Sunbeam Institute of Information Technology
Pune and Karad

Module - Embedded Operating System

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

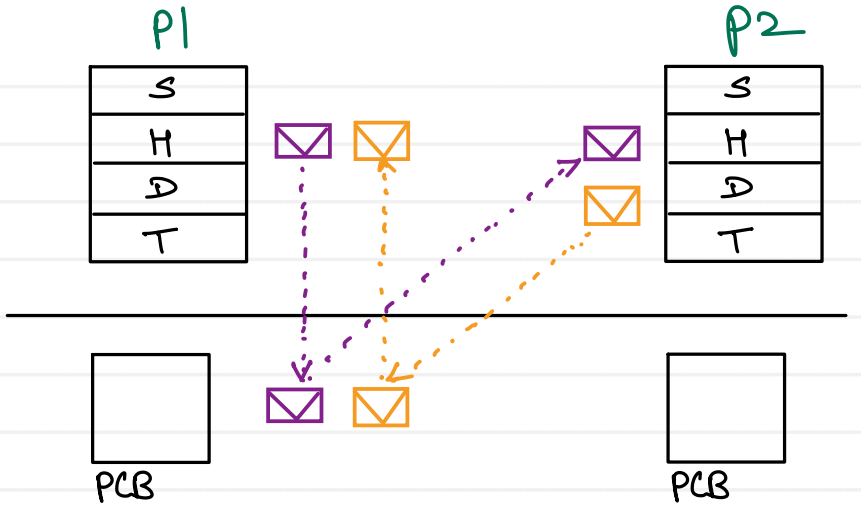
Message queue

- UNIX sys V R4 (1980)
  - 1) msg queue
  - 2) shared memory
  - 3) semaphore

  } IPC

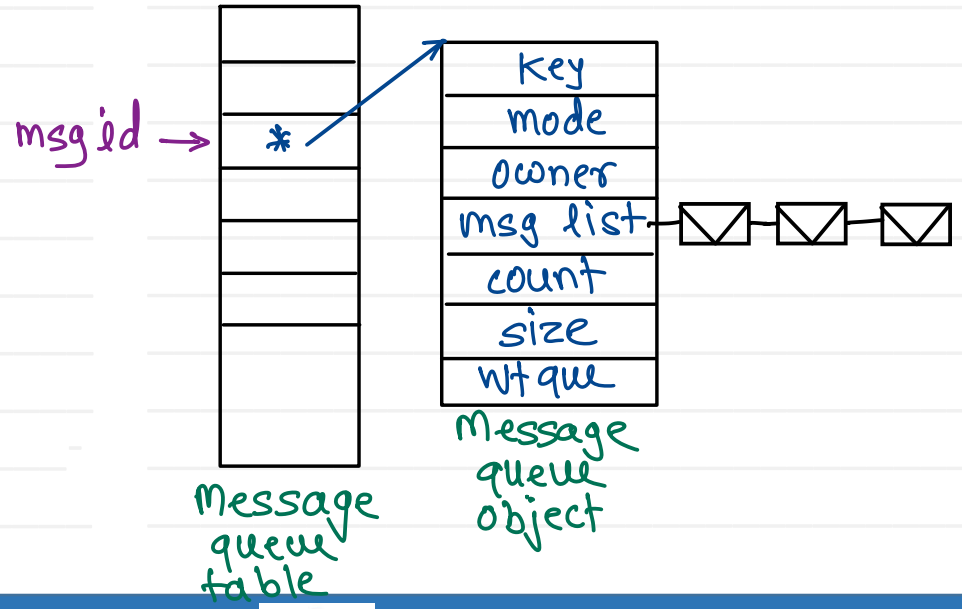
- bidirectional
- packet based (struct)

To see status of IPC
```
cmd> ipcs
cmd> ipcs -m
cmd> ipcs -q
cmd> ipcs -s
```
To remove IPC
```
cmd> ipcrm option key/id
```



System calls :

