SP After Mids:- 27/11/2023

Background Process Skipped.

select :-

```
int select ( int maxfd, fd_set *readset,
             fd_set *writeset, fd_set *exrset,
             struct timeval *t);
```

tv_sec tv_usec

```
int main( ) {  // infinit delay
    printf ("Hello");
    select (1, NULL, NULL, NULL, NULL); //block
}
```

will wait indefinitely

Now we create delay using select.

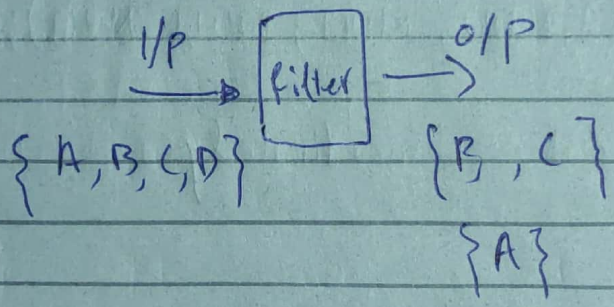
// 5 sec & 10 μsec delay

Now

```
int main () {
Struct timeval mytime;  printf("Hello \n");
mytime.tv_sec = 5;  mytime.tv_usec = 10;
int = 0  Select(1, NULL, NULL, NULL, &mytime);
    printf("world \n");
}
```

```
int main(){
    .
    .
    .
    // monitor 2 files
    .
    .
    .
int r= select(maxfd+1, &readset, NULL,
               NULL, &mytime);
printf("Remaining time = %d \t %d usec",
        mytime.tv_sec, mytime.tv_usec);
```

Filters & Redirection:-



O/P is a subset of i/p

head
tail
more
sort
grep

pipe

ls | head

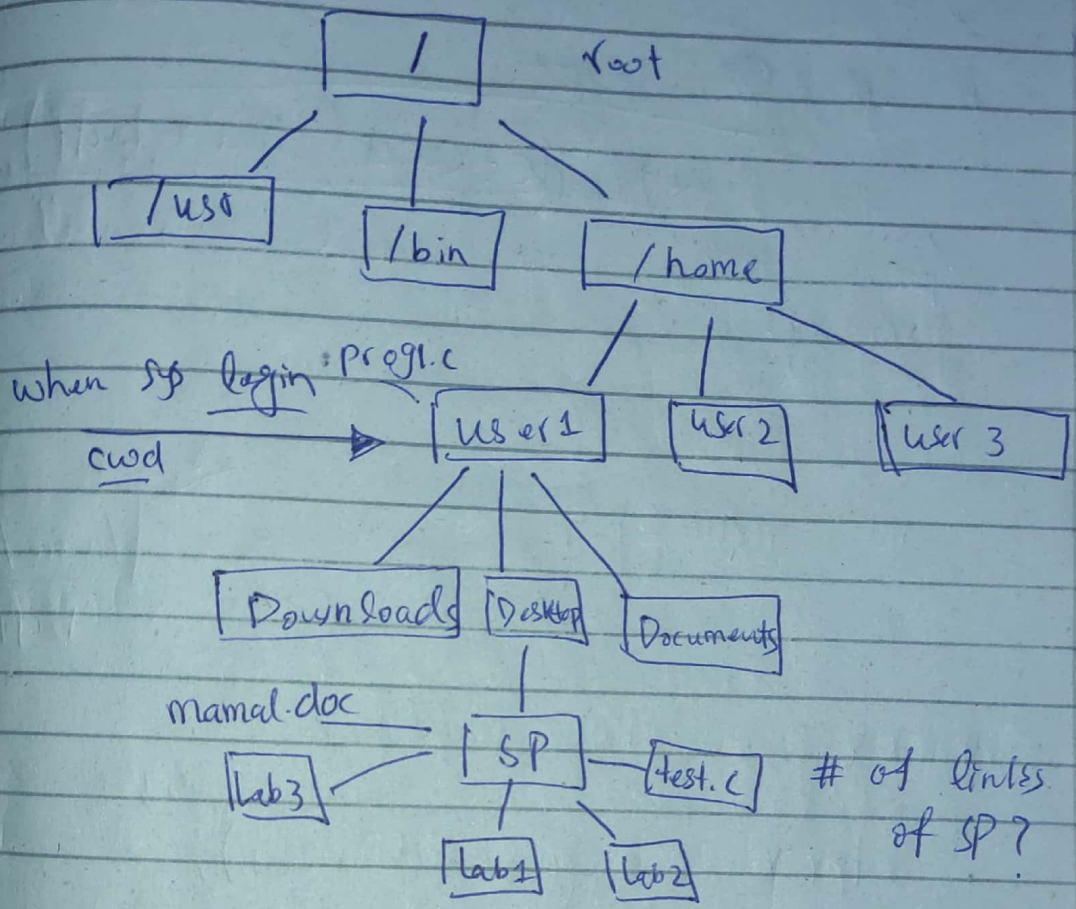
ls | tail

ls | more

Space bar for more

Chapter 5: Files & Directories

Directories → hierarchical



cd /home/usr1/Desktop/SP
cd ./Desktop/SP
cd ../usr1/Desktop/SP] relative
cd Desktop/SP

cd /bin/../home/usr1/Desktop/] absolute

Progl.c

```
main() {
        char * val = getenv("PWD");
        printf("my CWD = %s\n", val);
}
```

const char *

No need to allocate space for val.

