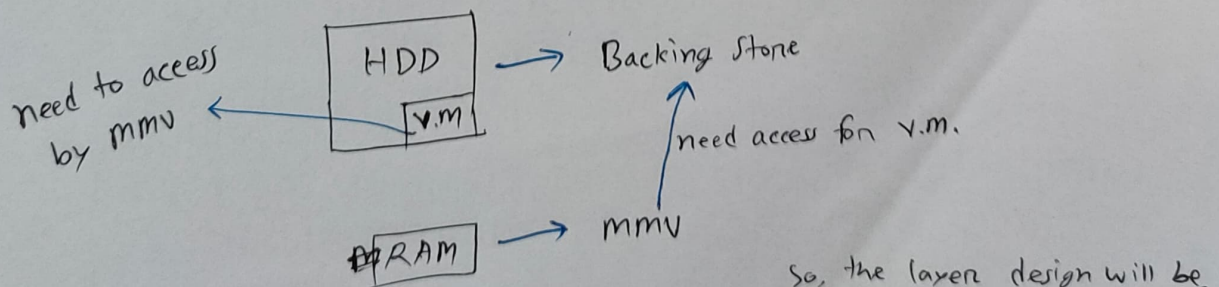
- Top layer ⇒ User Interface
- bottom layer ⇒ handware
- Best for debugging



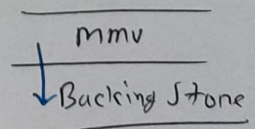
Problems:

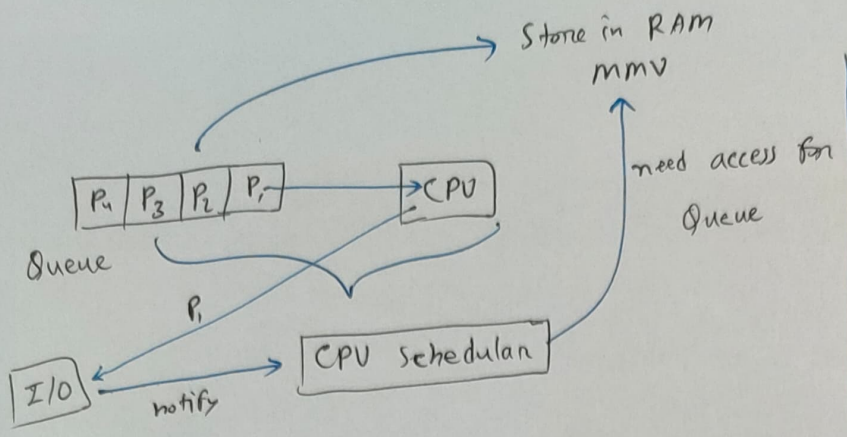
- layer access is slow process
- designing layer is challenging

⇒ need to make sure all function access bellow function. They can't access function from above layer



So, the layer design will be



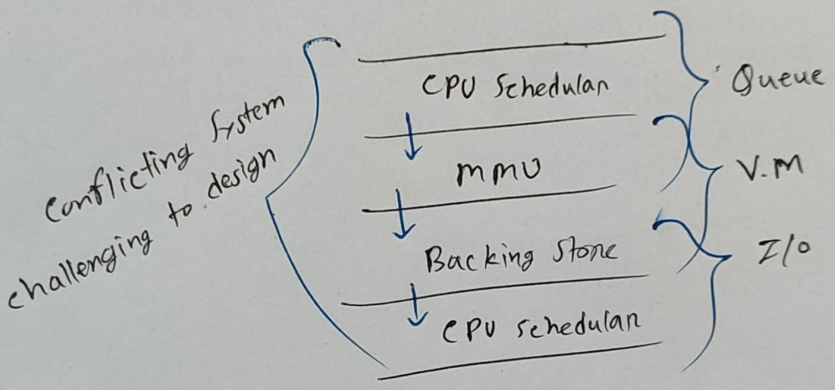


if a instruction need I/O operation then the inst. will move to I/O Device Driver. After I/O operation, instruction will be back to Queue again. CPU schedulan need to notify about the comeback.

⇒ CPU schedulan need to access MMU

and Backing store need to access CPU schedulan

MMU need to access Backing Store



- Microkernel System Structure!

  - Part of OS outside of kernel, run in user mode.
  - communication between user modules using message passing.

⇒ Benefits:

- Easier to extend a microkernel
- Easy to port to new architecture
- more reliable, secure

⇒ Performance overhead.

Slide - 2.33

Modules

Microkernel vs Modular Approach

- loadable kernel modules.
  - uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interface
  - loadable in boot time or run time.
  - similar to layer but more flexible, any module can call any module
  - kernel size vary on usage
  - primary module has only core function - similar to microkernel without invoking message passing.

Slide - 2.35

Hybrid System:

- monolithic
- run in single address space
- module add in the kernel dynamically

macOS and iOS structure:

⇒ User experience layer:
- macOS uses Aqua UI
- iOS uses Springboard UI

⇒ Core frameworks
- Quicktime and OpenGL.

Slide - 2.37

⇒ Application frameworks layer: → iOS
- Cocoa & Cocoa Touch frameworks
  - provide API for the Objective-C and Swift programming language.

↙ macOS

Darwin!

- consists of Mach microkernel and the BSD UNIX kernel.
- two system call interface
  - Mach system call (traps)
  - BSD system call (POSIX)
- Mach kernel
  - MMU
  - CPU scheduler
  - IPC - Interprocess communication
- kernel extensions or kexts
  - I/O kit for device driver
  - dynamically loadable module

Slide - 2-38

OS Debugging:

- generate log files containing error info.
- for application program fail ⇒ core dump
  - capture memory of the process
- for system failure ⇒ crash dump
  - kernel memory
    - trace listings
    - Profiling

⇒ Kernighan's Law:

- Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Operating System Generation: SYSGEN

- Obtain information concerning the specific configuration of the hardware system.
  - CPU model and architecture
  - Disk formate, partition details
  - Memory availability
  - Device availability

Then determine
  - Operating System Option or parameter
  - buffer of which size
  - which CPU scheduling algorithm
  - maximum number of processes.

Mid-1
Upto This
23.03.2024