CS 219 Lecture 9

signals & the system call mechanism

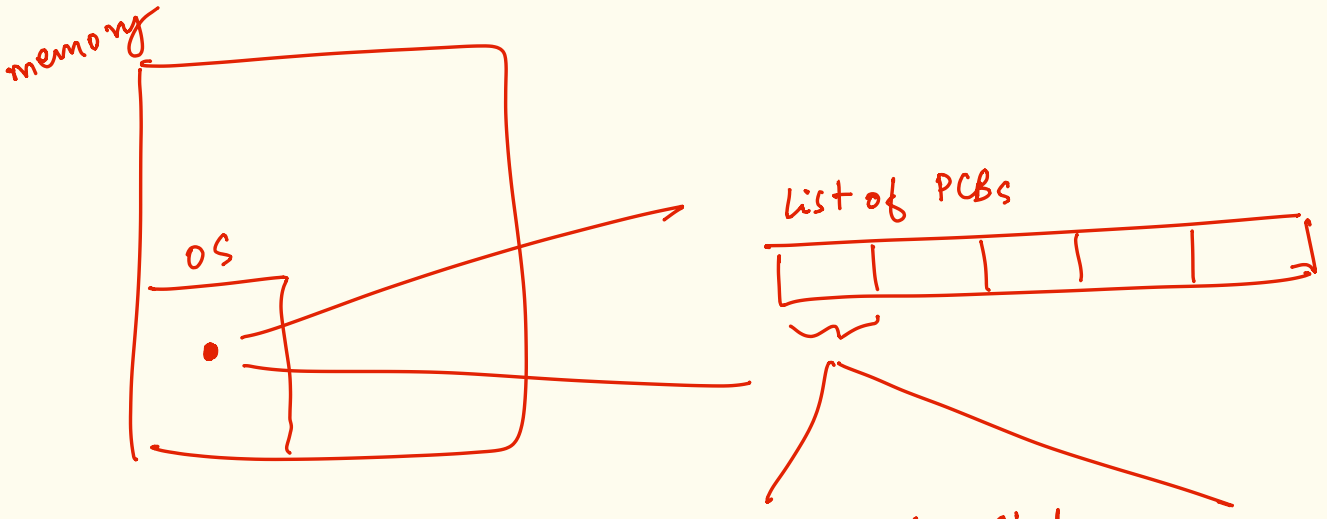
28th - Quiz1 830 am

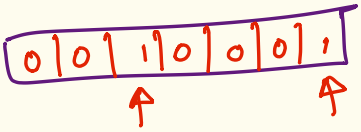
29th Lab - Quiz1

2pm

signals!

- a primitive for IPC (inter-process (control) communication) ~ interruption!
- (pending) signals part of PCB
- signals can be blocked, ignored, handled/caught.
- signals preserved across fork/exec.
  - handling
    - signal to parent process also delivered to child process.



- pid, state...
- ptr. to parent PCB
- open files
- memory info
- signals 
- context
- ....

- delivery signals
  - manipulating the signal object/variable in PCB of target process.
- on schedule the signal state is checked for handling it.

		block	default action	register handler
SIGKILL	9	X	terminate	
SIGINT	2	✓	terminate	
SIGSEGV	11	✓	terminate ~ dump core	
SIG STOP	19	X	BLOCKED / INTERRUPTED	
SIG CONT	18	X	READY	
SIG CHLD	17	✓	on child exit deliver to parent	
SIGTERM	15	✓	terminate	

deliver signals using Kill system call

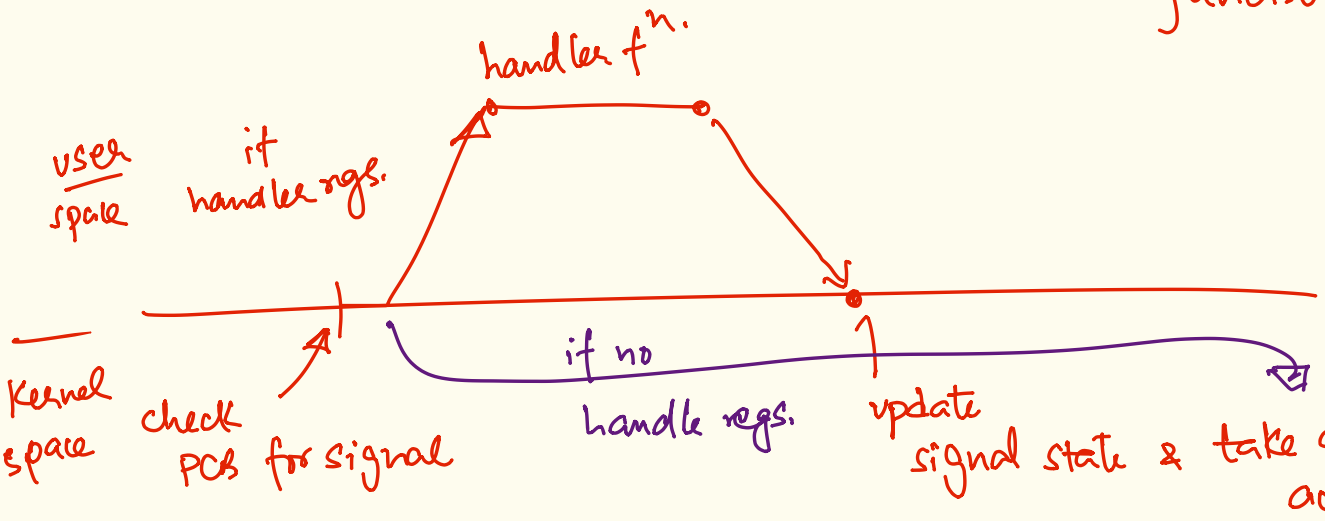
Kill (pid, signal number)

tool $ Kill pid signum

signal (signal no., function);

↑ System call to register signal handler

↑ address of a signal handling function.

