Lecture #6 CS347 18.8.2023

signals & scheduling.

recap: fork, exec, copy-on-write.

(i) signals

- OS-assisted/ s/w defined mechanism to deliver events (interrupts/ control signals to processes.
- a signal / pending signals are part of OS state!
- signals are processed before a process is scheduled on the CPU.

types of signals.

- override or not
- type of default action
  - terminate
  - dump core (memory content of process to a file)

signal handling

- default OS action
- user-mode handler per signal (via registration)

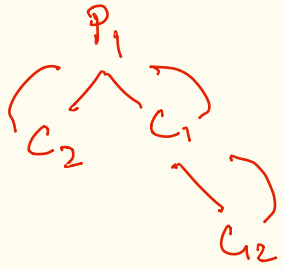
signal delivery

depends on the system calls. (KILL)

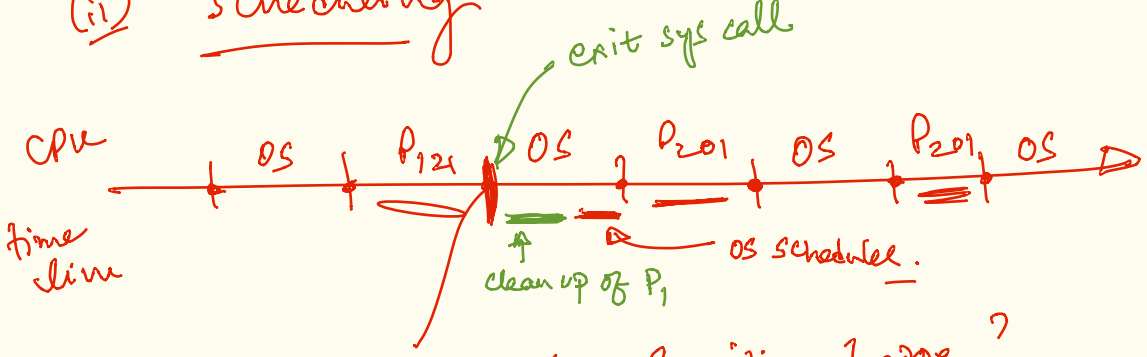
SIG KILL	9		terminate cannot override
SIG INT	2	interrupt	terminate override
SIG SEGV	~~15~~ 11		override, dump core
SIG STOP	19		cannot override change process state to blocking
SIG CONT	18		cannot override change status
SIGCHLD			can override. (signal on child process terminate) ignore

sigchld

```
void handlechildexit( ) {
    pid = waitpid(-1, &status, WNOHANGUP);
}
main() {
    signal( handlechildexit, SIGCHLD);
    fork();
    fork();
    // long running code
}
```



(ii) scheduling



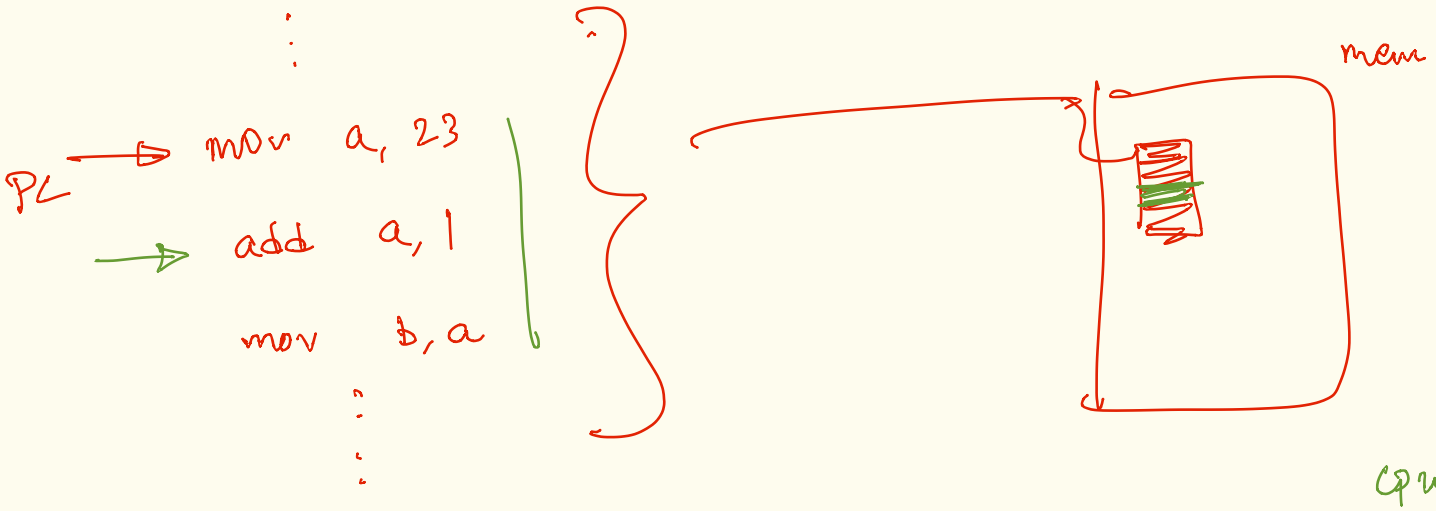
- when does this transition happen?
- how does this transition happen?
- what does the os do in its time on the CPU?

* list of actions that a scheduler has to do to switch from P_1 to P_2 ...

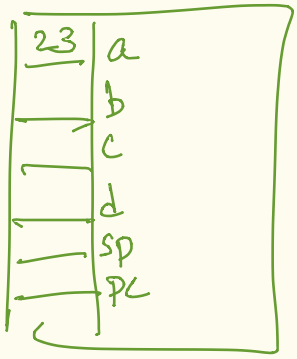
(i) an event that triggers the scheduler — interrupt / system call

(ii) context switch.
- save context of outgoing process,
- restore context of incoming process.

scheduling policy (ii) choose the next process to execute on CPU.



CPU



* Context of a process ⇒ all CPU state (registers).

(iii) scheduling mechanism

* ready queue is the set of all schedulable processes.

* multi-CPU setup
  - single ready queue needs synchronization
  - per CPU readyQueue