IPC:-

Mutex - shared mem
Unnamed semaphores - shared mem
Named semaphores - any process (file path)

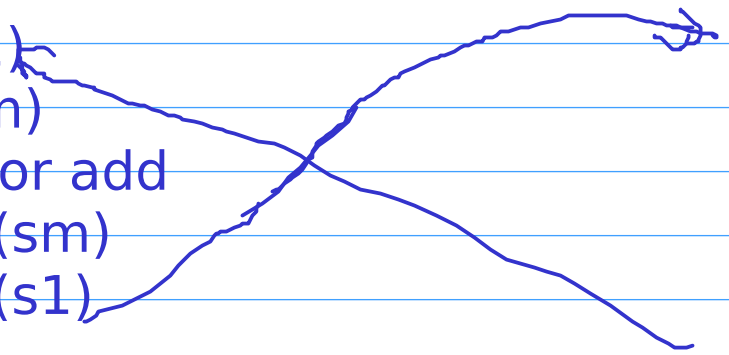
#define _GNU_SOURCE
#include<pthread.h>
(or)
gcc -D_GNU_SOURCE demo.c

gcc -DPI=22.0/7.0 demo.c

https://visualgo.net/en
data structure visualization ==> cs.usfca.edu
Stack, Queue, LinkedList
Complexity - O(1), O(n), O(logn), O(nlogn), O(n^2)

```
Semaphore sm;       sm.val=1                    //Rule-3
Sempahore s1;       s1.val=0;                   //Rule-1
Semaphore s2;       s2.val=n;             //Rule-2
```

Prod	Cons
lock(s2)	lock(s1)
lock(sm)	lock(sm)
//push or add	//pop or remove
unlock(sm)	unlock(sm)
unlock(s1)	unlock(s2)

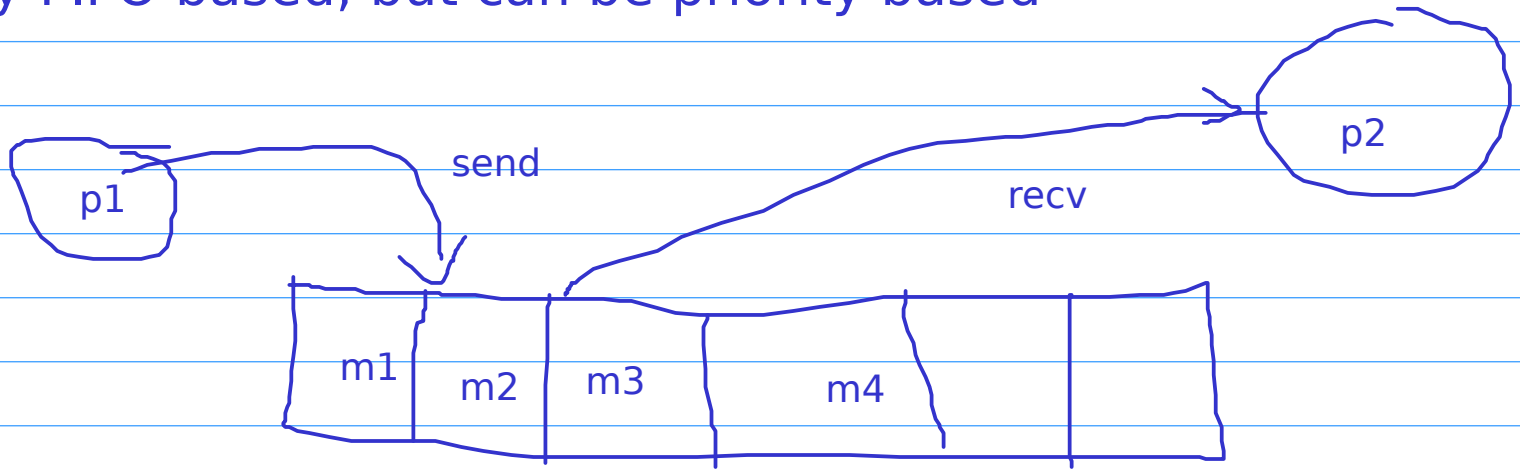


sm can be mutex instead of semaphore

Semaphores for processes (diff addr space)
* Named seamphores
* Sys V semaphores

Message Queues:-

Sender --> store messages
Receiver --> receive messages
* internal sync support
  * if there are no message (Q ie smpty) receiver will block
  * if Q is full, sender will block
* message descreter m1:10, m2:20 , one recv : m1
* Typically FIFO based, but can be priority based



