Process

|

Classification — Types of Processes

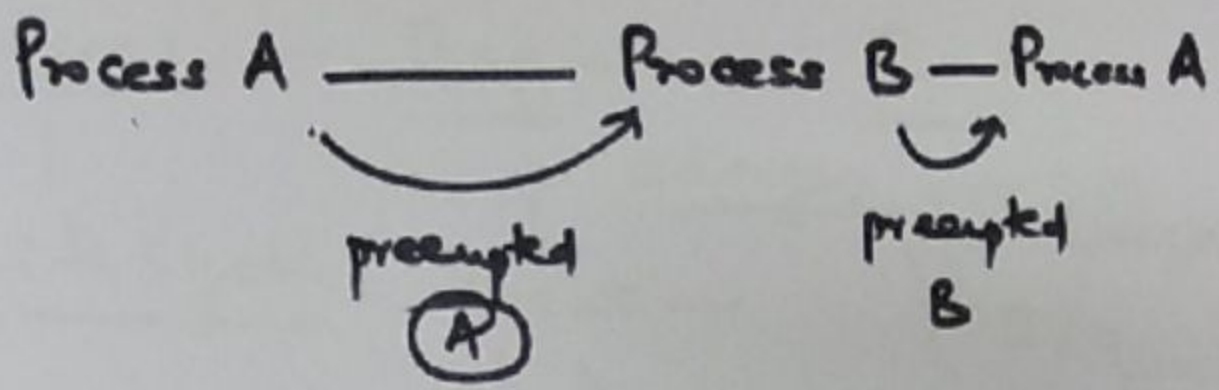
could be used to gather

1. User level process (low)
   vs.
   Kernel level process (high)

2. CPU-bound process
   CPU-intensive process
   vs.
   I/O-bound process
   I/O-intensive process
   | Type of activity.

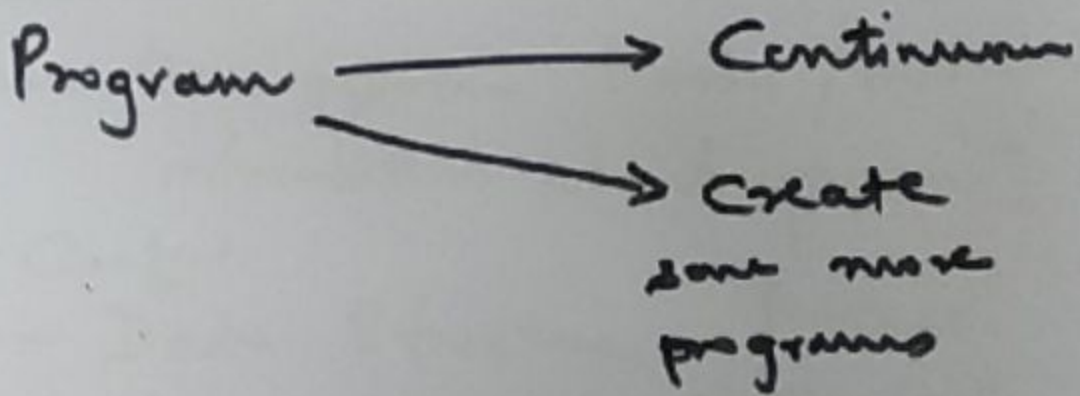
3. Cooperative process
   vs.
   Non-Cooperative process (Independent)
   | Relation both processes.

