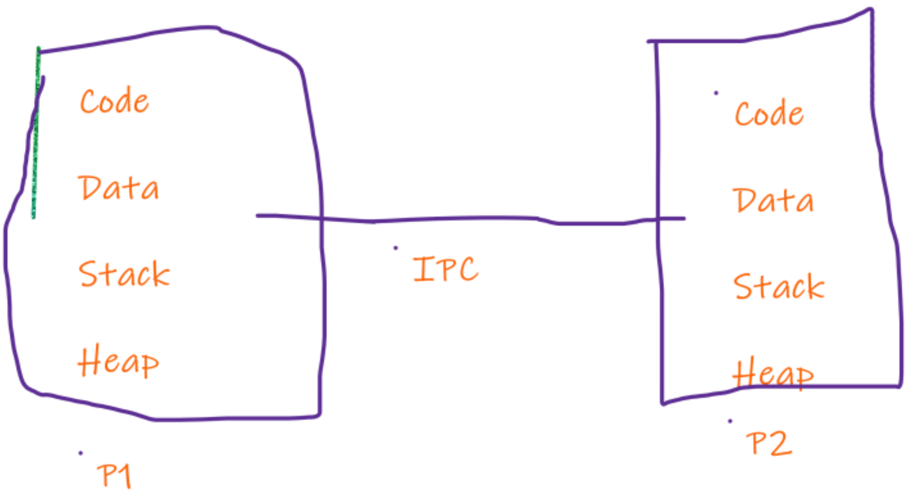
Wednesday, March 30, 2022 10:39 AM

Process -- gets an address space , process space ------- code, data , stack heap
----- is this space shared between two processes ? Or is it exclusive to each process?

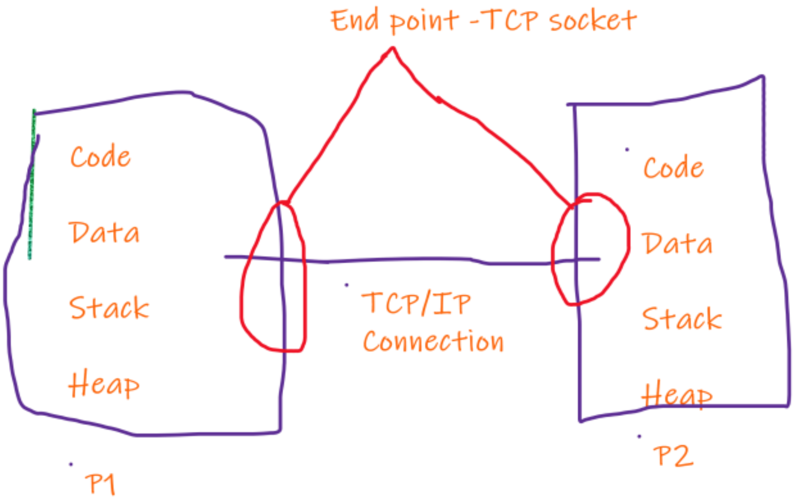
PROCESS space is exclusive to each process!!!!!

Many times the processes will have to communicate with each other in order for the application to work !!!

Inter Process Communication



Many ways to achieve inter process communication ------
The most popular way is using SOCKET communication !!!!!



Linux based OS have some famous IPC techniques ---

1. Pipes
2. Message Queue
3. Shared memory

4. Semaphores

Other ways of IPC ----
Signals

---

1. Pipes -

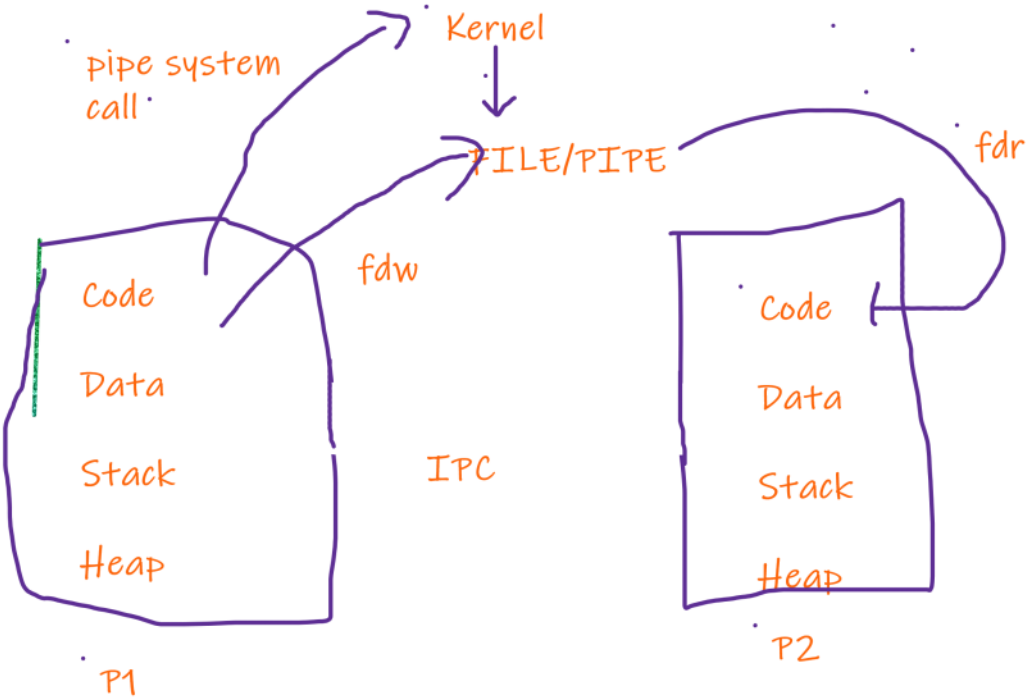
WHAT are pipes ? Pipes are a popular IPC technique