POSIX Message Queues
Sys V Message Queues (later)

POSIX Message Queue:-

mq_open mq_notify
mq_send mq_setattr
mq_recv
mq_close
mq_unlink
mq_getattr

```
mqd_t mqid;

mqid = mq_open("sample", O_WRONLY);                    //O_RDONLY
(or)
mqid=mq_open("sample", O_WRONLY|O_CREAT, 0666, NULL);
(or)
struct mq_attr attr;
attr.mq_maxmsg = 10;
attr.mq_msgsize = 128;
mqid=mq_open("sample", O_RDONLY|O_CREAT, 0666, &attr);
```

sending:-

```
        char msg[]= "Hello Linux";              //strcut Student
        int len = strlen(msg);                  //sizeof(struct Student)
        int prio = 1;
        mq_send(mqid, msg, len, prio);      //mq_send(mqid, &s1, len, prio)
```

Receive:-

```
        char buf[64];
        int maxlen=64;
        int prio;
        mq_recv(mqid, buf, maxlen, &prio);
```

Close:-

```
        mq_close(mqid);
```

-lrt (or) -lpthread

ls /dev/mqueue ==>

cat /proc/sys/fs/mqueue/msg_max
cat /proc/sys/fs/mqueue/msgsize_default
mq_unlink
---
msgget, msgsnd, msgrcv, msgctl ==> Sys V Message Queues

File System:-

In UNIX/Linux (*nix) everything is a file

UART device/port ==> /dev/ttyS0

What is a file system?

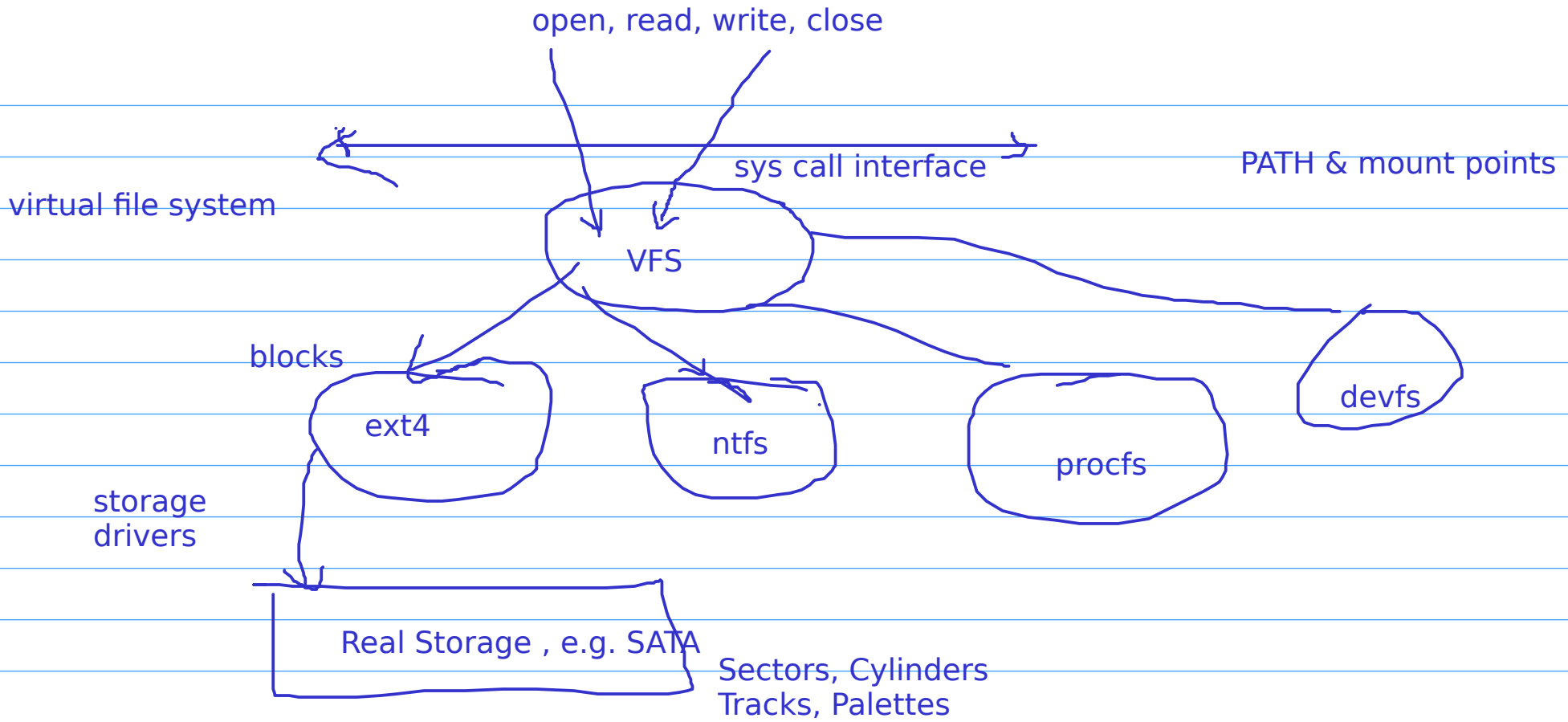
File System -- logical, Storage Device/Disk -- physical
SATA/USB/Flash/Optical/IDE