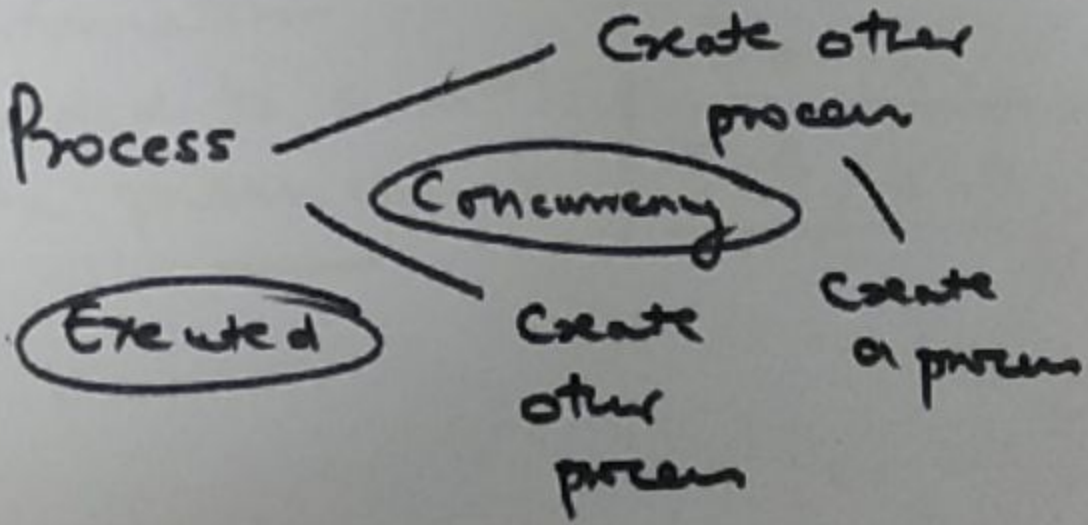
4. Preemptive vs Non-preemptive

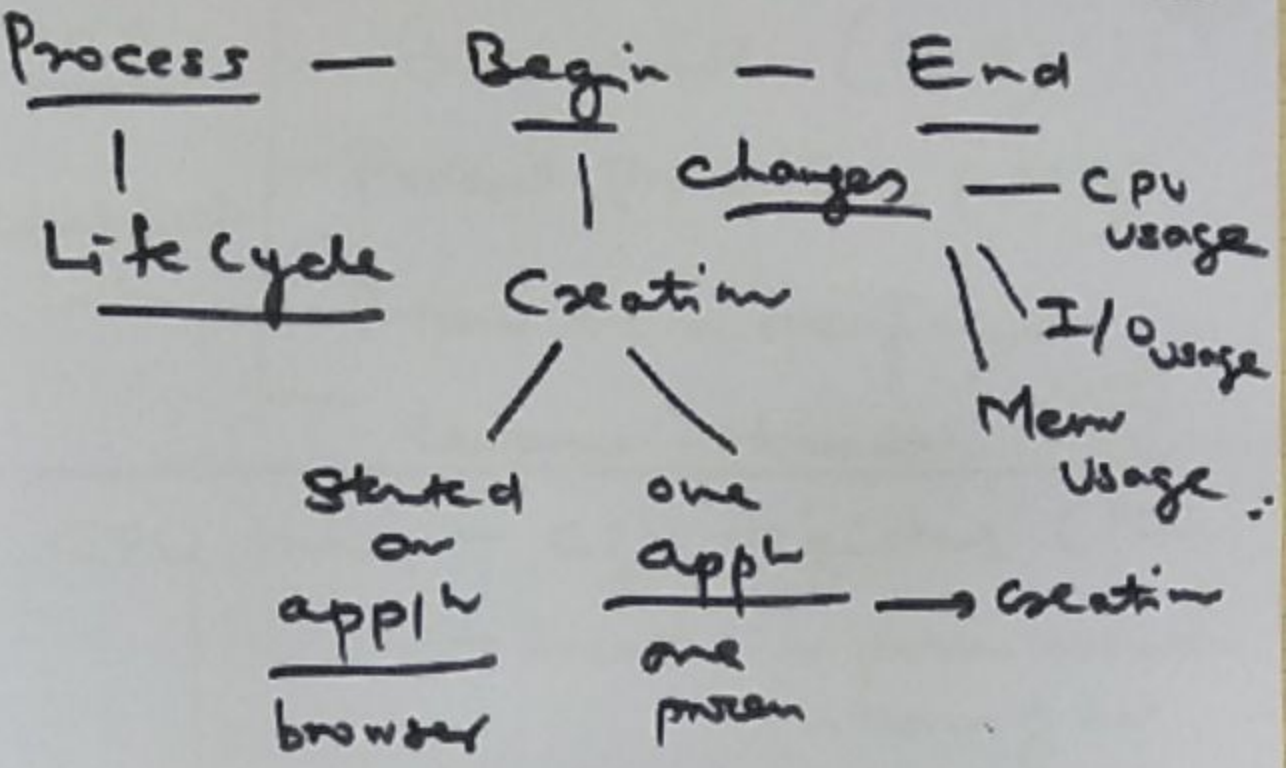


Process ——— Concurrency

Two or more processes could together







How the system tracks / maintains information about prosen?

