CS 744 Lecture 6.

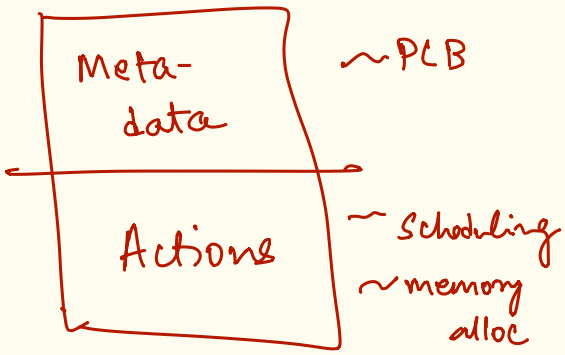
(*) the process abstraction

~ how to interact with the OS for process mgmt — setup, termination, scheduling etc.

~ the system call interface

/

fork, exec, wait, exit ...



two-box expln./view of all OS abstractions.

the OS game plan.

- on boot up / start of a m/c, OS loads itself in memory. & executes whatever it has to execute } program / logic PL ⇒ Ring 0

- needs atleast one user space process to consume its services

(#) - hand-crafts the first user process — init process PID = 1

- create a PCB ~ PID = 1
- allocate memory
- copy program to memory
- ...
- jmp to first instruction w/ PL = 3

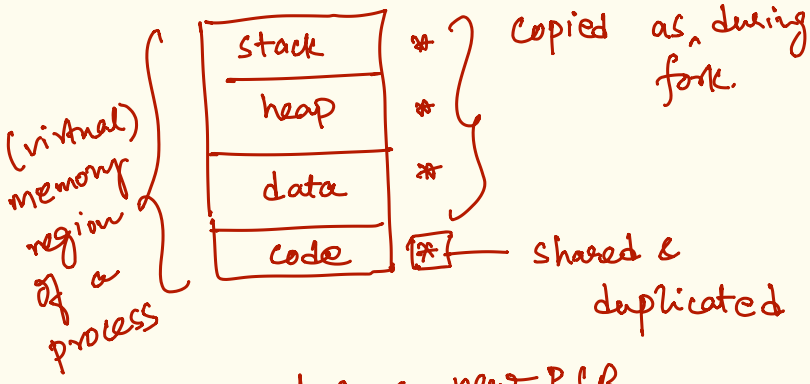
- first user space / user-mode process is available once can use all of the system call interface for OS functionality.

(*) process basic set of system calls for process related activity.

fork, exec, exit, wait

fork

- creates a new process
- duplicates the current process



- creates a new PCB
  - PID & PPID ~ parent process (calling process)