A sys call exist.

size

```
char * getcwd(char * buff);
```

for this we need to allocate memory.

```
int main(){
```

x → /home/kesri
buff

~~char *x~~ = char buff[255];

size

char * x = getcwd(buff,);

}

Now we want to change our current dir.

```
int chdir(const char* path);
```

error -1

success 0

```
int main(){
    char buff[255];
    Char *x = getcwd(buff);
    printf("My CWD = %s\n", buff);
```

Can be absolute/relative Path.

```
int xv = chdir("Desktop/SP");
x = getcwd(buff);
printf("Path after chdir is: %s \n",
        buff);
```

Directory → A special file which contains directory entries. → dirent

- name → d_name
- inode # → d_ino

SP

name	inode #
test.c	57
manual.doc	126
Lab1	500
Lab2	1023
Lab3	1110
.	1234
..	1552
NULL	

Now we want to open SP directory

Call.

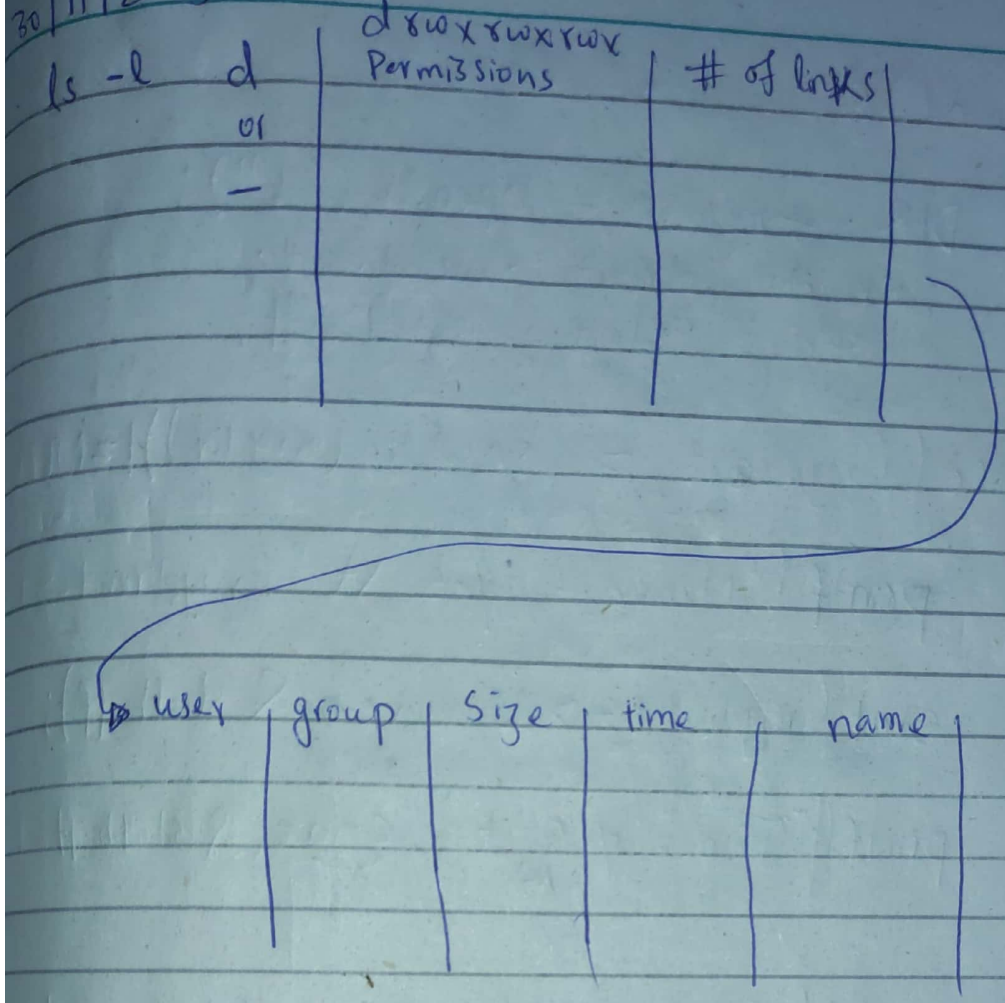
DIR * ~~char *~~ opendir (const char *dirname)