Context
— Data Structure — Process Control Block (PCB)

Information needed to control the prosen execution

TCB / JCB

PCB

Identification info.
- Process ID (PID)
- Parent process ID (PPID)
- Process for a specific user (UID)

Resource information

CPU info.
- CPU registers (PC)
- location - process execution in terms of inst.

Scheduling — order of process execution

Mem usage — location of process

Secondary storage usage ..

I/O devices ...

Files — used/opened ...

For every process — PCB

Process Creation — how?

System call — privileged operation (Not in user mode)

fork()

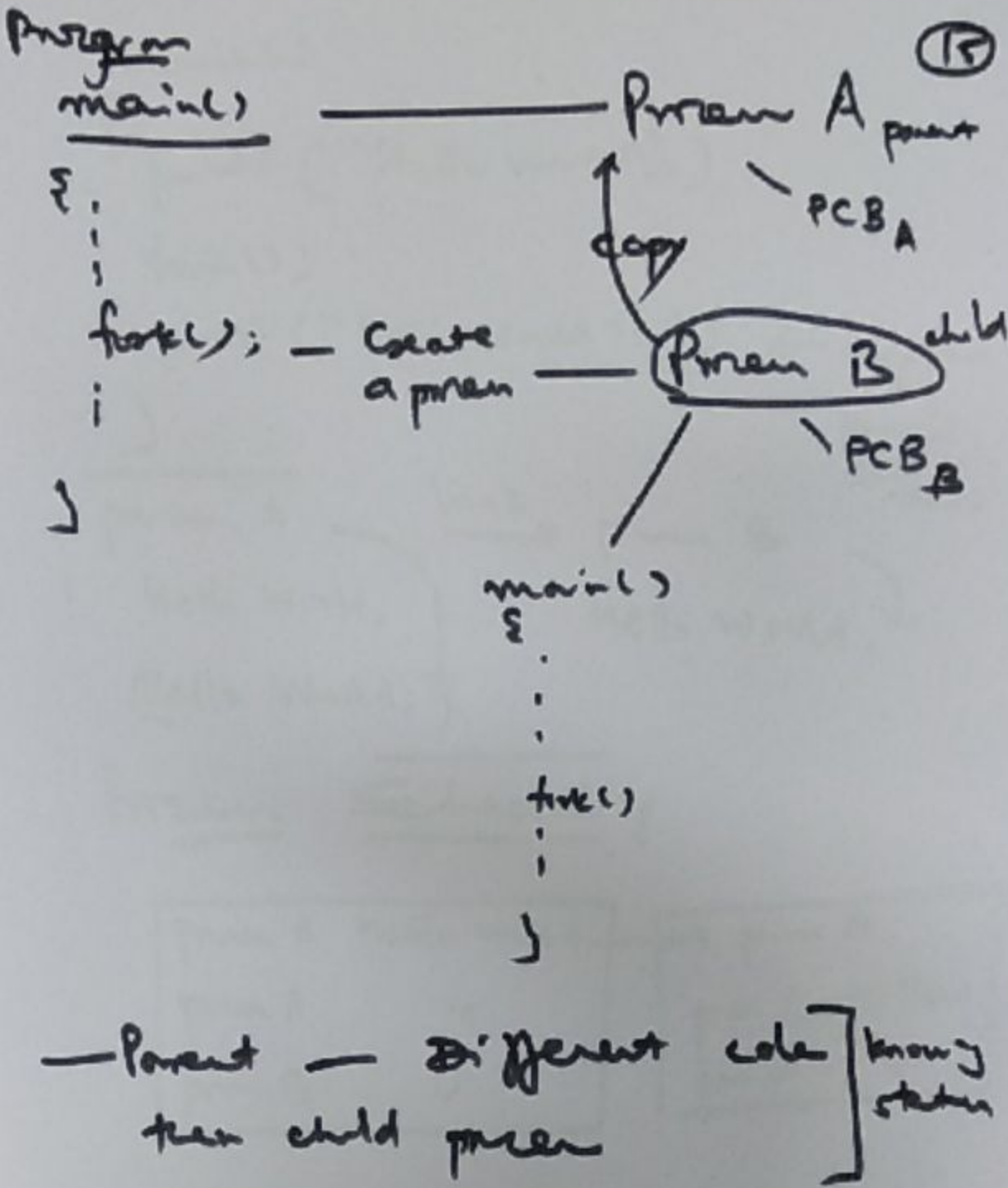
initiated / called in user mode

after only — kernel mode

```
main()
{
    ....
    User mode
    System call (user mode)
    ....
}
```

