CS 219 Lecture 10

Quiz 1, 28th Jan Wed. 8.30am

Labquiz 1, 29th Jan, Thu 2pm - 5pm

the system calls mechanism

~ interface of all OS services

- list/set of functionality (which looks / is a function)

~

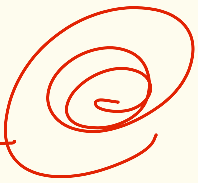
API vs ABI (system call)

- helloworld.c - source

~ printf
scanf } object files

~ add() - source

~ sub() - source



compilation + linker

↓

helloworld binary.

~ source code/ object file interaction.

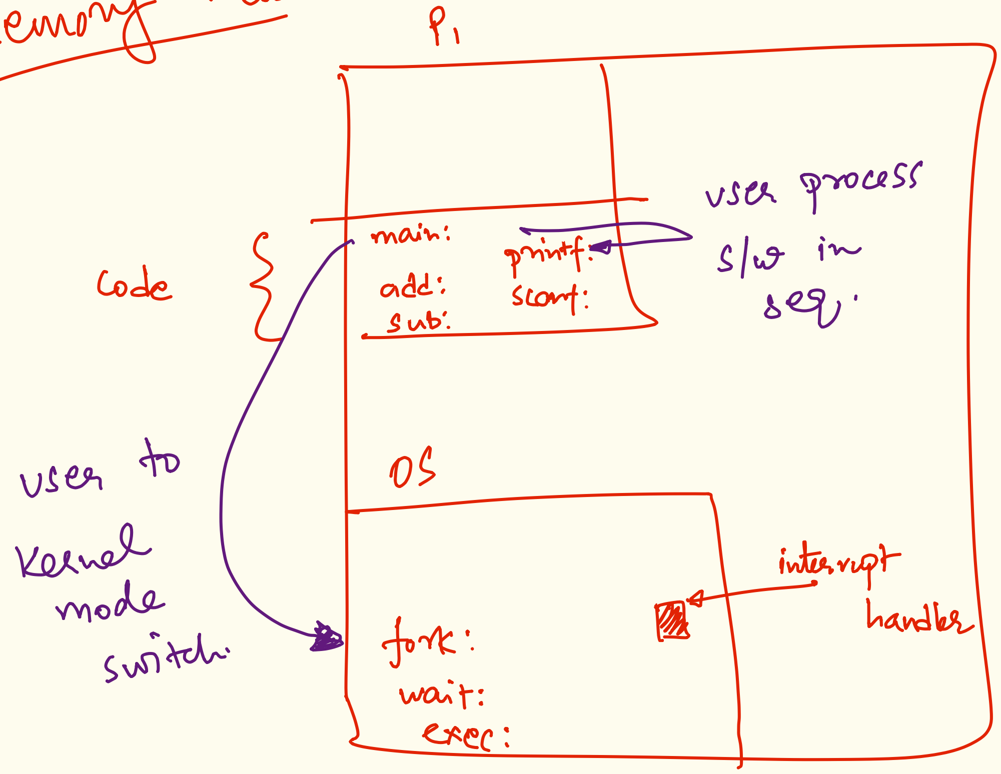
heap
stack
data
code

~ binary interface

~ interaction between (pre-loaded) binary regions in memory

~

memory view



requirements

- ~ how to invoke ?
- ~ mechanism to switch privilege levels ?
- ~ context save & restore
- ~ mechanism to handle/ pass arguments
  return values.
- ~ mechanism to specify system call identity
- ~ mechanism to locate system call code !

(*) all systems calls are interrupts ?

interrupts

- hardware — external IO devices.
- software (due to instruction execution)
  - implicit
    - ~ divide by zero
    - ~ NULL ptr. deref.
  - explicit
    - system calls

⊛ ISA-specific mechanism

x86	risc-v
INT 0x80	ecall

on an explicit s/w interrupt —

~ s/w to CPU privileged mode / Kernel mode

~ save context of running ~~the~~ instance (process).

~ switch to Kernel config. < stack, memory

~ JMP to interrupt handler

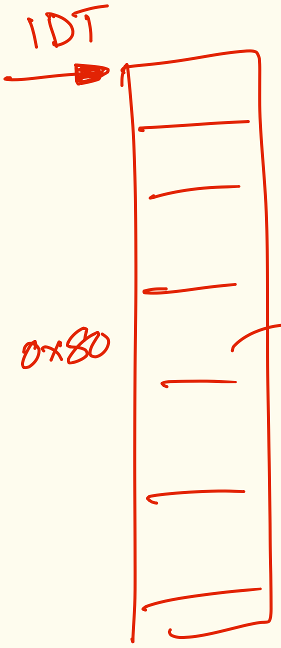
~ ....

⊛ ~ iret /sret ~ return from handler.

x86

IDTR — interrupt descriptor table register.

— stores loc^n. of IDT



f^n ptr. to interrupt handler 0x80

riscv

stvec

scause ~ $\boxed{8}$