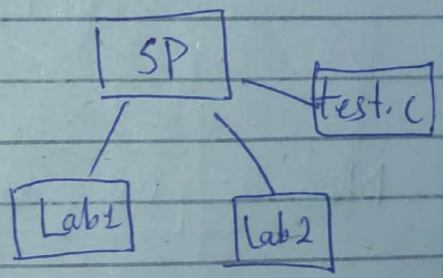
Now we want to read from SP another sys call,

struct dirent * readdir(DIR * dirp);

```
#include<dirent.h>
int main() {
        DIR *dirp = opendir("sp");
        struct dirent *direntp;
while (1) {
        direntp = readdir(dirp);
        if(direntp == NULL)
                break;
        printf("Dirent name = %s it's, dirent
          inode = %d\n, direntp=>d_name,
          direntp->d_ino);
}
```

30/11/2023





Now to get other info, a sys call exist.

<sys/stat.h>

int stat(const char *name, struct stat *statbuff);

Struct Stat

1) int st_dev
2) int st_ino
3) int st_mode
4) int st_nlink
5) int st_uid
6) int st_gid
7) time st_atime
8) time st_ctime
9) time st_mtime
10) time st_atime

```
main(){
    DIR *mydir = opendir(".");
    struct dirent *mydirent;
    struct stat statbuff;

while((mydirent = readdir(mydir))!=NULL)

    printf("Name = %s \t", mydirent->d_na

    stat(mydirent->d_name, &statbuff);

    printf("Time = %s \t", ctime(&statbuff.st_a

    printf("Size = %d \t, gid = %d \t uid = %d
        Links = %d \t", statbuff.st_size, Statbu
        st_grd, statbuff.st_uid, Statbuff.st_nlin
```

