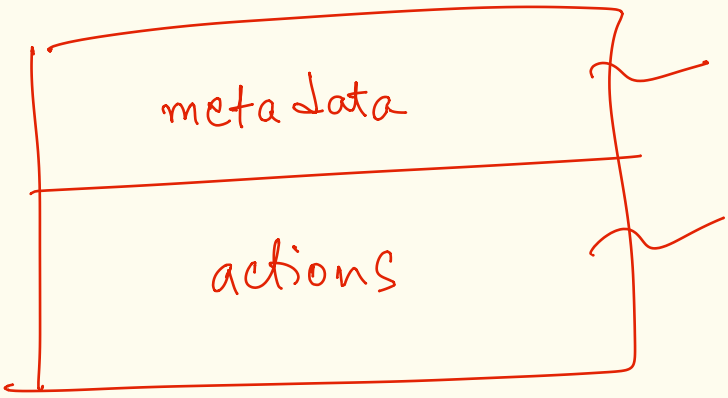
CS 347 — Lecture #5

* Lab2 is available — process system calls usage.

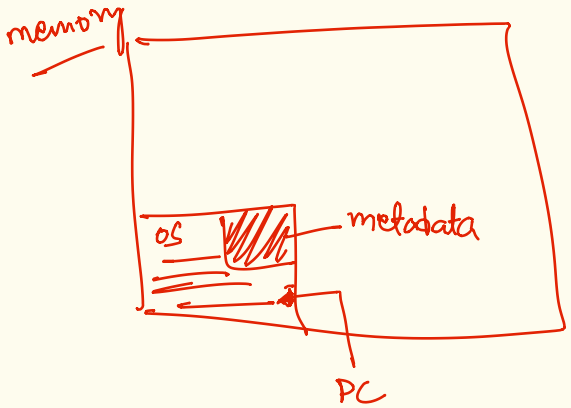
* the process abstraction
  - program vs process
  - process state & transitions
  - PCB ~ process control block

metadata about the process abstraction (maintained / used by the OS)

two-block representation of all-things (abstractions) OS



① game plan of an OS



step 1: (OS) load itself in memory (boot up process)

step 2: what next? Q

~ w/o invoking the OS interface nothing happens.

~ "handcrafts" a (user-level) process (init process)
- s/w's mode to user mode & jmps to process start.

user mode (processes) { { {

interface (ABI) system call

process mgmt. | mem mgmt | IO mgmt

invoke OS functionality - services

Kernel mode

H/w

(*) Step 2 is useful because
- first consumer of services
~ can invoke service to create new processes:-
(generalization + decoupling)
- start
- stop
- load new programs
- pause
- open files

(2) common process related/mgmt. system calls

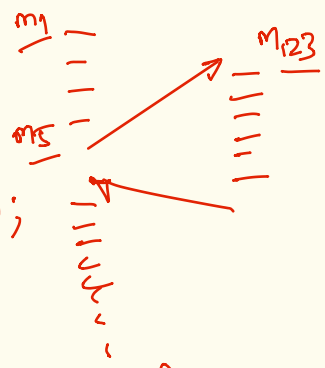
fork()
- creates (duplicates) a process.
- create a new PCB
- populate PCB entries
} OS metadata variables
- first instruction to execute is the new (child) process is after fork.

exec()

wait

e.g: P_1
int a = 23;
fork();
P_1
P_2 (child)
return PC loc^n in both P_1 & P_2
printf(a);
$ 23
$ 23

main() {
int a, b, c;
c = sum(a, b);
}



(*) System call vs function call } user level processes
~ s/w to Kernel-mode on invocation & back to user mode on return
all in user mode

(*) return value of fork in parent & child process is different.

- PID of child process in the parent process
- 0 in the child process.

---

eg:

```
int a=23;
if (fork()==0){      //Child process
    a=a+1;
    printf(a);
}
else {
    a=a-1;
    printf(a);
}
```