P_1 ~> fork

P_1 P_2

```
p = fork();
```

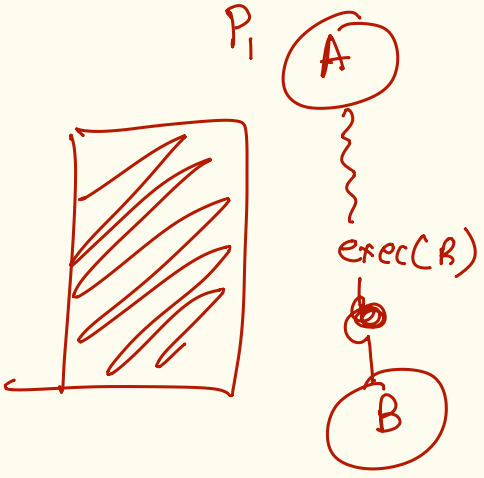
p = pid of child process in parent

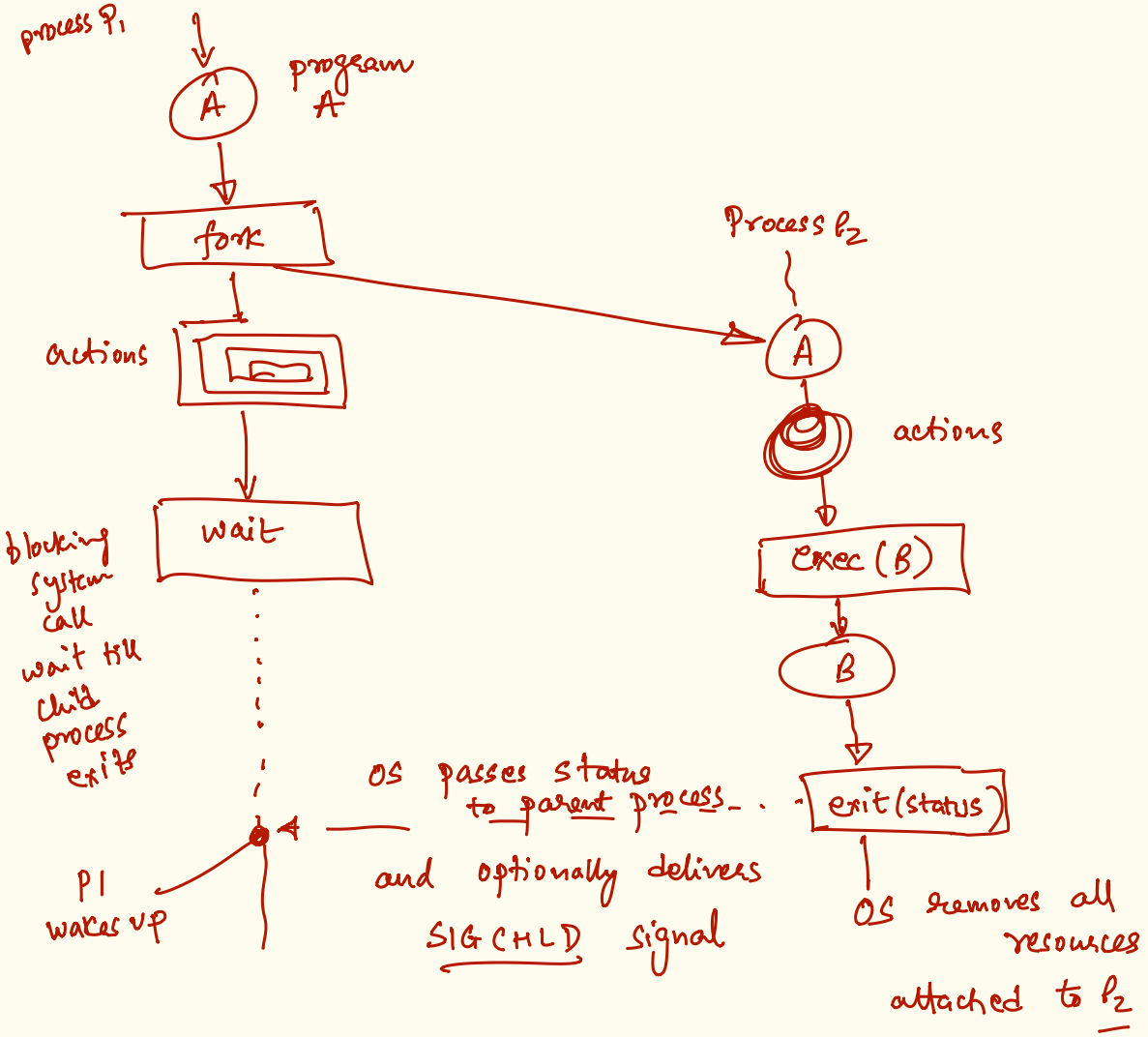
p = 0 in child process

spawn

exec

- replace / load a program for execution. into current process
- exec (programname)
- ~ loads & reset memory contents w/ new program
  - ~ copy binary into code section
  - ~ wipe out stack & heap
  - ~ setup static variables to data from program binary
  - ~ jmp to start address of program




process P_1
A
program A
fork
Process P_2
A
actions
actions
wait
blocking system call wait till child process exits
exec (B)
B
OS passes status to parent process
exit (status)
and optionally delivers SIGCHLD signal
P1 wakes up
OS removes all resources attached to P_2