FAT32, NTFS, ext4
xfs, zfs and many more

Logical FS maps to physical storage with the
help of drivers (storage drivers)

uspace:- open, read, write, close

Making a new file system (like format)
mkfs.*



Pseudo file system , e.g. devfs, procfs, sysfs, tmpfs
pipefs, debugfs

No data on real disk, only file entries

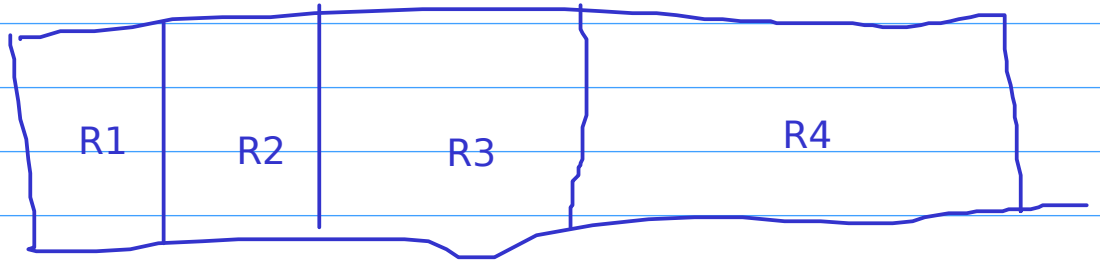
UART device/port ==> /dev/ttyS0
/proc/cpuinfo

Memory based file system - ramdisk

one partition of a disk - one file system

FS Layout:-

Boot block (R1)
Super block (R2) - meta data about filesystem
i-node blocks (R3) - meta data abt files (attributes)
data blocks (R4)