1) msgget() – to create a msg queue
2) msgsnd() – to send msg in msg queue
3) msgrcv() – to receive msg from msg queue
   - blocking
4) msgctl() – to control the msg queue
   - delete



Message queue

```
struct expr {
    int op1, op2;
    char opr;
};
```

```
struct msg1{
    long mtype;
    struct expr exp;
};
```

Process1

```
main() {
    msgid = msgget(KEY, IPC_CREAT|0600);
    pf("Enter op1 opr op2:");
    scanf("%d %c %d", &op1, &opr, &op2);
    struct msg m1 = {11, op1, opr, op2};
    msgsnd(msgid, &m1, sizeof(struct expr), 0);
    struct msg m2;
    msgrcv(msgid, &m2, sizeof(struct expr), 22, 0);
    pf("result = %d", m2.op1 );
    msgctl(msgid, IPC_RMID, 0)
}
```

process2;

```
main() {
    msgget(KEY, IPC,CREAT|0600);
    struct msg m1; int res;
    msgrcv(msgid, &m1, sizeof(expr), 11, 0);
    if(m1.opr == '+')
        res = m1.exp.op1 + m1.exp.op2;
    :
    m1.exp.op1 = res; m1.mtype = 22;
    msgsnd(msgid, &m1, sizeof(expr), 0);
}
```

sleep and wakeup

Block / sleep / wait

1. get PCB of process
2. remove pcb from ready queue
3. pcb state = blocked
4. add pcb in waiting queue