The Kernel is creating a Shared space on the Hard Disk (like a file) -------the File Descriptor (File IO using C , file is opened for reading or writing, the ID of the opened file is called as File Descriptor) of this file is given to BOTH the processes.

ONE process writes to the file and another process reads from the file - in this way they exchange information

ONE way communication

P1 ----FDw--------> Write to the PIPE -------Read From Pipe (FDr)----->P2



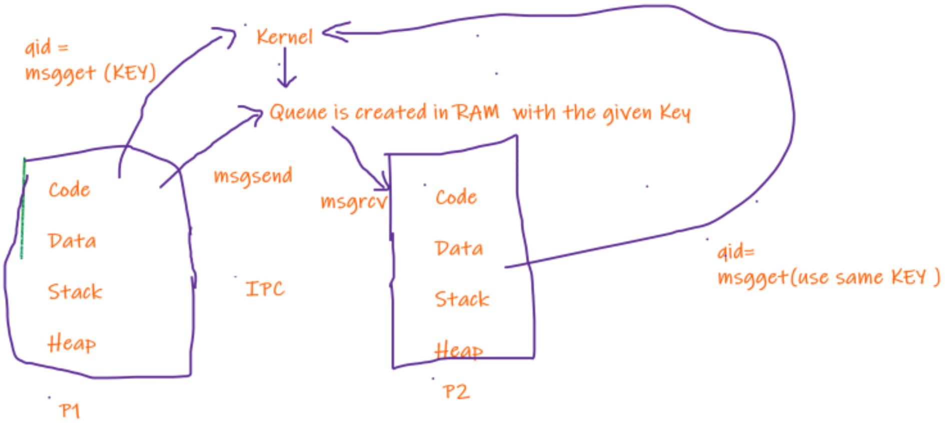
Algorithm of PIPE example

```
void main()
{
        int  file-descriptor[2] ,pid  ;
        file-descriptor  =  pipe();

        pid = fork() ;

        if ( pid > 0 )
        {
                ....
                ....
                .....
                fwrite ( file-descriptor[0]  ,  pass the data to be
                shared )  -- write to the PIPE
```

```
                    ….
                   ….
        }
         else
        {
                    ….
                    …
                     fread  (  file-descriptor[1] ,  address where the data should
                    be read )  --- Read from Pipe


        }
}
```

2. Message Queue ------ Structured Message is created and added to the queue at then end by one process , the second process receives the message from the front of the queue

```
struct msgbuf {
   long mtype;       /* message type, must be > 0 */
   char mtext[100];    /* message data */
};
```



Algorithm for Message Queue

First Process

```
   main  ()
  {

STEP 1  = 1 CREATE  A  KEY    key_t
```

STEP 2
qid = msgget(KEY , IPC_CREATE) -------system call , tells the kernel to create queue with given key

STEP 3 **create a message**

```
        struct msgbuf {
            long mtype;       /* message type, must be > 0 */
            char mtext[100];    /* message data */
        };

          msgbuf    mymessage
```

STEP 4 msgsnd (qid , mymessage)

```
        ..............
        ..................
        ...........
```

}

____

Second Process

```
   main ()
  {
```

STEP 1 = 1 CREATE A KEY key_t // SAME key as the first process

STEP 2
qid = msgget(KEY ,) -------system call , tells the kernel to get the queue with given key

STEP 3

```
        struct msgbuf {
            long mtype;       /* message type, must be > 0 */
            char mtext[100];    /* message data */
        };

          msgbuf    *   message
```