Now `int st_mode`

MACROS

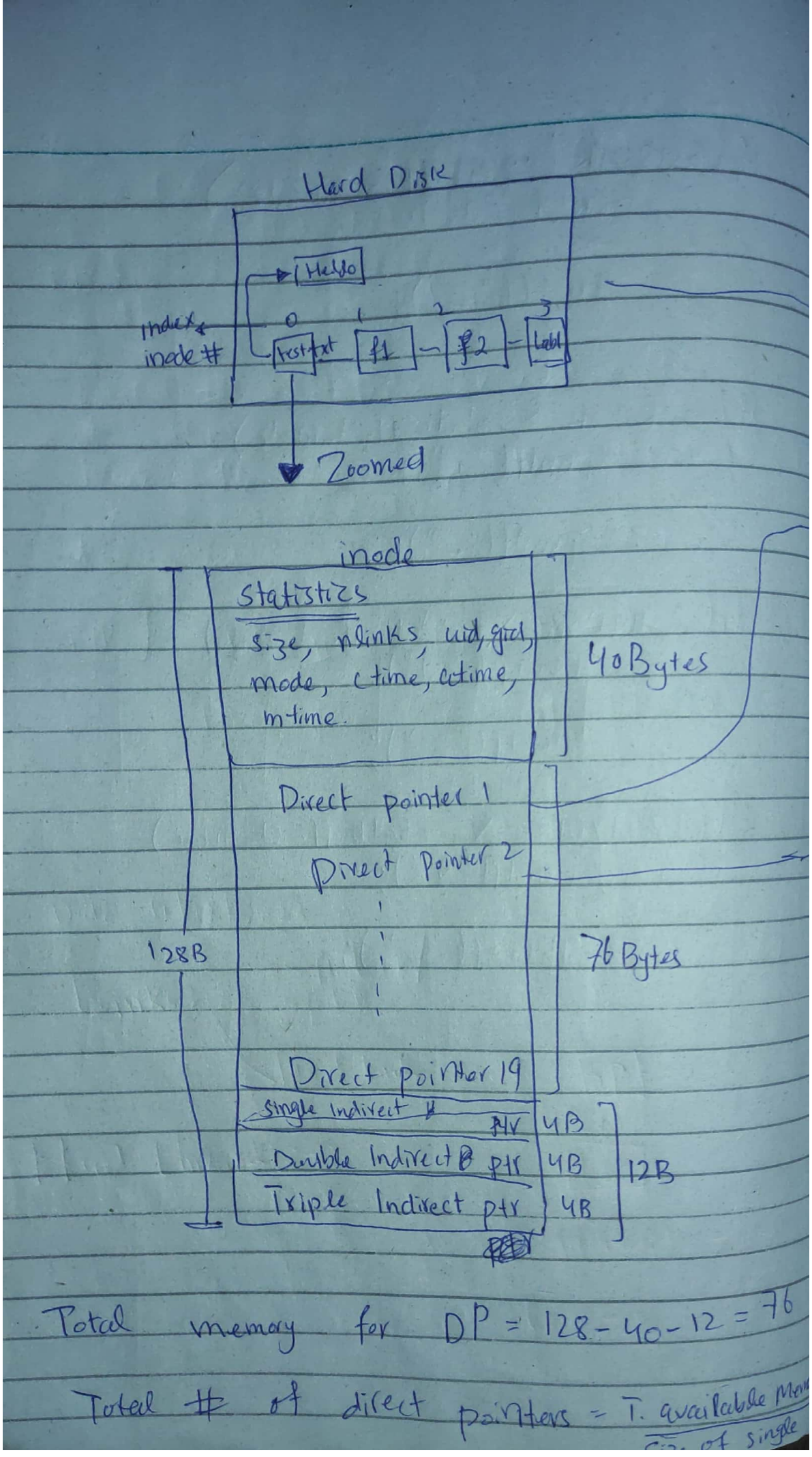
Type of file

S_ISDIR(m)
S_ISREG(m)
S_ISFIFO(m)
S_ISSOCKET(m)

Permissions

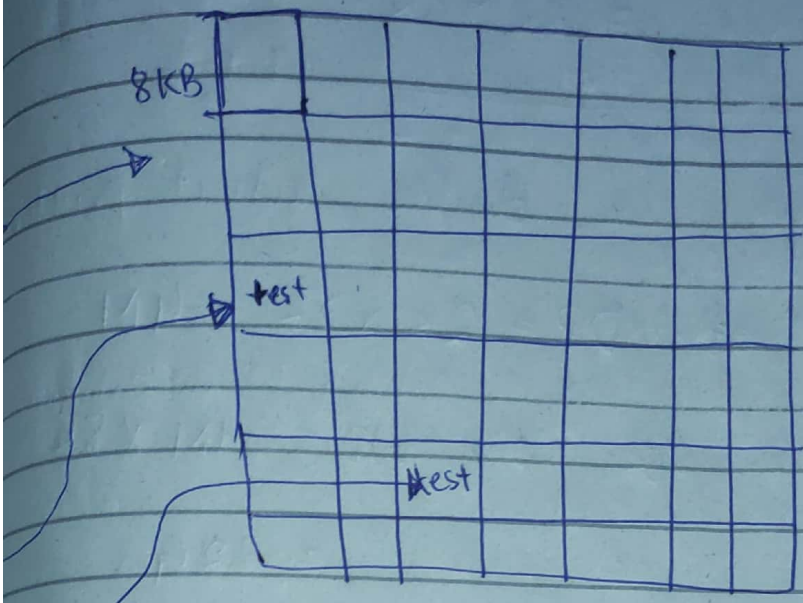
S_IRUSR

```
if(S_ISDIR(statbuff.st_mode))
        printf("d");
else
        printf("_");

if(S_IRUSR & statbuff.st_mode)
        printf("r");
else
        printf("_");

if(S_IWUSR & statbuff.st_mode)
        Printf("w");
else
        Printf("_");

if(S_IXUSR & statbuff.st_mode)
        printf("x");
else
        printf("_");
```



Total memory for DP = 128 - 40 - 12 = 76

Total # of direct pointers = T. available Mem / size of single



$= \frac{76B}{4B} = 19$ pointers

Max file using DPs = # of ptrs & size of 1 block

$= 19 \times 8KB = 152KB$

SID

Total Memory available for DP = 8KB

Total # of DP = $\frac{TAM}{Size\ of\ P} = \frac{8kB}{4B} = 2K = 2048$

Max file using SID = # of ptrs * Size of block

$= 2K * 8K = 16M$

DIP

How we use double indirect pointer

Total # of DP = $2K \times 2K = 4M$

Max file size using SID = $4M \times 8K$

// // // = 32G

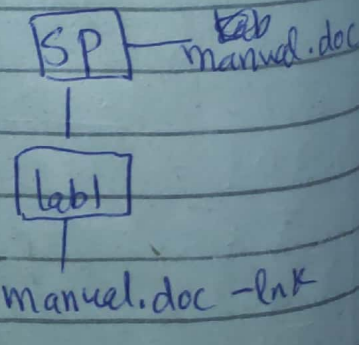
TIP Total # of DP = $2K * 2K * 2K = 8G$

Max file size using TIP = 64 TB

Date 4/12/2023

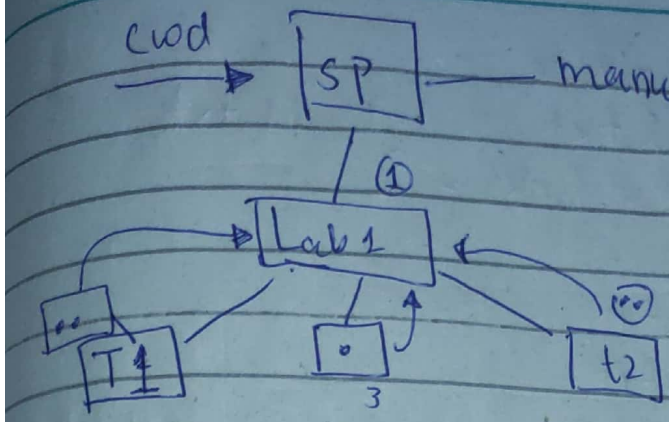
Links: — Soft Link / Symbolic
— Hard Link

