Lecture 8 CS 333/347 22.8.22

recap: bootup process ⇒ OS in memory

code data

first user process

↳ drives for rest of the work (user processes)

↳ system calls (appl^n binary interface)

(*) system call needs

(i) mechanism to switch privilege modes

(ii) mechanism to invoke a system call

(iii) mechanism to pass arguments

(iv) mechanism to identify (& index) system calls (handlers)

(*) e.g: Intel x86 ISA

(1) ⇒ int 0x80 ~ software explicit interrupt

- switches privilege mode of CPU to higher mode
- saves context of user process on the kernel stack
  - PC
  - gen-purpose CPU regs.
  - SP
- jumps to system call handler.

iret

- restores context on CPU from kstack
- switches to user mode
- jumps to PC of user process

one per process

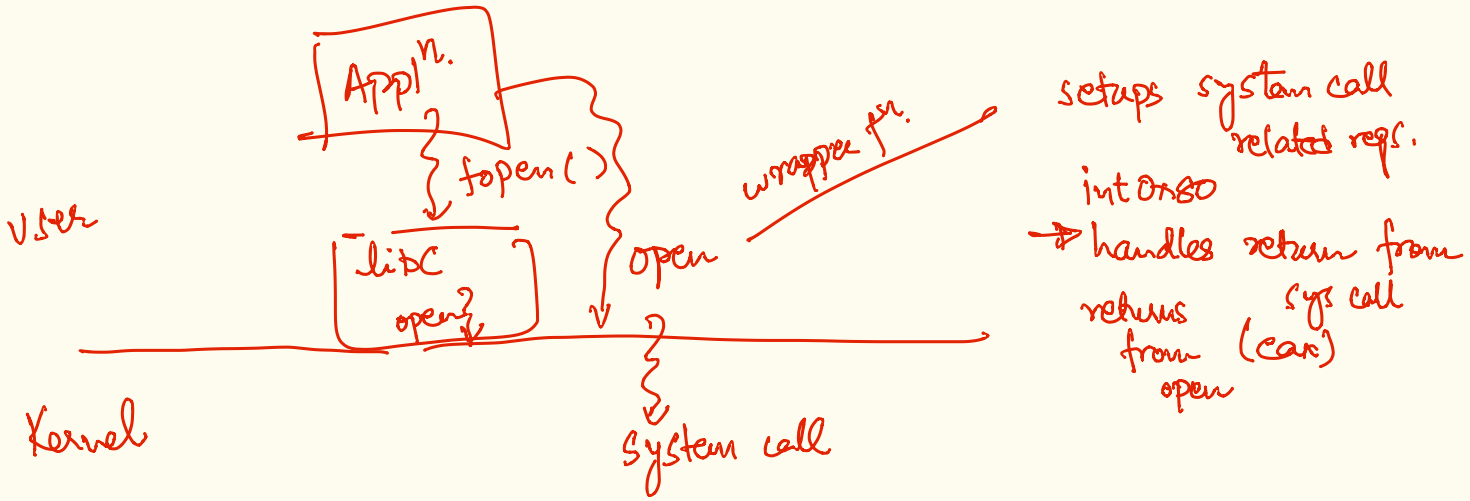
(2) arguments / return values.

~ general purpose regs. are used for arguments.

eax ← store system call number

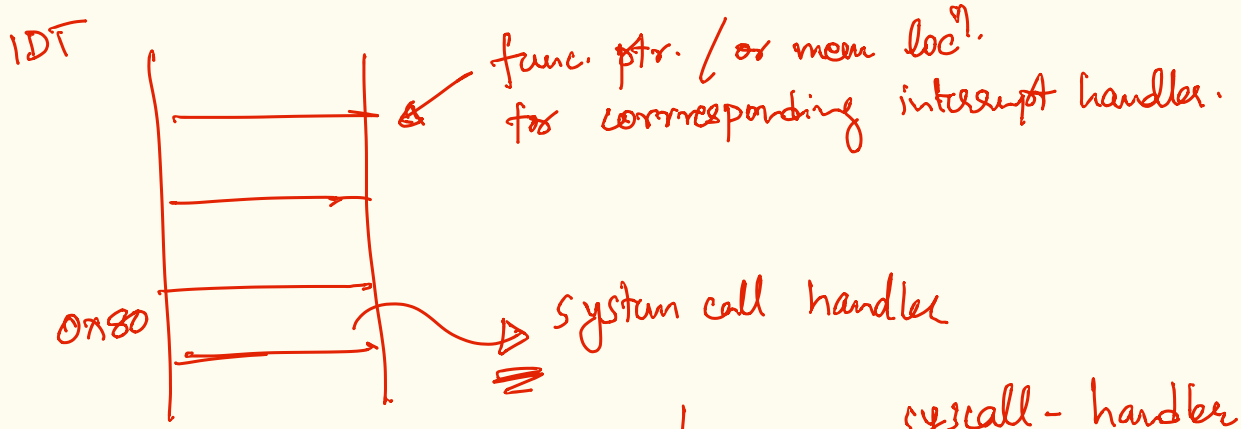
ebx
ecx
cdx
⋮
} arguments of the call.

– int 0x80



(3) IDT — interrupt descriptor table

IDTR — IDT register.




```
syscall-handler() {
  fn = syscall-table[eax];
  call fn;
  iret;
}
```