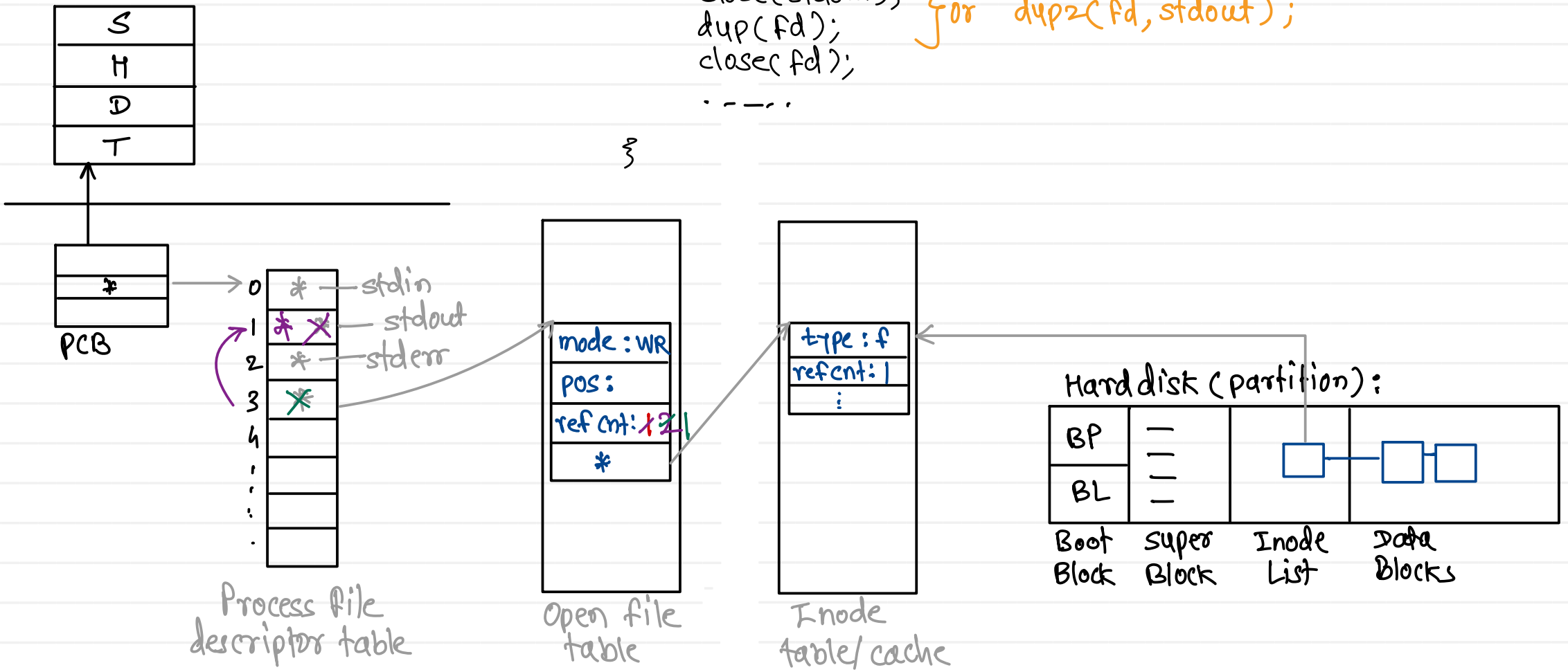
Wakeup

1. remove pcb from waiting queue
2. pcb state = ready
3. add pcb in ready queue.

Redirection

```
main() {
    fd = open("output.txt", O_CREAT|O_TRUNC|O_WRONLY, 0644);
    close(stdout);   } or dup2(fd, stdout);
    dup(fd);
    close(fd);
    ....
}
```

./program.out > output.txt

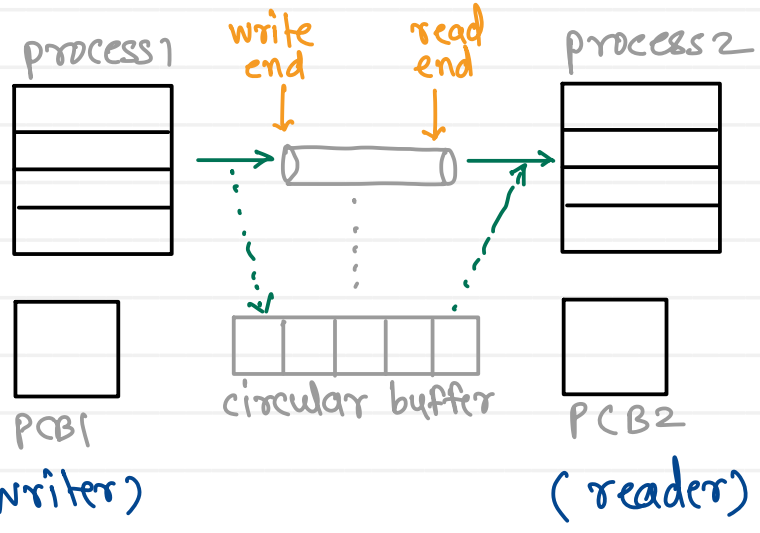


Pipe

- pipe is an IPC mechanism
- unidirectional
- stream based
- every pipe is internally a circular buffer