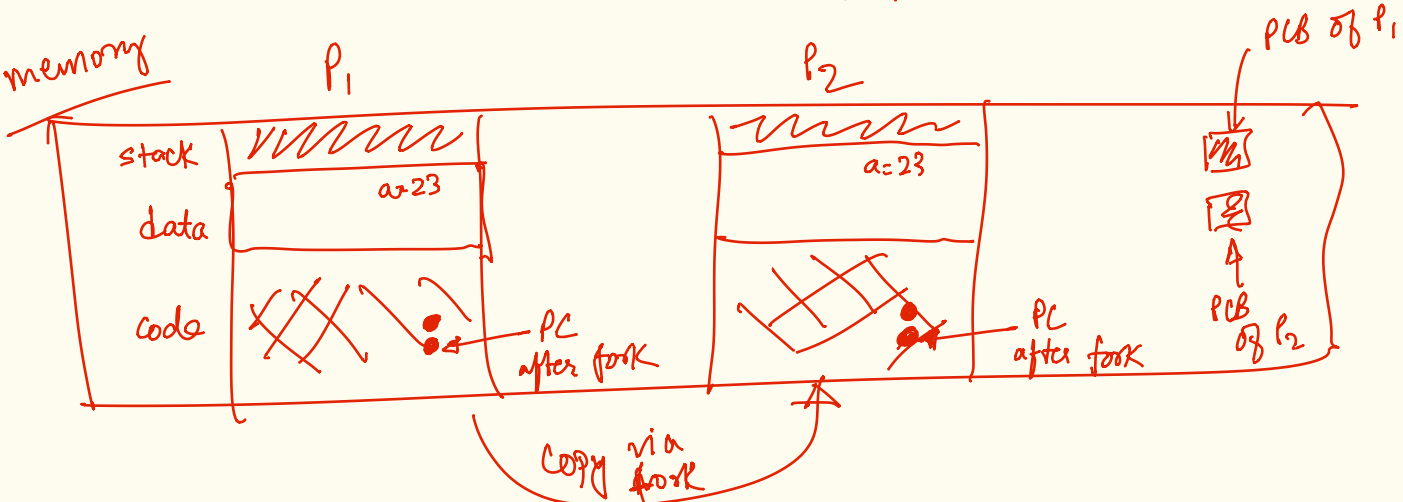
$ 24

$ 22

---

fork creates / duplicates a process

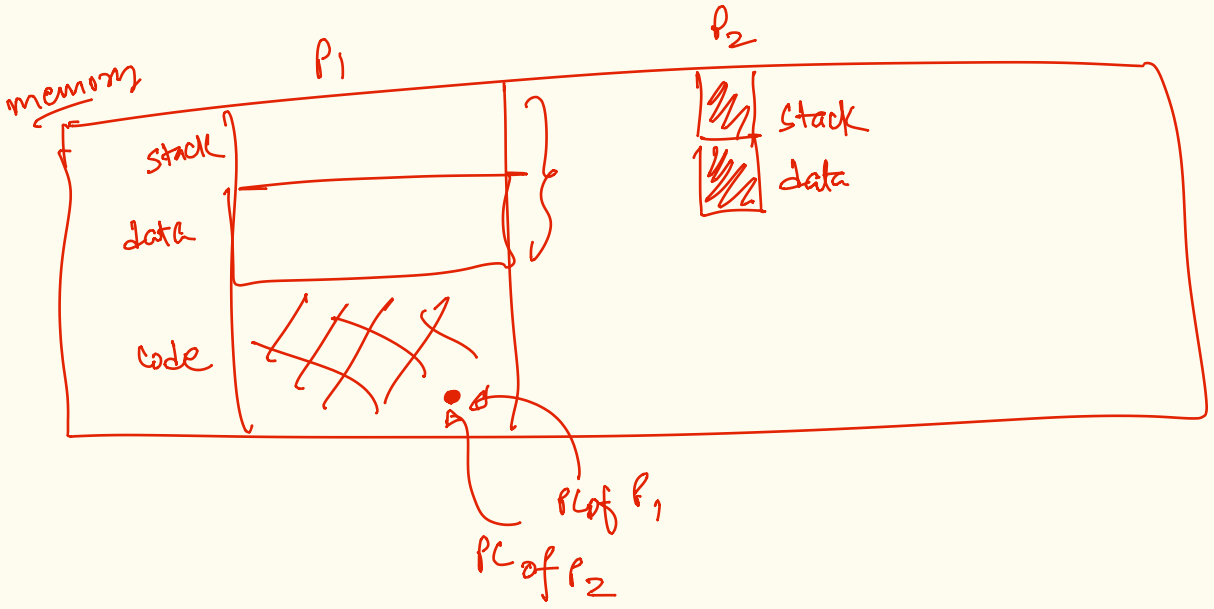
- copies all memory contents of parent process to memory of child process
  - code
  - data
  - stack



copy-on-write (CoW) optimization:

on fork

- code section is read-only
- stack is per process
- data usage is also per process



- code section not copied
- mark data/stack regions as read-only
  - only on write create a copy

// bear cost only when needed!