Chapter-6

IPC (Inter Process Communication): ১টা Process আরেকটা Process এর সাথে Communicate করা

Reasons for IPC —

i) Information Sharing – Information share করতে হলে

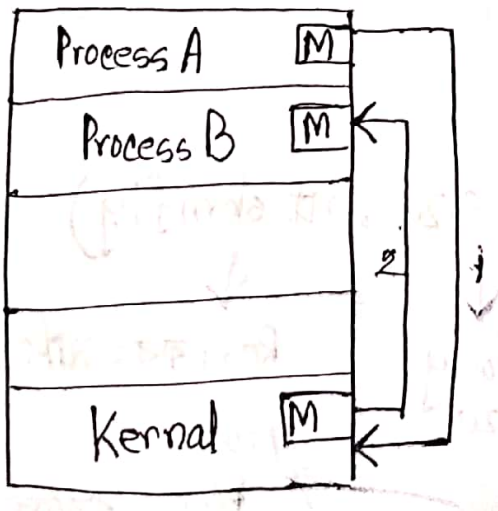
ii) Computation speedup – ১টা বড় program কয়েকটা process মিলে Run করলো।

iii) Modularity – ১টা Process এর ১টা Module ২নং ,, ,, ২নং Module.

iv) Convenience.

Two models of IPC (২ ধরনের IPC হতে পারে)

1. Shared Memory.
2. Message Passing.

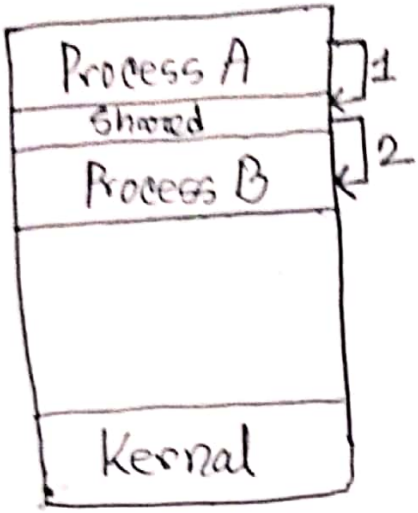


(a)

(a) Message Passing:

Process A Kernal এ message পাঠালো, Process B Kernal থেকে Message টা Read করে নিলো।

(b)



A আর B এর ভিন্ন ভিন্ন Memory. ওদের Memory-র বাইরে অন্য একটা Memory র মধ্যে জায়গার মধ্যে Memory Allocate করে দিলাম, যেটা A-ও Access করতে পারবে, B-ও Access করতে পারবে।

A এর যদি কিছু বলার থাকে Shared Memory তে Write করে দিল, B এসে Read করে নিল। Same B Write করলে A read করে যাবে।

*) shmget( ) → system call এর মাধ্যমে Shared Memory Access করতে পারব। কিন্তু Memory-র কোন জায়গা? এজন্য Key

* int shmget(key_t key, int size, int shmflg)

- key → Memory Address (কোন জায়গা?)
- size → Memory Size.
- shmflg → Read করব নাকি Write

Suppose,
A Write করলে 0
B Read করলে 1

Signals

- some signal have fixed meaning.
- some signal can be user defined.

২ধরনের { Signal এটা process ⟶ আরেকটা Process কে পাঠাতে পারে
অথবা OS-ও ⟶ " " " " "

Signal আসলে, Signal handler ওই instruction গুলো execute করে, পরে আবার যেখান থেকে আসবে।

** Some signal handler can be overriden to do other things.

⇒ কিছু কিছু signal এর ক্ষেত্রে আমি চাইলে user হিসেবে এগুলো Overwrite করতে পারি। কিন্তু কিছু signal আমি overwrite করতে পারবনা।

exit() → overwrite করতে পারবনা।

Ctrl+C → overwrite করতে পারি।

Socket

Socket এর মাধ্যমে এটা process আরেকটা process এর সাথে communicate করতে পারে।

same machine / different machine এ

Linux system এ

Unix Socket → local machine এ ২টা process communicate করতে লাগে।

TCP/UDP Socket → Different Machine এ ২টা process communication এ লাগে
(∵ তখন Networking এর Layer গুলার কথা চিন্তা করতে হবে Application, Data Link etc)

* Socket → full duplex A ↔ B (A, B ২জনই Read/Write পারবে)
* Pipe → half duplex A → B অথবা A ← B.
(A শুধু Read, B শুধু Write
অথবা A শুধু Write, B শুধু Read)

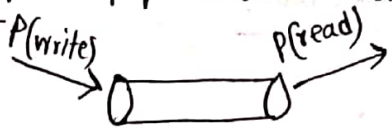
* Pipe একটা system Call. It has two file descriptors. Read handler & Write handler.

২ ধরনের pipe আছে।

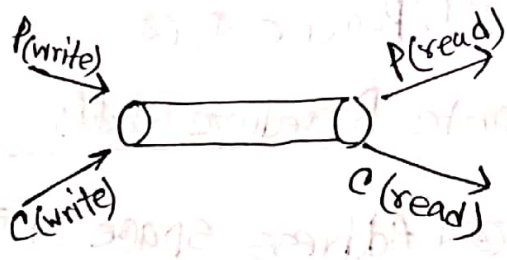
① Regular Pipes:

② Named Pipes:

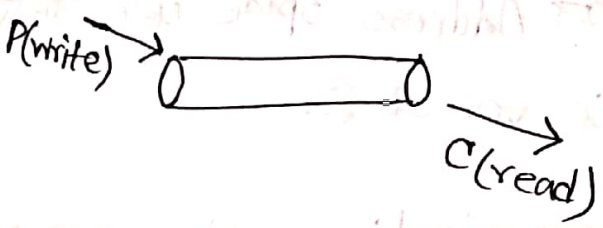
① Regular Pipe: এটা process নিজের মধ্যেই ~~Read~~ Message pass করে।



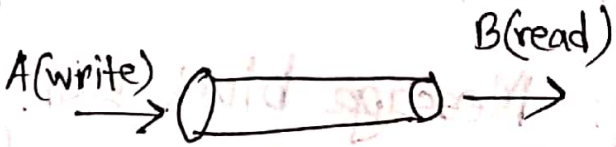
fork এর মাধ্যমে child create হলে, child ও same pipe use করবে।



এখানে P(write), C(read) করলে ঠিক আছে



② Named pipes: two endpoints of a pipe can be in different processes.



Message Queues

Process চাইলে নিজের ১টা Mailbox বানাতে পারে। Mailbox এ Message Send/Receive করবে।

Remote Procedure Calls

১টা Process নিজের Address space এ না, অন্য ১টা Address space এ Run করতে চাচ্ছে। তখন RPC call করে অন্য Process এর Address space use করতে পারি।

Client-Server এ হয়।

① Blocking communication: pipe এর buffer এ আর space নাই, full হয়ে গেছে তাই, নতুন Message টা Block হয়ে যাবে/হারায় যাবে।

② Non-blocking: Message block হবার পালাপালি, যে Process Message পাঠাচ্ছিলো Write করতে চাচ্ছিল ওকে Message পাঠাবে যে তোমার Process টা হারায় গেছে, আমি ~~Read~~ Write করতে পারিনি, তখন Message টা ওই process আবার পাঠায়।

same reading হতে গেলেও এটা হবে → buffer খালি।

Threads

CPU time scheduling করে আলাদা run করতেছে।

Thread এর সাহায্যে Multicore Programming এর problem solve হবে।

২ রকমের Thread : 1) User Threads
2) Kernal Threads