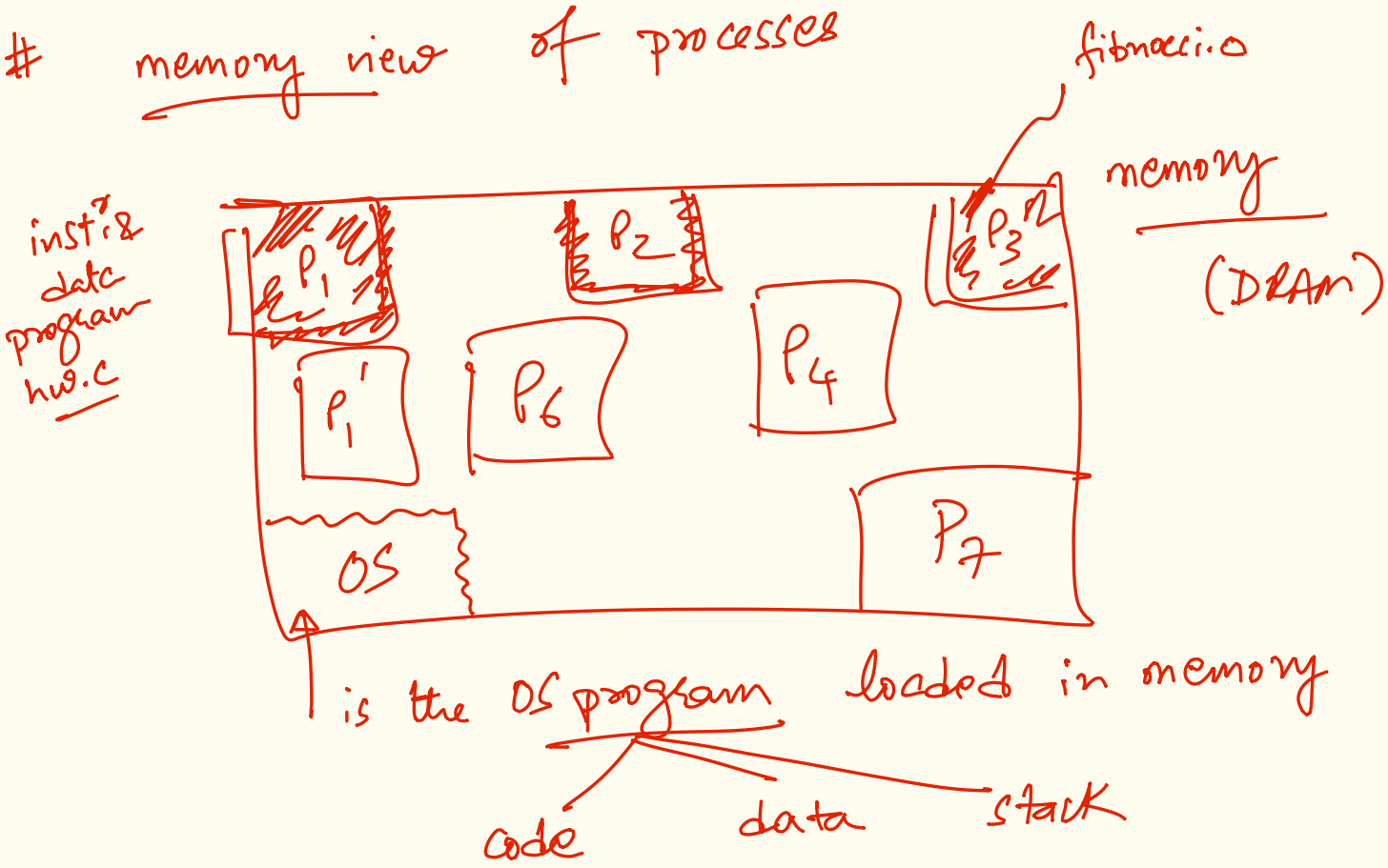
CS 219 Lecture 6

the process abstraction & the process API

memory view of processes

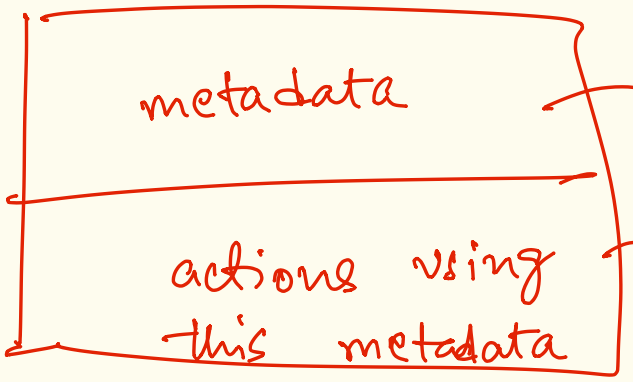


~ for every OS abstraction, the OS stores metadata for correct implementation of the abstraction.

~ for the process abstraction

- memory region / bounds
- pid, ppid
- CPU context (on switch-out)

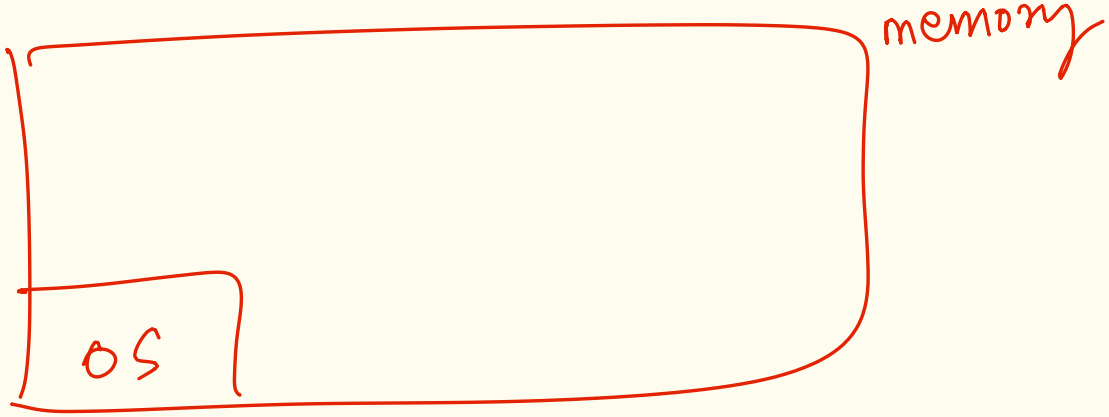
for every abstraction ~ process



PCB ~ process control block

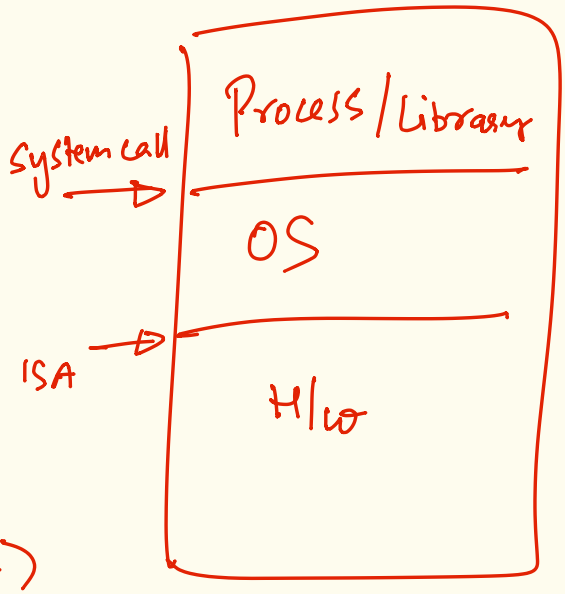
~ creating, exiting scheduling, memory allocation...

the OS game plan



fork()

- after boot
- hand craft a "default" user process (init)
- and switch to its execution.



fork

- creates (duplicates) a process
- creates a new PCB populates PCB entries
- first instr. / PC to execute in the new one process is after the fork instruction.

exec

- load binary program (code & data) into memory
- attach to process

wait

- wait till / check if child process terminates
- can capture exit status of exiting child process.

