File

File : collection of data/characters

File = Data + Metadata

Data ↓ Actual Content

Metadata ↓ Information about file ↓

1) type
2) permissions
   r-read, w-write, x-execute
   - user/owner, group, other
3) link count
4) user/owner, group
5) size
6) timestamps
   - create, modify, access
7) name

Types :

1) regular file (-)
2) directory file (d)
3) link file (l)
4) pipe file (p)
5) socket file (s)
6) character special file (c)
7) block special file (b)

Permissions :

$\underbrace{rw-}_{\text{user/owner}} \; \underbrace{rw-}_{\text{group}} \; \underbrace{r--}_{\text{others}}$

rw-	rw-	r--
110	110	100

(664)

g+w, o+r

rwx	r--	---
rwx	rw-	r--
111	110	100

764

Command line arguments

1) int main(void)

2) int main(int argc, char *argv[])

argc - count of cmd line args

argv - array of cmd line args

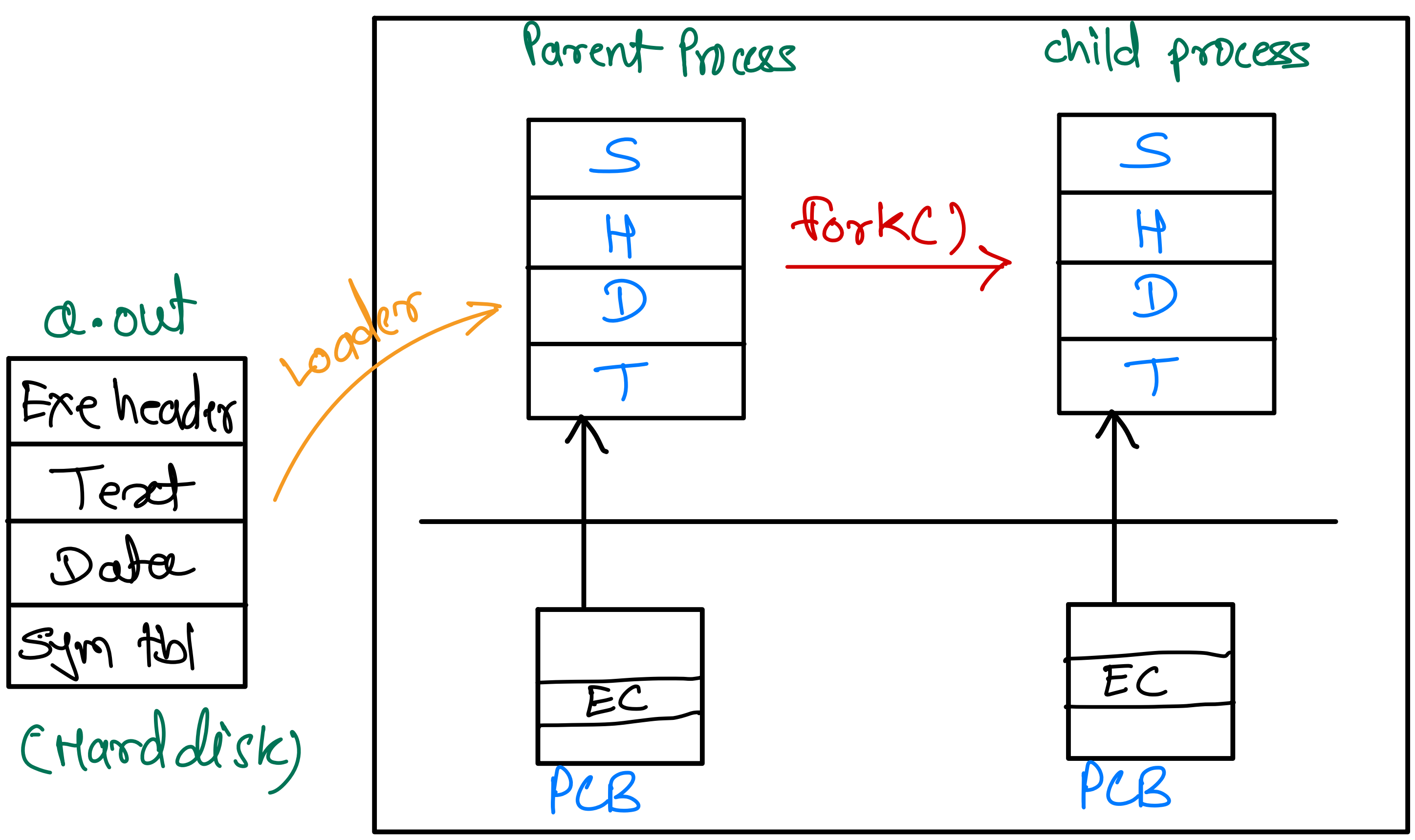
argv[0] — name of program

3) int main(int argc, char *argv[], char *envp[])

envp - array of environment variables

- main is called as "callback function"

Fork


a.out
Exe header
Text
Data
Sym tbl
(Hard disk)
Loader
Parent Process
S
H
D
T
fork()
child process
S
H
D
T
EC
PCB
EC
PCB

Exec

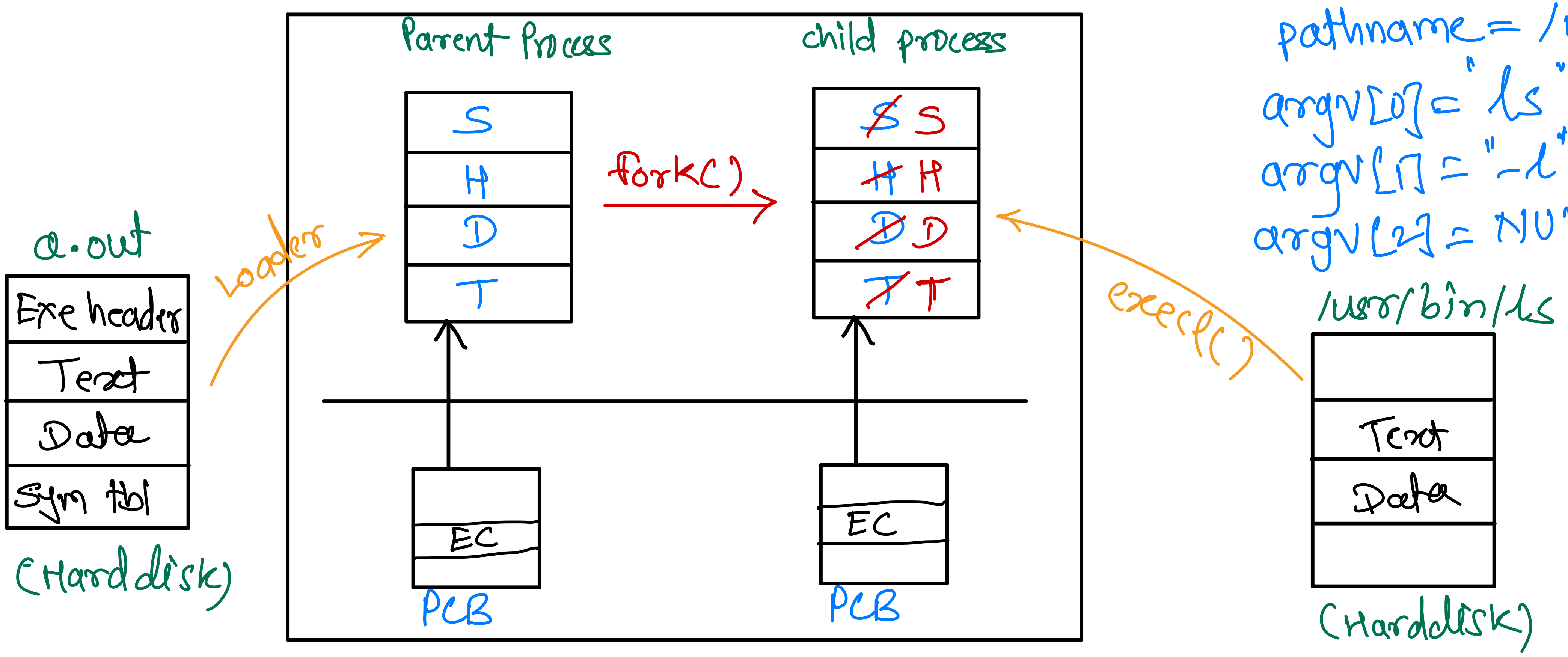
ls -l

pathname = /usr/bin/ls

argv[0] = "ls"

