

process abstraction

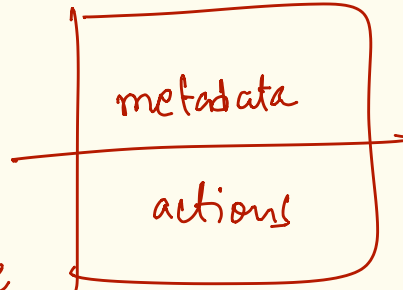
- program vs process

- PCB

pid, ppid
memory
files
,

(execution) state

OS abstractions-view

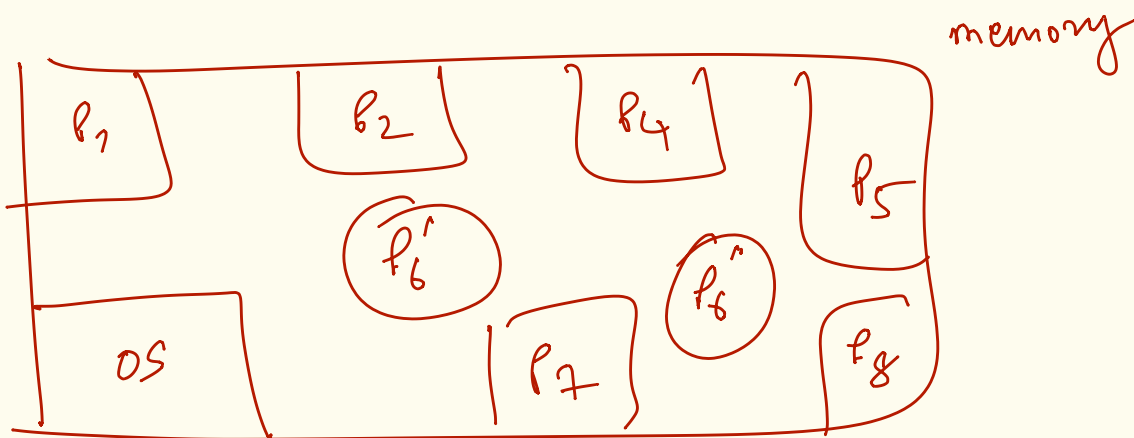


Lab2 (ongoing)

Lab3 (xv6)^{risc-v}

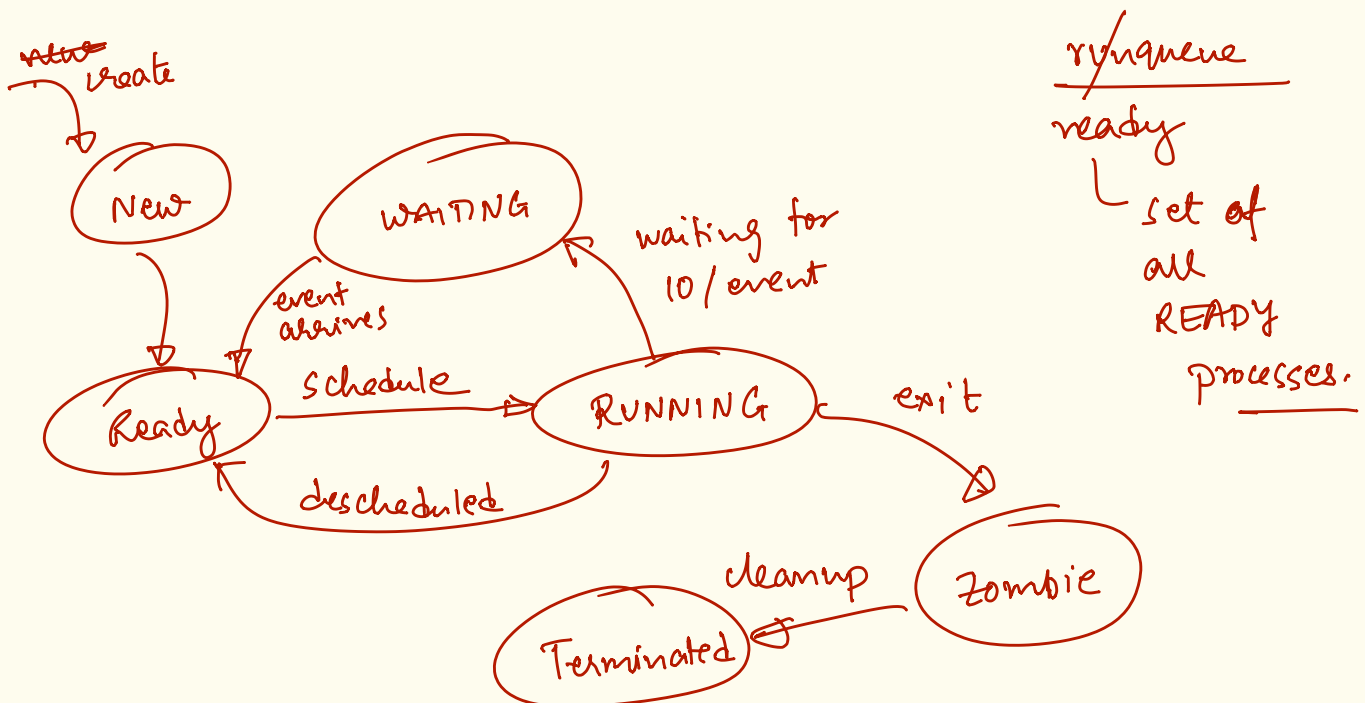
(OS-internals)

(9.31am today!)