kernel

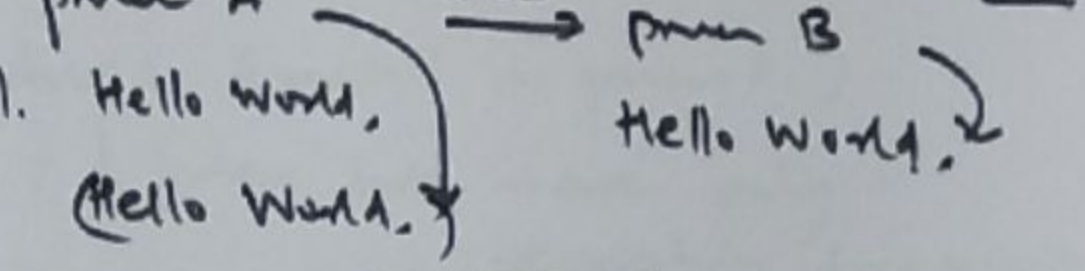
fork() — Process Creation



— Parent — Different code — know status
than child proc

```
main()
{
1.  printf("Hello world\n");
    fork();
    printf("HelloWorld\n");
}
```

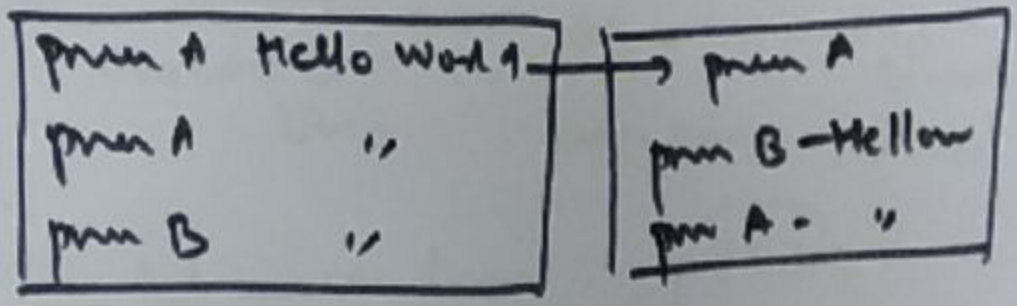
← executed by parent child

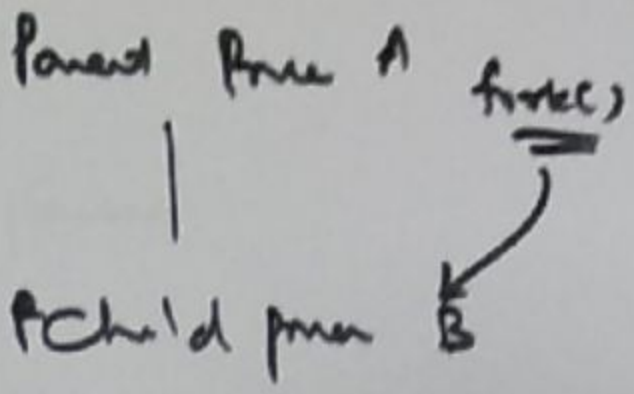


1. Hello World.
   Hello World.

Hello World.

Process Scheduling



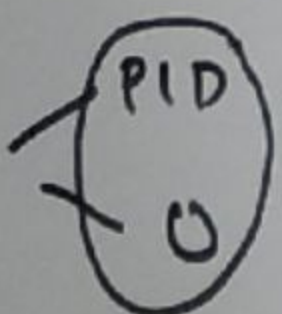



```
main()
{
    5 LOC
```

6. fork() (executed in parent A)
7 Line : ↓ below fork — one only remaining line executed)
}

Parent A	Parent B
execute 13 lines	copy — has 13 lines execute — 7 lines

fork() — return status



— code.
if ... else

```
if (PID)
    : parent
else
    : child.
)
```

if (PID)

8.
9. | parent
10.
else

11.
12. | child.
13

Prem A

1- 6, 8- 10

Prem B

11 — 13

— Invoke one appln from another
system call

- Concurrency