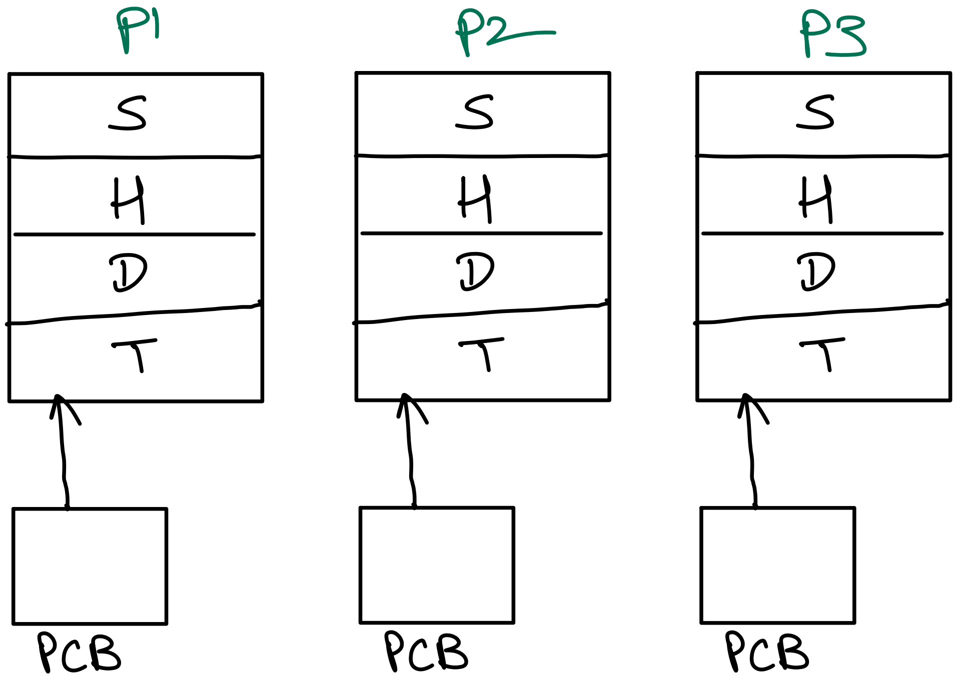
argv[1] = "-l"