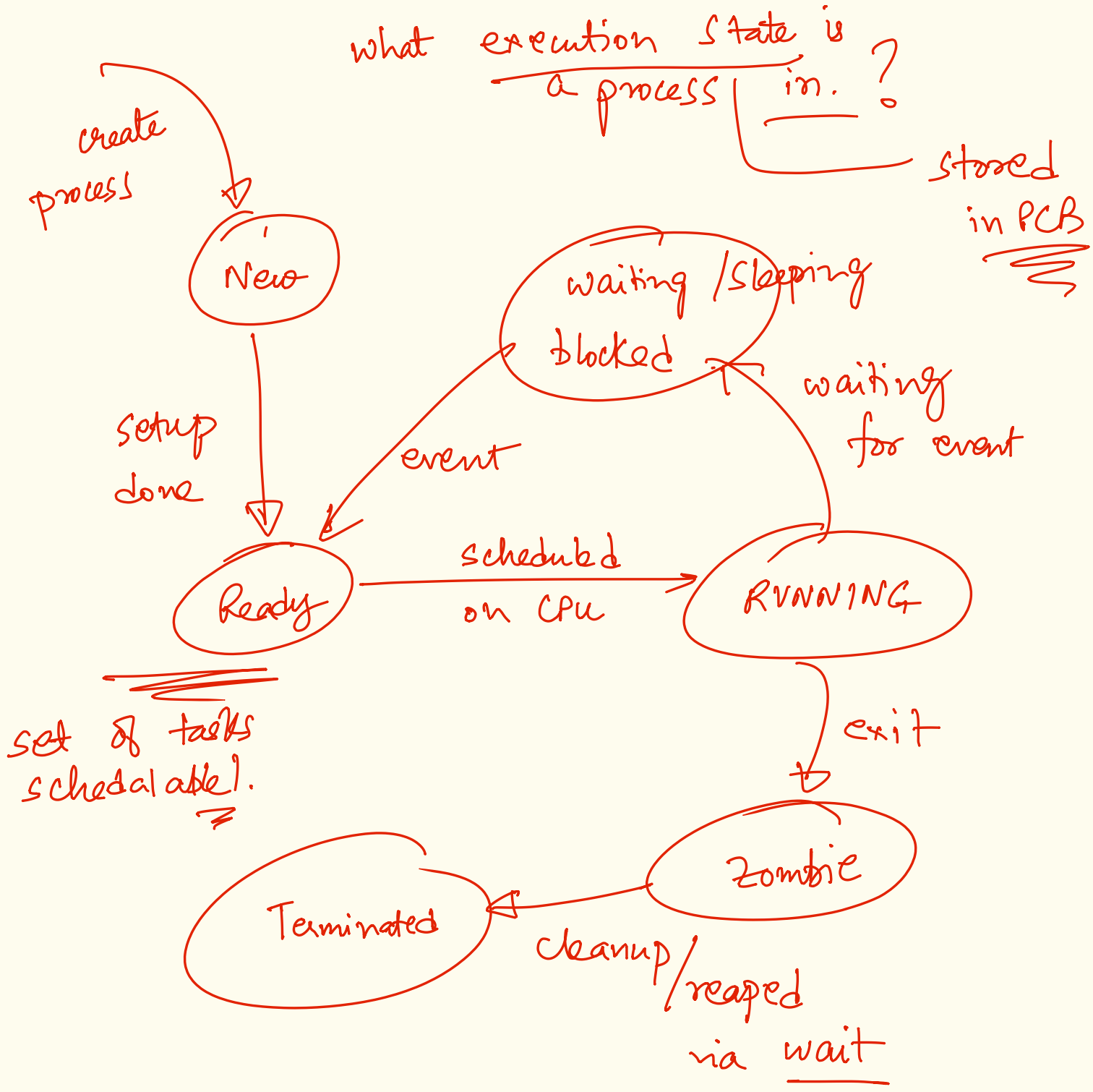
```
int a=3;
if (fork()==0)
{
    a=a+1;
    printf(a);
}
else
    printf(a);
```

fork requests a new (child) process & duplicates all state in new process

fork returns
- pid of child process in parent
- in child process returns 0.

④ child process

③ in parent process


what execution state is a process in ?
stored in PCB
create process
New
waiting / sleeping
blocked
waiting for event
setup done
event
Ready
scheduled on CPU
RUNNING
set of tasks schedulable.
exit
Zombie
Terminated
cleanup / reaped via wait