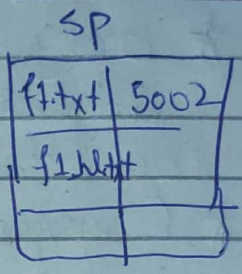
Hard Links:-



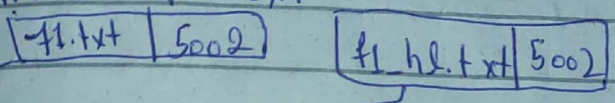
ls -l H.L

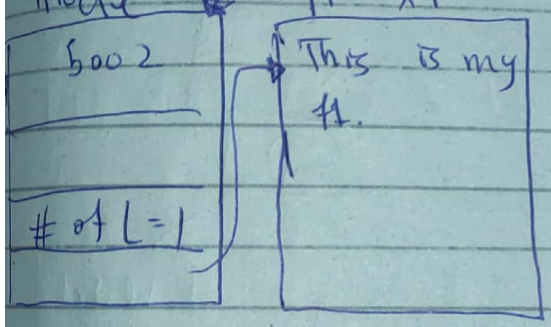
drwr--- # links

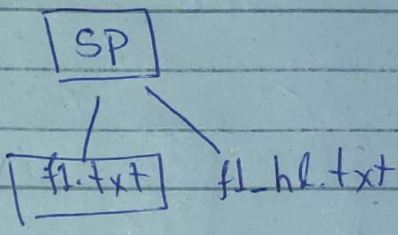




(A) Vim f1.txt







when new file created.

① inode
② Disk block
③ Directory entry

(B) Link

Command: ln f1.txt f1_hl.txt

Syscall: int link (const char *src, const char *dst);

Now ln f1.txt f1_hl.txt

When we create a hard link only directory entry is created in backend. and ... links incremented.

(C)

Now if we write

cat f1_hl.txt | cat f1.txt

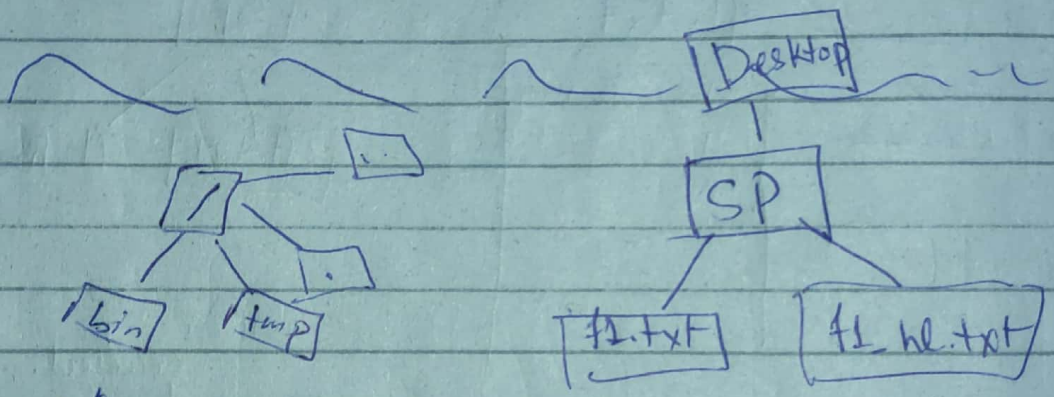
and no. of links = 2

(D) rm f1.txt

cat f1_hl.txt

★ only directory entry will be deleted and # of links will be decremented.

★ Changes will be same in both links.



ls -l

# of link	Name
2	f1.txt
2	f1_hl.txt

Soft Link :-

Command: ln -s f1.txt f1_sl.txt

Sys Call : int Symlink (const char* src, const char* dst);

① vim f1.txt

② ln -s f1.txt
f1_sl.txt

③ cat f1_sl.txt
f1.txt

Both show same context.