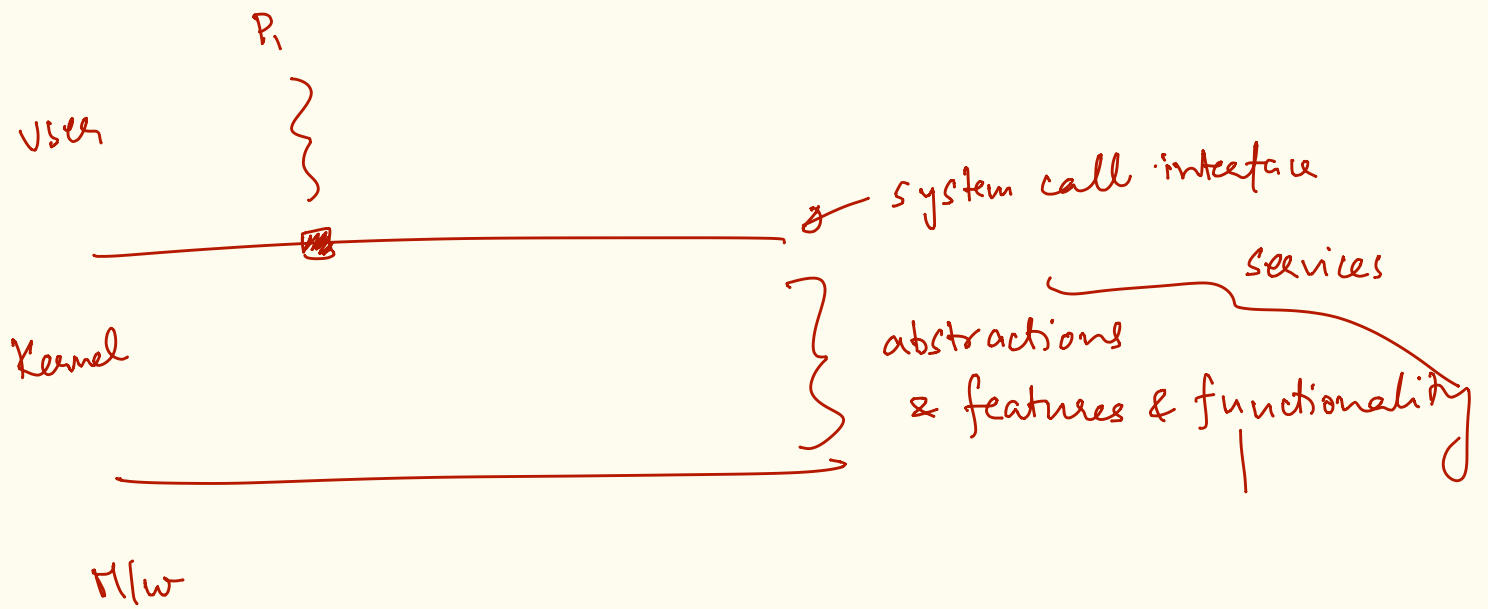


[Q]

when a process is executing on the CPU, will it always occupy the CPU?
should it (till it finishes).



ready
ready
set of all READY processes.



⑧ what is the catch? for offering / consuming OS services.

~ on boot up:

- BIOS check
 - loads the boot loader
 - loads the OS
 - OS starts execution
- load code/program in memory

~ the OS game plan

- create a handcrafted process
 - init
 - systemd
 - launchd
 - user program?
 - switches to user mode
 - jumps to start of process.
- user

- ~ allocate & initialize PCB
- allocate / reserve memory
 - copy program into memory
 - set PC to start of code & ptr to user space

⑧ Systems calls for processes

└ fork, exec, wait, exit, signal, ...

fork

- creates a new process
- duplicates the calling process
parent
 - └ create a new PCB
 - └ copy all items of current PCB to new PCB
 - └ except pid, ppid

⑨ fork returns in two processes!

- └ parent: returns pid of child process
- └ child: returns 0.

exec

- load/replace a program into /of current process
- exec (program name, path);
- load into memory of current process the program
- └ jmp to start of program.

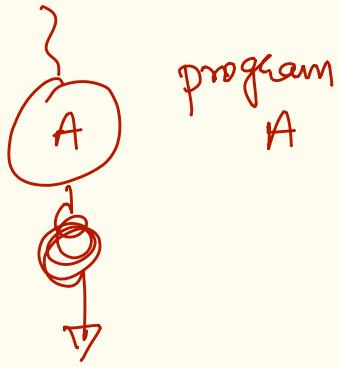
HW

Q why split-model?
fork + exec.

why not:-

spawn(program name)

process P_1



fork()



...

wait()

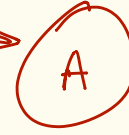
blocked ←
till
child process
terminates.

...



process

P_2



exec(B);



event!

exit()

B is done
(terminate)