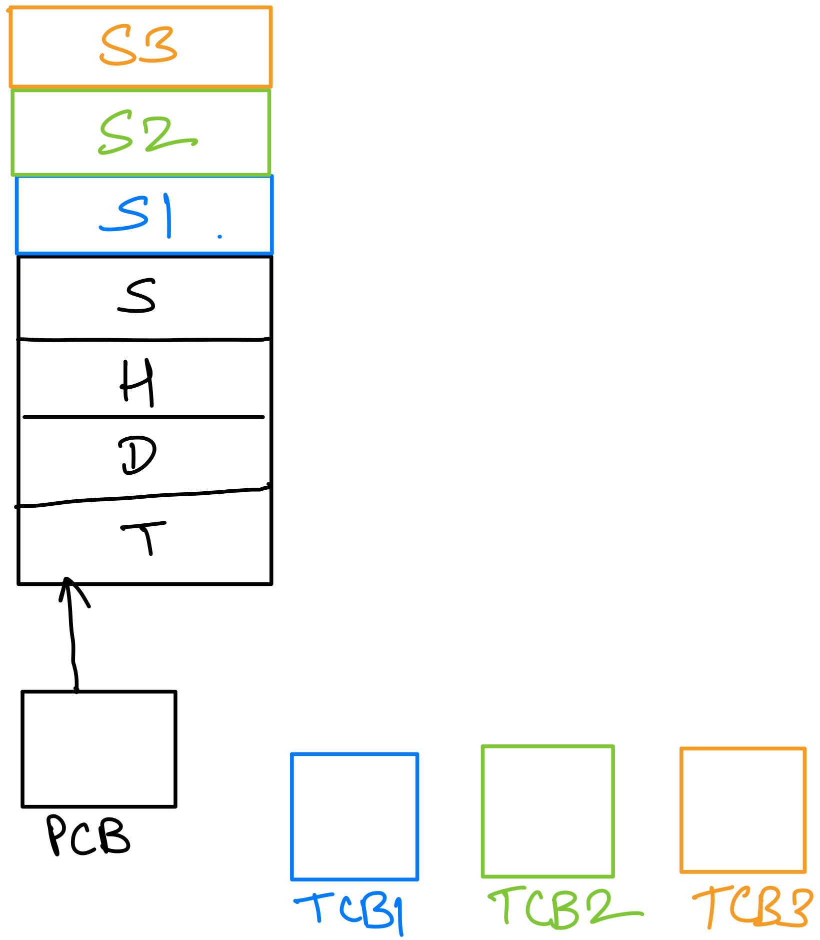
argv[2] = NULL



Process Vs Thread

Thread : Light weight process





Semaphore

- internally a counter

Operations:

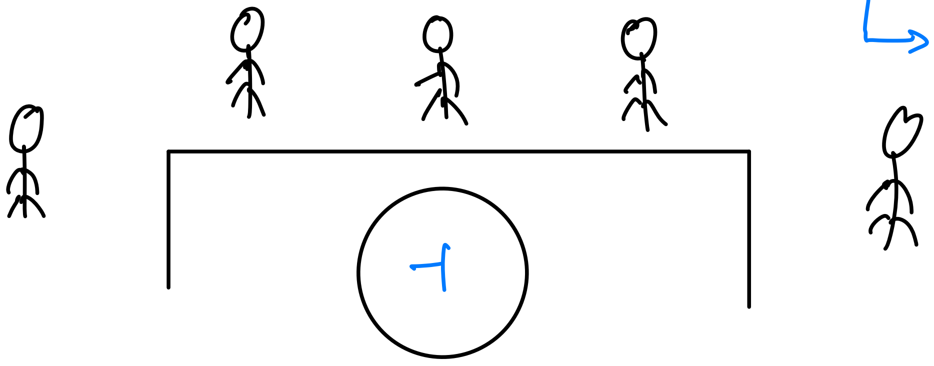
1) Dec/wait/P
   - counter --
   - if counter < 0, then block calling process

2) Inc/post/V
   - counter ++
   - if processes are blocked on this semaphore, then wakeup one

Semaphore functions:

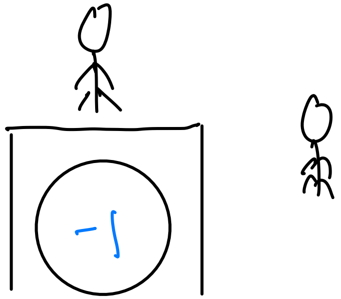
1) sem_init()
2) sem_wait()
3) sem_post()
4) sem_destroy()

Counting Semaphore



→ no. of available resources
→ no. of processes waiting for resource

Binary Semaphore



10
num

~~1~~ ~~0~~ -1
s

```
Thread1:          sem_init(s)         Thread2:

for(i=0; i<10; i++)                  for(i=0; i<10; i++)
{                                    {
    sem_wait(s);                         sem_wait(s);
    num++;                               num--;
    sem_post(s);                         sem_post(s);
}                                    }
                  sem_destroy();
```

Mutex

- Mutual Exclussion (one at a time)
- it is same like binary semaphore
- mutex operations – lock() , unlock()
- process who locks the mutex becomes owner of that mutex
- only owner can unlock() the mutex

~~0~~ ~~1~~ 0

```
Thread1                     Thread2
                            sem_wait( );
pf("sunbeam");              pf ("pune");
sem_post()
```

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com