Now if we update the link by

④ vim f1_sl.txt

⑤ rm f1.txt

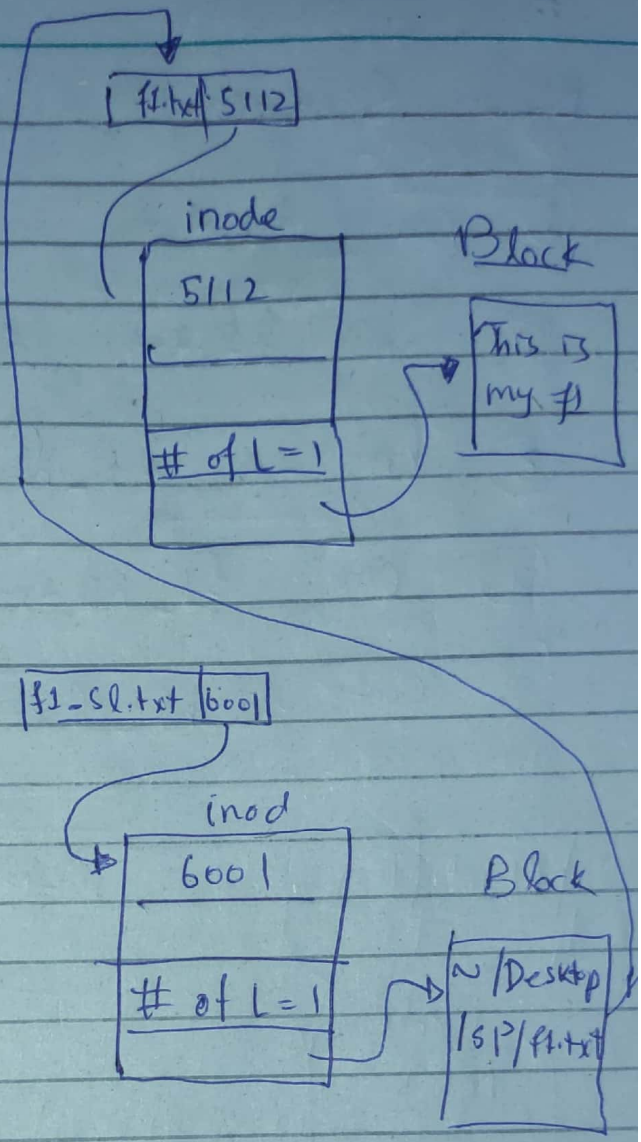
⑥ cat f1_sl.txt

↳ file doesn't exist error.

⑦ vim f1.txt

hello world

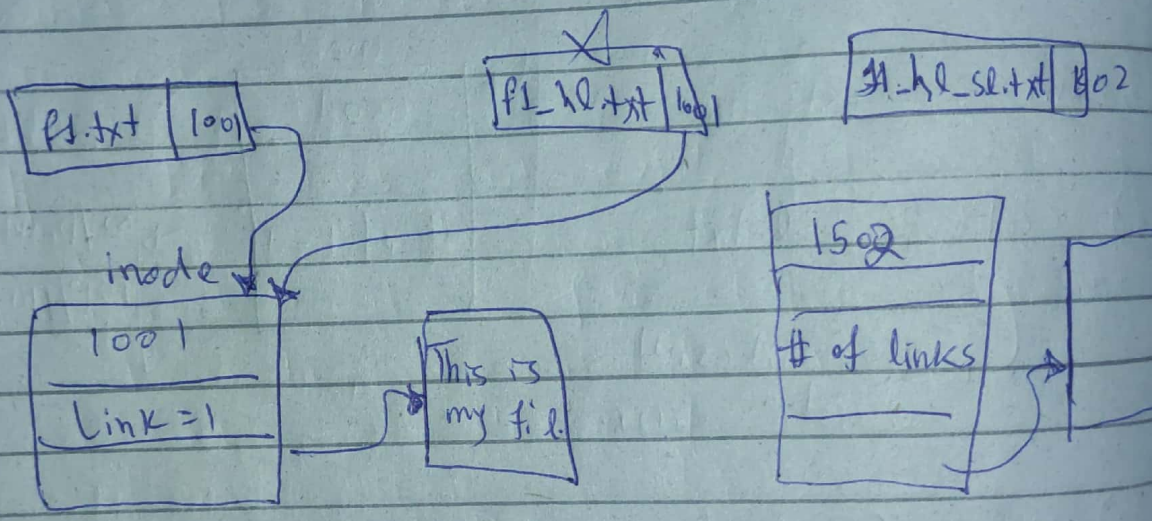
cat f1_sl.txt.
will print hello world.



It means if we create another file with same name then we can still access it through Soft link.

New Scenario:

1. vim f1.txt
2. ln f1.txt f1_hl.txt
3. ln -s f1_hl.txt f1_hl_sl.txt
4. rm f1_hl.txt
5. cat f1_hl_sl.txt

