CS 347 Lecture #6

recap: process abstraction
- OS-support/ mechanisms
- usage (the process system call API)
- scheduling
- resource allocation
  - (memory)
  - files
  - ....

① System calls

fork, exec, wait, exit.

getpid, getppid, signal, ...

setpriority, getpriority, ...

* cow — optimization

② fork — duplicates a process!

exec — a short name for a set of system calls that,

- loads a new program (into a process)

Q — ??

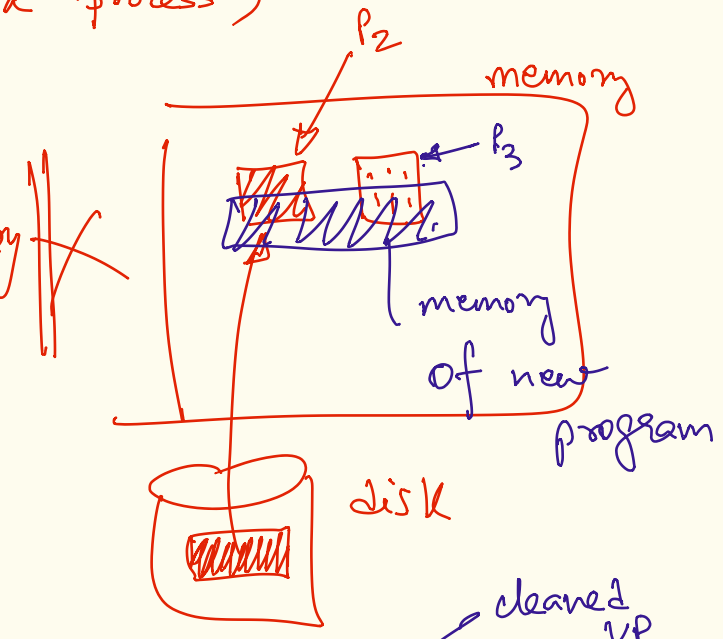
ⓐ copy binary executable program into memory

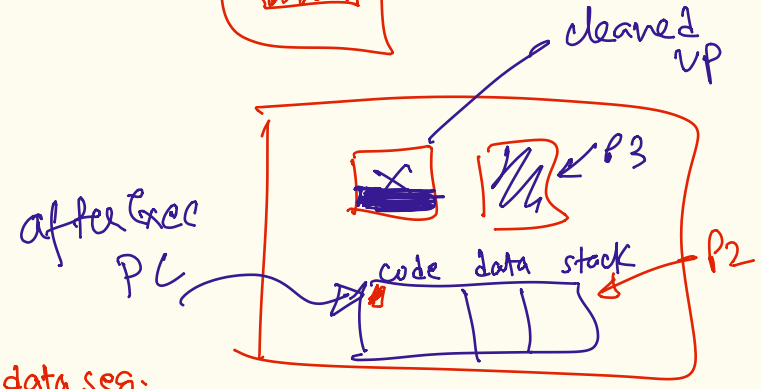
ⓑ clean up /remove/ move to free set the old memory area (if not shared)

ⓒ update the PCB for memory region info.

ⓓ clean up CPU context
- all registers (gen purpose, stack ptr. code seg, data seg.

ⓔ switch PC to first inst of new program





Q. Why split fork & exec functionality?

design a new system call that does both!

Spawn

~ fork is great for parallel processing.

(i)
- web-server
- data/file server
- data processing

(ii) fork allows for customization of the env. before program is loaded.

