CS 744 Lecture 7 cont.

* signals, file model, pipes, scheduler (?)

① Lab Exam 1
28. Aug. Wed.
830 am
SL1, SL2, SL3
* keep a watch for email w/ details.

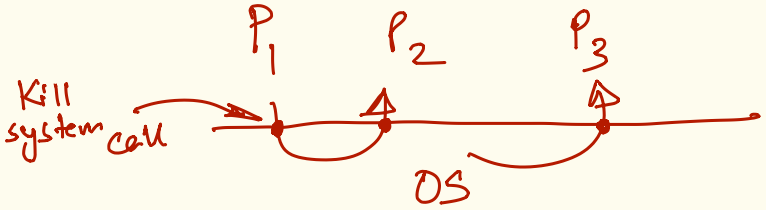
② Lab 2
due today.

③ Lab practice problems by EOD today

① signals

- mechanism to convey events during its execution
  - either initiated by the OS or another process.
- similar to hardware interrupts,
  - ~~but~~ handled by the OS completely in software.

mechanism



- user-mode process (P_1) issues a system call (kill) with pid & signum (signal identifier)
- OS stores the signal for the intended process (P_2) in its PCB (pending signal list)
- before scheduling / switching to user mode for process (P_2) OS checks pending signal state
  - if signal pending
    - take default action
    - jump to user-space signal handler.

user-space fn

* signal (fnptr, signum) ← via the signal system call

signal handler is registered by the process during its execution

SIGKILL	9	default action terminate process
SIGINT	12	
SIGSEGV	15	
(*) SIGCHLD	17	— default action ignore
SIGSTOP	19	— move to WAITING state
SIGCONT	18	— move to READY state

↑ #define ↑ signal number

SIGSTOP and SIGCONT: cannot be ignored or registered via handler.

$ Kill -9 <pid>

P_1 —forks→ P_2 P_1 { P_2 {

Case 1

P_1

```
{
fork()
}
wait( );
```

blocking call →

Case 2

P_1

```
{
signal(sigchldhandler, 17);
}
fork();
```

switched out

sigchldhandler()
{
pid = wait(&status);

after signal handler
}
return

Should switch back here on next scheduling on CPU

if SIGCHLD pending

File IO model

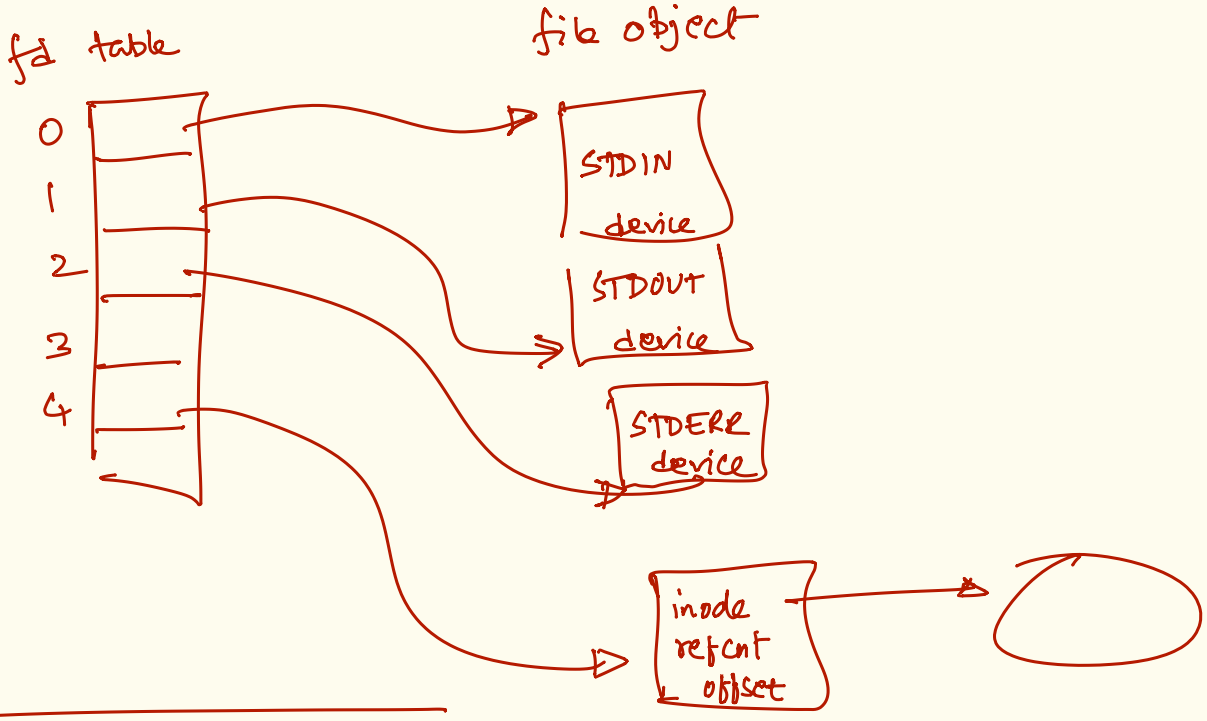
non-negative number

~ in unix like systems the file descriptor primitive is used interface (access for IO commands) for a variety of mechanisms / abstractions — regular files
socket
pipes
FIFO
devices

~ system calls.

- open, close
- read, write, lseek
- dup, dup2, pipe

~ each process inherits by default 3 file descriptors



$ cat ⤵ reads STDIN
write STDOUT

$ cat helloworld.txt ——

shell

fork();

child process

close (0);
open(filename);
exec ("cat");