cmd1 | cmd2

process1 | process2



- to create a pipe,

```
int arr[2];
int pipe(arr);
```

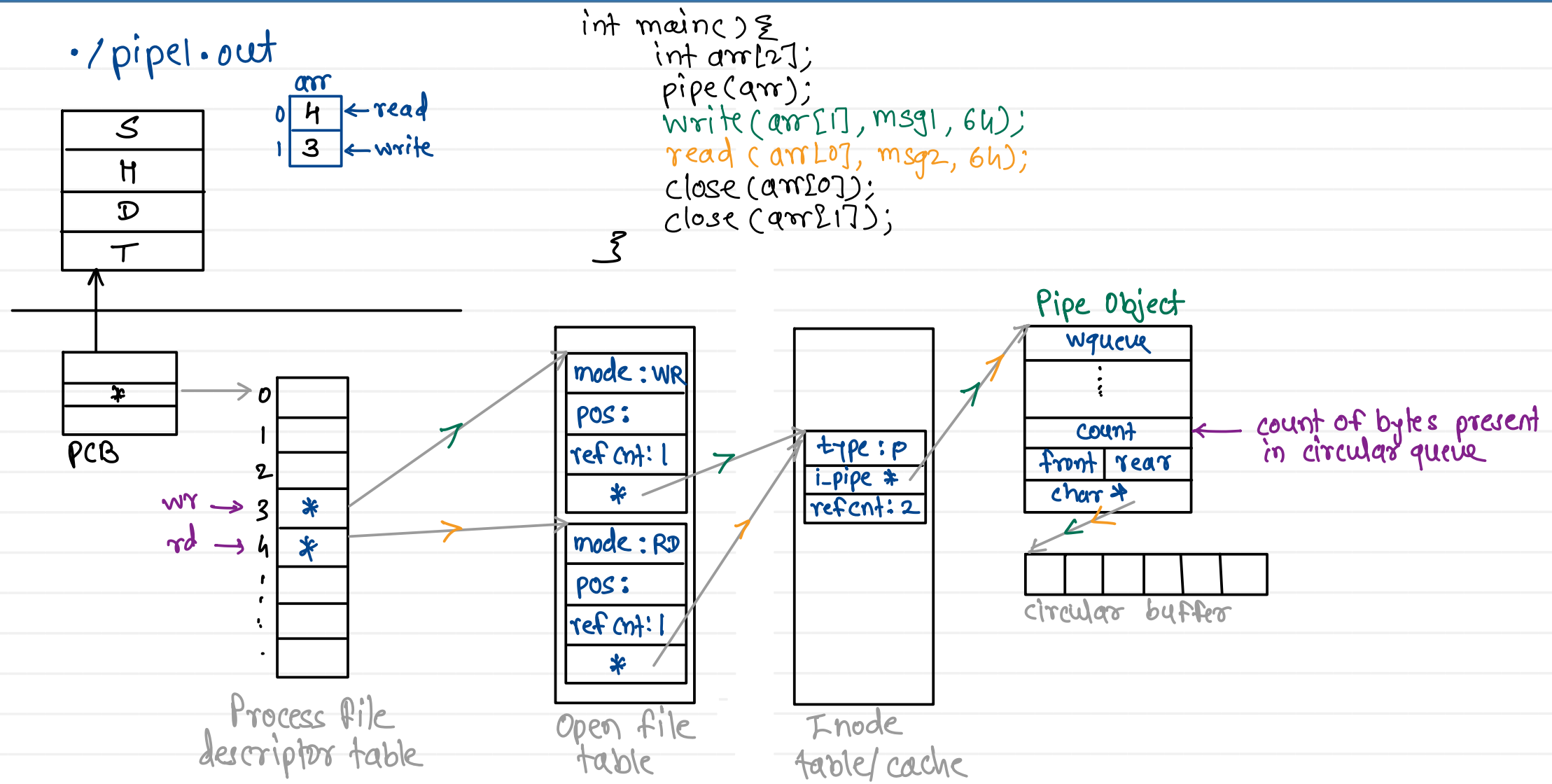
- this creates pipe & returns two end of pipe as file descriptor in arr

