Lecture 3 19.7.2018 Lecture 3

19 July 2018 10:25

Recap:

- OS provides several services
  - process creation / deletion
  - memory allocation / separation
  - safe device access

via different abstractions
- files
- processes
- address space
- Network interfaces

- two main tasks performed by an OS
  (i) resource mgmt. / multiplexing
  (ii) isolation & control.
- Key requirement to do above tasks is that the OS owns all resources.
  ⇒ all access to resources need OS mediation.

Types of OSes.
(i) monolithic vs. microkernel vs. unikernel
(ii) server vs. desktop vs. handhelds vs. embedded

---

two Key building blocks for OSes
(i) (privileged) modes of execution
(ii) interrupts / interrupt-driven ops.

(a) to execise control on resources & their mgmt.
(to prevent bad things)
- memory isolation
- safe / fair / no-starvation scheduling

(b) idea: all tasks related to resource control / mgmt. to be handled by OS, rest user programs can directly execute on CPU.
- scheduling, allocation etc.

user-mode — less privileges

Kernel-mode — for OS actions

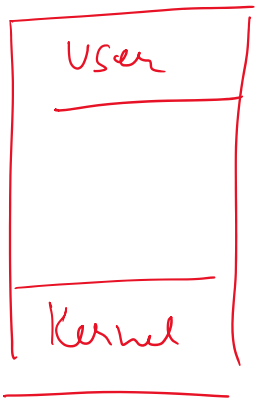
(c) how are the modes put in action?
- requires h/w support (in CPU)
- every instruction requires a minimum privilege level to execute.
- h/w regs. stores CPL, which CPU implicitly checks.

(d) e.g: to access memory of a process certain h/w regs. need to be setup.

Ideal virus (x86)

- a process certain h/w reg. need to be setup.
  - e.g. regs. CR3 on x86 systems.
  - updation of CR3 requires Kernel mode (Ring 0)
  - writes from user mode have no effect or result in error.
  - write to interval timer (same idea).

logical view (x86)



(ii) interrupts

- an interrupt, interrupts current execution on CPU so that other work can be done.
- interrupts "handled" by interrupt handlers, interrupt service routines (ISRs)./functions.

two building blocks
(i) (privileged) modes of execution
(ii) interrupts

Why interrupts important ?

(i) the world is non-deterministic. "anything can happen any time"
  - pkt. arrivals, key presses, disk reads.
  - how does a CPU/OS know when to process these events.
    - interrupts, polling.

(ii) basis mechanism to invoke OS (for services).
  - use of interrupt to switch to Kernel mode for OS service
  - implicit ——— access unallocated memory (page fault)

vs.

explicit — system call (issues a software interrupt)

interrupts
- h/w — IO events
  - kbd
  - pkts
  - timer
- s/w
  - implicit
    - divide by zero
    - page fault
  - explicit — system calls / traps

Q. — what is special about the timer interrupt ?

— how does CPU know which ISR to execute ?

— constraints / conditions on ISR code ?

Lab 2 Heads up. Processes.

Process creation, mgmt from user space.

— fork & co