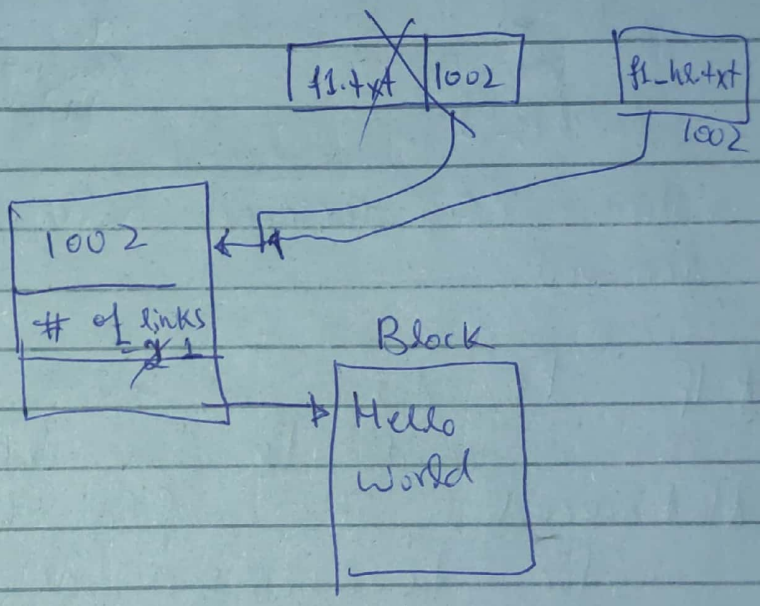


1. vim f1.txt ("Hello World")
2. ln f1.txt f1_hl.txt
3. prog.c

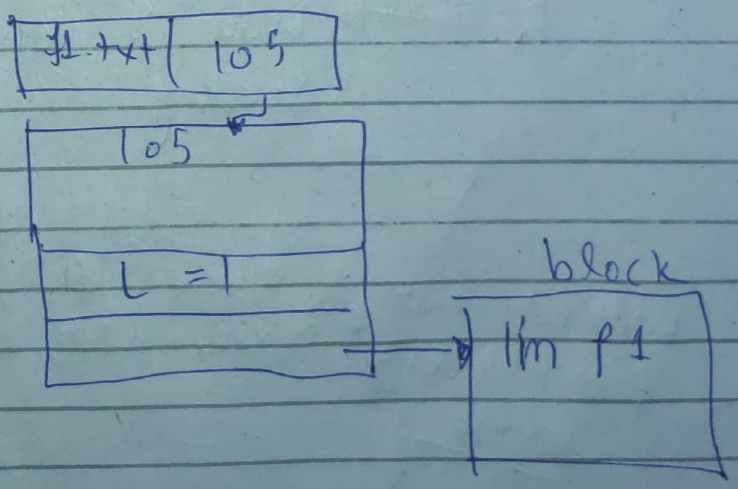
open f1.txt
read f1.txt into buff.
rm f1.txt

create f1.txt
write ("Im f1");
close.

(4)



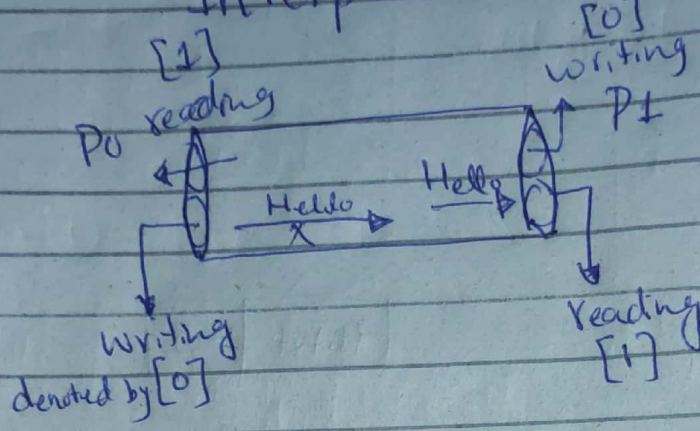
creating f1.txt



Date 7/12/2023

Pipes:-

Interprocess Communication

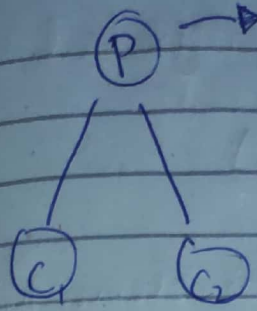


When P1 recieved hello, then it is removed

* If P1 does not exist, then Po can read it's own data.
* In Backend, it's file. (Special file).
  For two reasons it is special
  1 ⟶ Unnamed
  2 ⟶ Data is erased after reading.

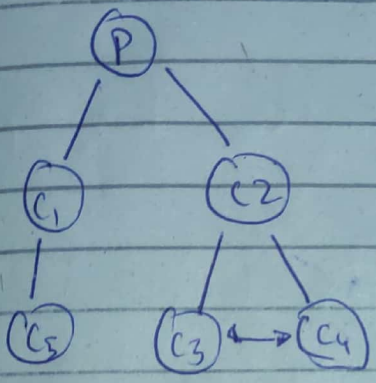
Will not wait if NONBlocking mode.

```
open("f1", O_NONBLOCK)
read(fd, buff, 100);
```