arr[0] ← fd of read end
arr[1] ← fd of write end

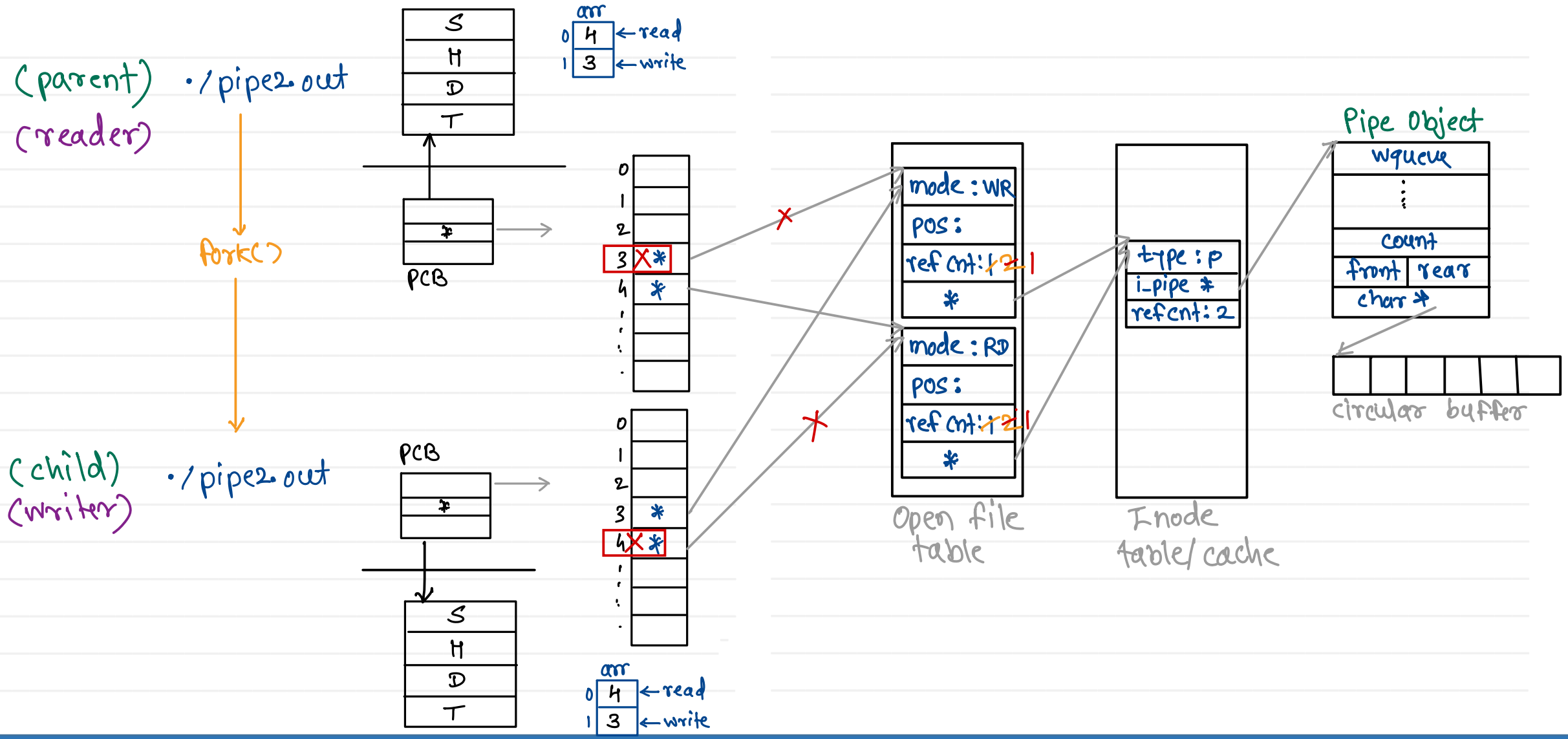
pipe

- unnamed pipe
  - if processes are related with each other
    - parent-child
    - siblings
  - (inode is created inside RAM only)
  - pipe()
- named pipe
  - if process are unrelated
    - parent of both is different.
  - (inode is created inside Harddisk partition) (with no data blocks)
  - mkfifo / mkfifo()