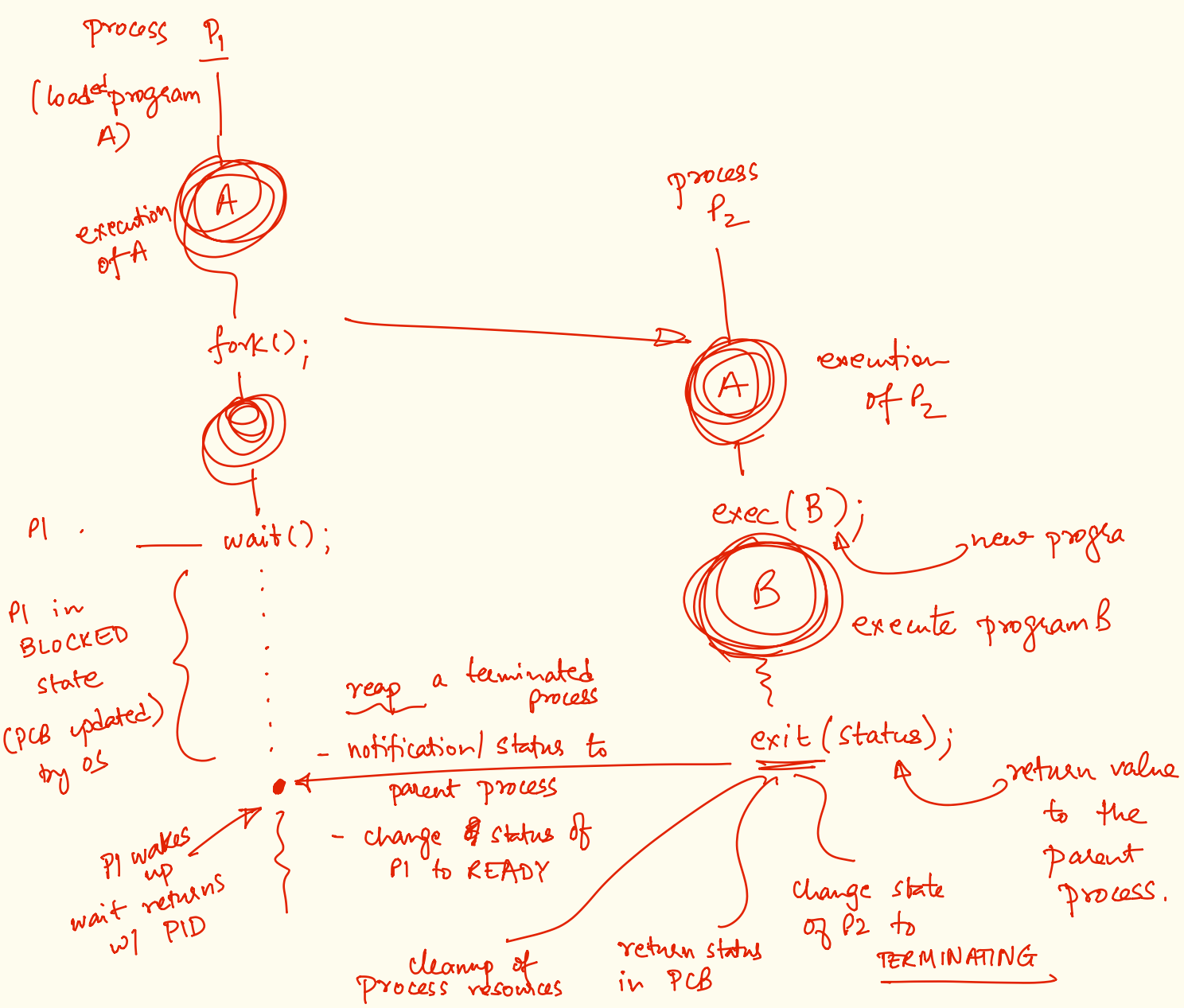
+ open files, cwd, group, priority, ...

---

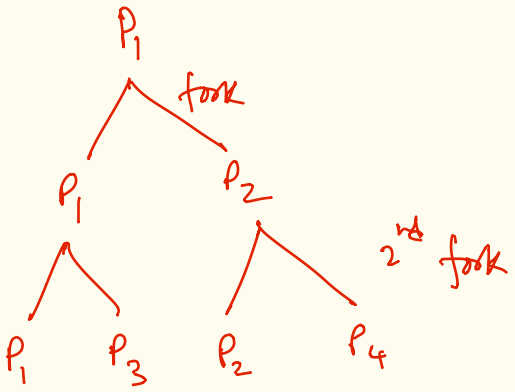
(3) wait(...) & exit(...)

- system call that pauses/blocks the parent process till child process terminates (waitpid)
- wait returns PID of the terminated child process.

- a call to terminate execution of calling process.




```
P1
int a;
fork();
fork();
wait();
```




```
int a;
if (fork() != 0)
    wait();
if( (fork() != 0)
    wait();
```