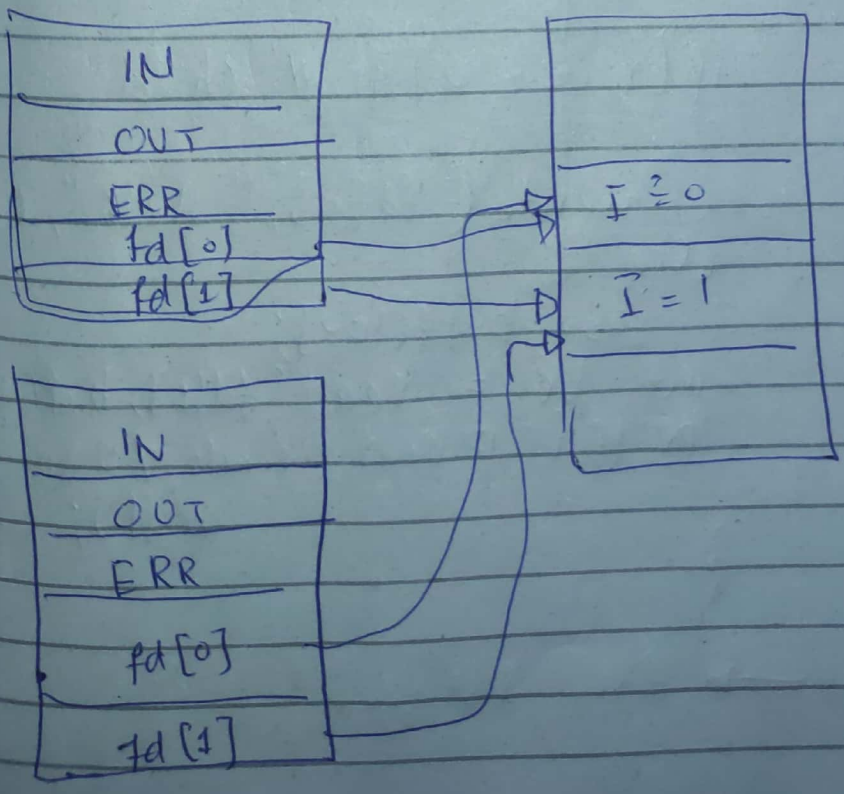
→ hierarchy top
pipe creator.

Parent child relationship must.



Parent

fd table



Now comes the pipe sys call.

int pipe(int fd[2])

success 0 ← error -1

fd[2] ← array of two elements.

Step ① Pipe creation:

```
int main(){
        int fd[2];
        int x = pipe(fd);          } Step ①
        int x = fork();
        if(x == 0) { // child
            write(fd[0], "Hello", 6);
        else {
            char buff[100];
            int br = read(fd[1], buff, 100
            printf("Parent read: %s\n",
                                buff);
        }
```

Ftp Server ⟶ file listing.

Error Case:- File Interrupt
Resources Exhaust

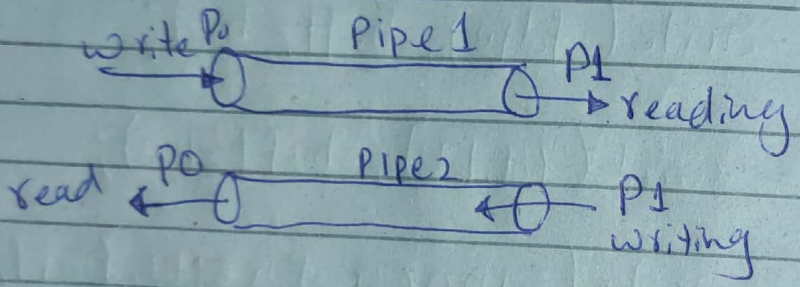
Two way Communication:-

```
int main () {
        int fd[2]; char buff[100];
        pipe(fd);
        x = fork();
if(x == 0) {

    br = read( STDIN_FILENO, buff, 100);

        write( fd[0], buff, br);

sleep
    br = read (fd [1], buff, 100);

        write(STDOUT_FILENO, buff, br);

    }

    else {

        int br = read( fd[1], buff, 100);

        writ (STDOUT_FILENO, buff ,br);

        br = read (STDIN_FILENO, buff

        write ( fd[0], buff, br );

    }
```

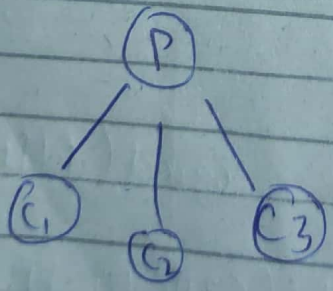
whenever there is write and then read put sleep(2);

```
write(fd[0]
sleep(2)
read(fd[1]
```

Another Solution, make two pipes.



Multiple processes.



```
int main(){
    int fd[2];
    pipe(fd);
    int N=3;

    for(i=0; i<3; i++){
        x= fork();
        if(x==0)
            break;
    }

    if(x>0){
        for(i=0; i<N; i++)
            write(fd[0], "Hello", 6);
    }
    else{
        char buff[100];
        br = read(fd[1], buff, 6);
        printf("child read %s\n", buff);
    }
```