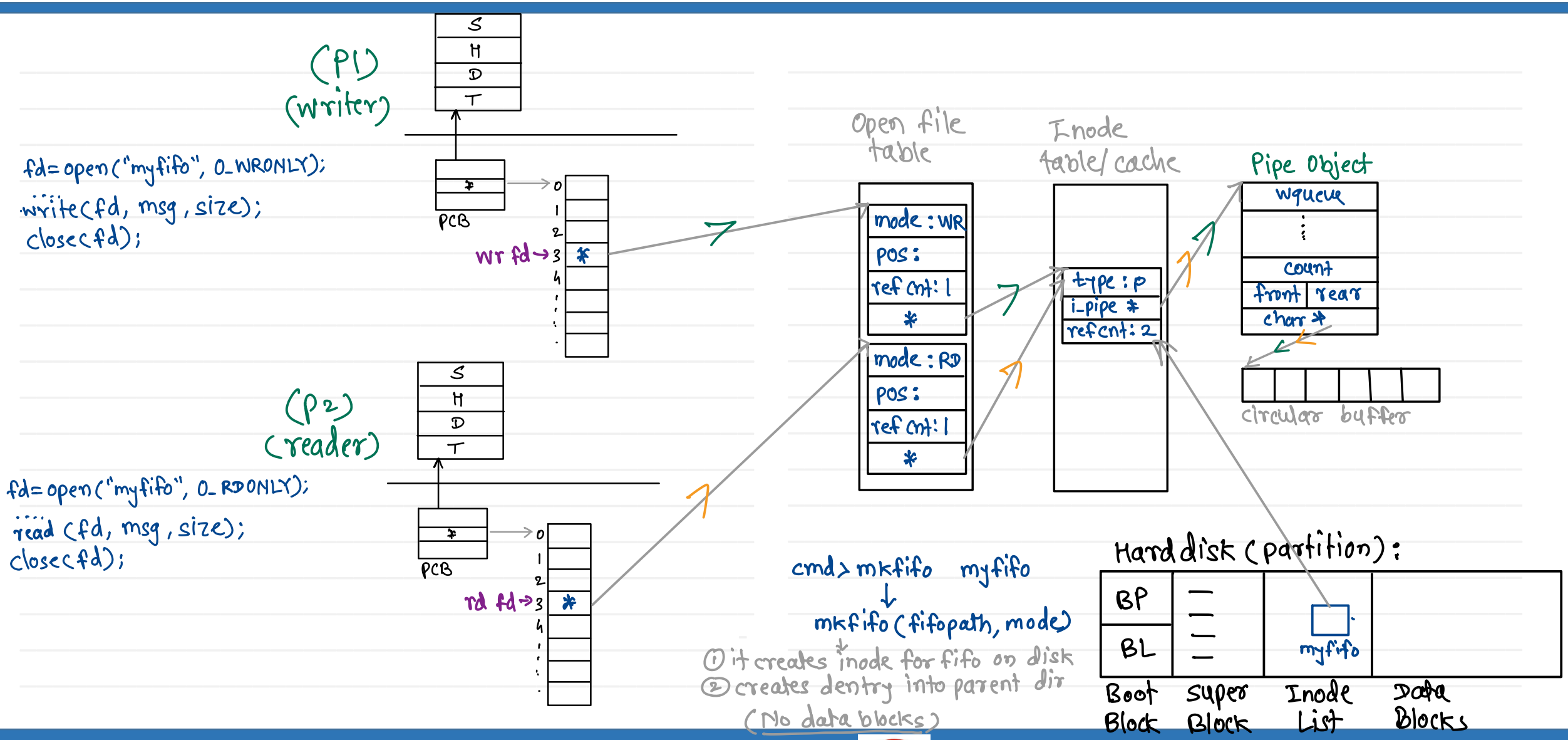
Unnamed pipe



Unnamed pipe



Named pipe


(P1)
(writer)
S
H
D
T
fd=open("myfifo", O_WRONLY);
write(fd, msg, size);
close(fd);
PCB
wr fd→
(P2)
(reader)
fd=open("myfifo", O_RDONLY);
read(fd, msg, size);
close(fd);
rd fd→
Open file table
Inode table/ cache
Pipe Object
mode : WR
pos :
ref cnt: 1
mode : RD
type : p
i_pipe *
refcnt: 2
wqueue
count
front
rear
char *
circular buffer
cmd> mkfifo myfifo
mkfifo(fifopath, mode)
① it creates inode for fifo on disk
② creates dentry into parent dir
(No data blocks)
Harddisk (partition):
BP
BL
myfifo
Boot Block
Super Block
Inode List
Data Blocks

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com