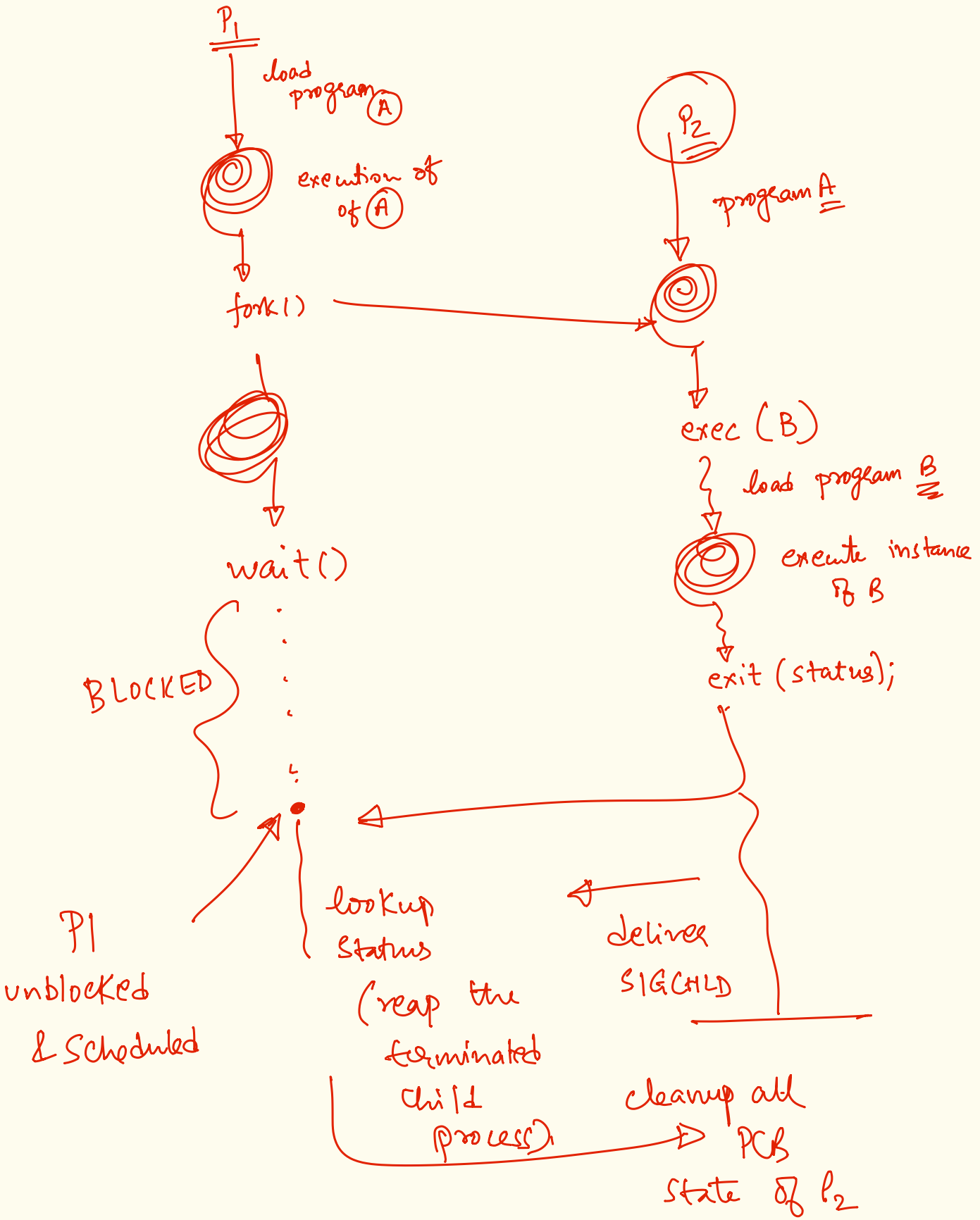


e.g.:

```
void sighandle () {
    cpid = wait (NULL);
        if (cpid == ____)
            ____
    }   else
            ____

int main () {
    ...
    signal (SIGCHLD, sighandle);
pid = fork ();
    if (pid > 0) {
            while (1)
                attempt world peace ();
    }
    else  while (a few times)
            attempt world peace ();
}
```

waitpid(~~pid~~, W.NOHANG ...)


P1
load program A
execution of A
fork()
P2
program A
exec (B)
load program B
execute instance of B
exit (status);
wait()
BLOCKED
P1 unblocked & Scheduled
lookup status (reap the terminated child process)
deliver SIGCHLD
cleanup all PCB state of P2