STEP 4 msgrcv (qid , &message) // message is received in the address

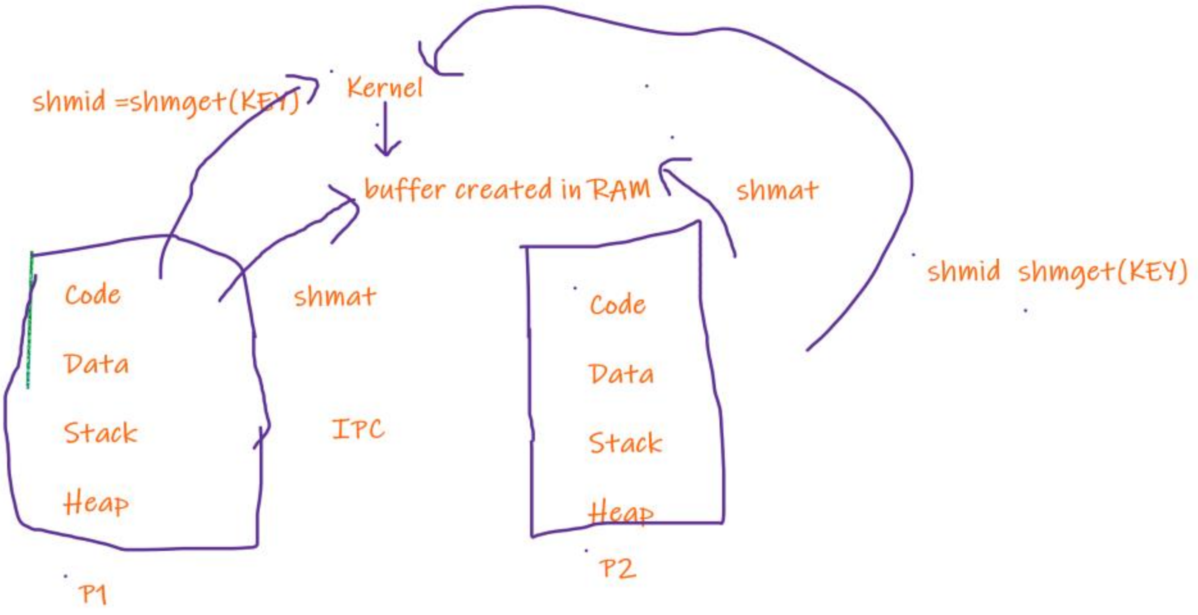
```
        ..............
        ..................
        ...........
```

Process the message !!!

}

____

3. Shared Memory IPC --- Kernel allocates a shared buffer in the RAM . The pointer to this shared memory buffer is made available to both processes . Both processes can read or write anywhere in the buffer !!!



Algorithm For P1	Algorithm for P2
main ()	main()
generate a Key	Generate a Key SAME as P1
shmid = shmget(Key, 128 , IPC_CREATE) A new space of 128 size is created with Key	shmid = shmget(Key, 128, 0) The id of the space created by P1 is received
char * ptr = shmat(shmid)	char *p = shmat(shmid)
*ptr = 'A' ; ptr ++; *ptr = 'B'	*p = 'Z' printf("%c", *p)

Pipe	Message Queue	Shared memory
File is created on HDD	Queue is created in the RAM	Buffer is created in the RAM
Unstructured messages	Structured messages	Unstructured messaged
One way - Unidirectional flow	Bidirectional	Bidirectional
pipe() and fork()	msgsend, msgrcvm , msgget are used	shmget() , shmat , shmdr
data is sent at one end and read at another---- flow	data is sent at one end and read at another---- flow	Data may be accessed for reading and writing at the same time same location SHARING data happens , this

		might lead to data sharing problems .

4 SEMAPHORES !!!!!!

____

Shell Scripting -----

SHELL = BASH = INTERPRETER

SOURCE FILE = xyz.sh

Code inside shell script can be commands that can be also run on prompt !!!

Environment Variable = variable that is **available in all programs** including kernel and user programs !!!

Type **env** on the prompt -- to see environment variables

Common environment variable ----- HOME , PWD ,
USER ,LOGNAME ,SHELL ,PS1 ,PATH

PS1 = Prompt script 1
\u = username , \h = hostname , \w = current working folder
PS2 = prompt script 2

How to set environment variable ?
export varname=varvalue

____

SHELL SCRIPT EXERCISES	
1	Show the current user who has logged in and show the home folder of current user and show content of home using HOME env variable
2	Accept a number from user and tell whether it is less than 100 or gt 100 or 100
3	Accept a number from user and show whether it is odd or even
4	Accept a name from user , if the name is not india then show you are a foreigner, else show you are indian
5	Accept 2 numbers and accept + ,- ,* , / depending on the operation entered calculate result Use if - elif
6	Use switch case , accept a month number from user - and show corresponding month name user enters 1 show January ….. If number other than 1 to 12 then show wrong month
7	On the prompt ---- ls -l , -r, -R , -a , -m , -c ,-s ,

TEST COMPARISONS	
Numerical comparisons	-lt -gt -le -ge -eq -ne
String comparisons	= , !=
File comparisons	-d , -f , -e