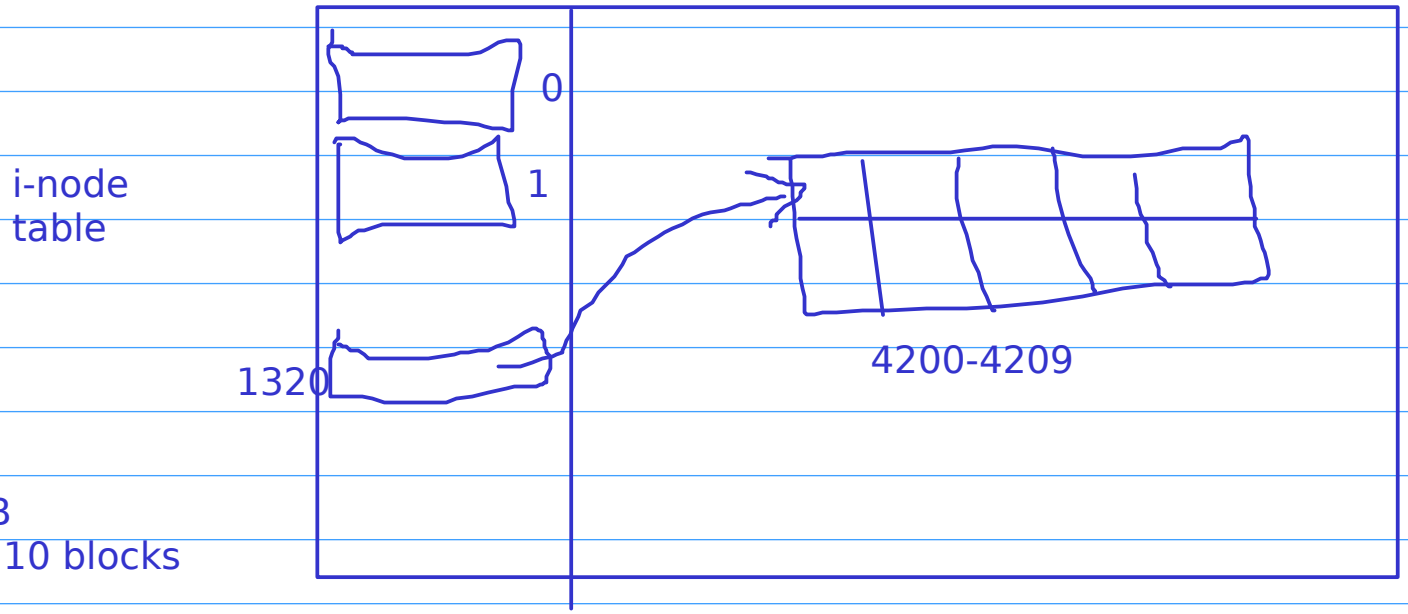


Multistage bootloader

primary bootloader (bootstrap) stored in ROM, e.g. BIOS
minimal hardware init & look for storage devices

second stage bootloader stored first few blocks of selected storage
e.g. GRUB, u-boot
last stage boot loader, locate kernel loads into memory (uncompress)
kernel will do full init of system
at end load initial process in user space, i.e. init
(based on init scripts in /etc dir and run levels)

Beginning of Disk - Master Boot Record (MBR)
Linux Boot Sequence



Block - 4 KB
40KB file - 10 blocks
sample.txt

Typical file attributes stored in i-node:-

* data block information
* mode : type + permissions
* inode ref/link count
* credentials - user, group
* size of the file (actaul, no.of blocks)
* timestamps - mtime, atime, ctime

ls -i
ls -li
ls -lid code

stat <file name>

inode ref count

cp f1 f2

rm command : unlink file name with inode
if no more links, then release i-node, data blocks

Hard links:-
ln f1 f2

file name vs inode no.s are maintained by directory

soft links / symolic links:-

Mounting:-

partitions --> primary, extended (logical)

1st ==> /dev/sda1
2nd ==> /dev/sda2 or /dev/sda5
3rd ==> /dev/sda6
4th ==> /dev/sda7 (Linux)

USB:- /dev/sdb1, /dev/sdb2, /dev/sdb3 etc
SD Card:- /dev/mmcblk0p1, /dev/mmcblk0p2

```
mkdir /mnt/c                    # mount point
# /mnt/c is blank before
mount /dev/sda1     /mnt/c
# can see sda1 (C:) content under /mnt/c

umount /mnt/c
#again blank
```

C:\samples\hello.c ==> /mnt/c/samples/hello.c

/dev/sda, /dev/sdb, /dev/mmcblk0 ==> for unpartitioned disks

sudo fdisk -l
sudo fdisk -l /dev/sda
mount

mount /dev/sdb1   /mnt/usb
umount /mnt/usb

df -kh

attach/mapping

single origin for entire FS -   /

Activity:-
* Go through FS notes, tutorials on Pipes/Fifos
* Any tutorials on relevant IPC concepts
* Assignments

* Please try first-steps, under kprog-drivers. username:- root
                                                no pasword