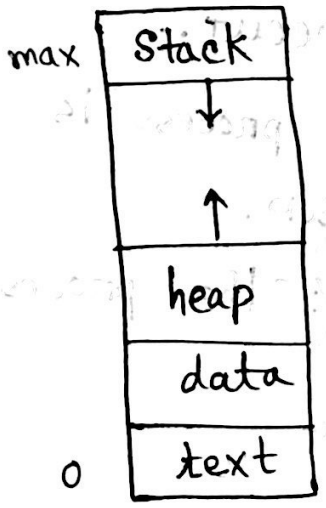
Process

Definition: Process is a program in execution where the execution must progress in sequential fashion. No parallel execution of instructions of a single process.

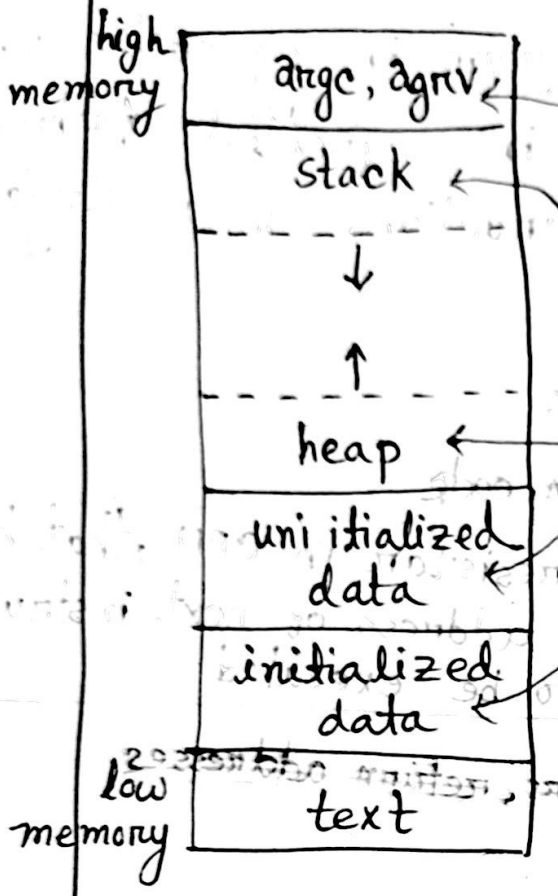
Multiple parts of process:

1. Text section → The program code
2. Program counter → is a resistant in CPU, that holds the address of next instruction to be executed
3. Stack → contain temp. data (local variables, function parameters, return addresses)
4. Data section → contain global variables
5. Heap → containing memory dynamically allocated during run time

- Program is passive entity stored in disk; process is active
- Program becomes process when executable file is loaded into memory
- A diagram of process in Memory



Memory layout in C-Program




```
#include <stdio.h>
#include <stdlib.h>
int x;
int y = 15;
int main(int argc, char *argv[])
{
    int *values;
    int i;
    values = (int *) malloc(sizeof(int)*5);
    for (i=0; i<5; i++)
        values[i] = i;
    return 0;
}
```

Process State:-

(I) **New**: The process is being created.

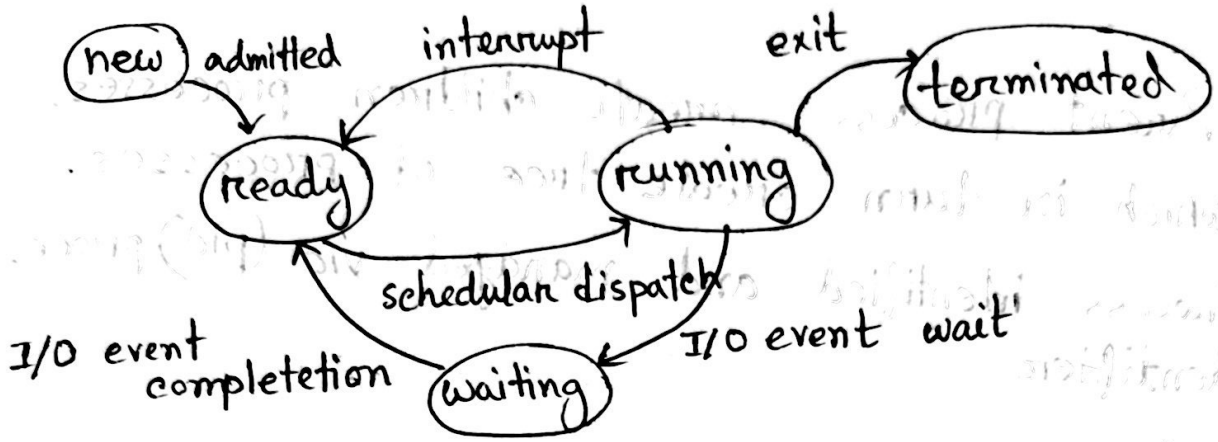
(II) **Running**: Instructions are being executed.

(III) **Waiting**: The process is waiting for some event to occur...

(IV) **Ready**: The process is waiting to be assigned to a processor.

(V) **Terminated**: The process has finished execution.

Diagram of process State



Process Control Block (PCB) → task control block ও বলা হয়।

Information associated with each process

running, waiting etc → process state

process number

location of instructions to next execute → program counter

registers

memory limits

list of open files

. . .

- CPU scheduling information – priorities, scheduling queue pointers.
- Memory management information – memory allocated to the process.
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files.

Operation on Processes

Process Creation

- Parent process creates children processes. which in turn create tree of processes.
- Process identified and managed via (pid) processor identifier

Resource sharing options

→ Parent and children share all resources
→ Children share subset of parent's resources
→ Parent and child share no resources

Execution options

→ Parent and children execute concurrently
→ Parent waits until children terminate

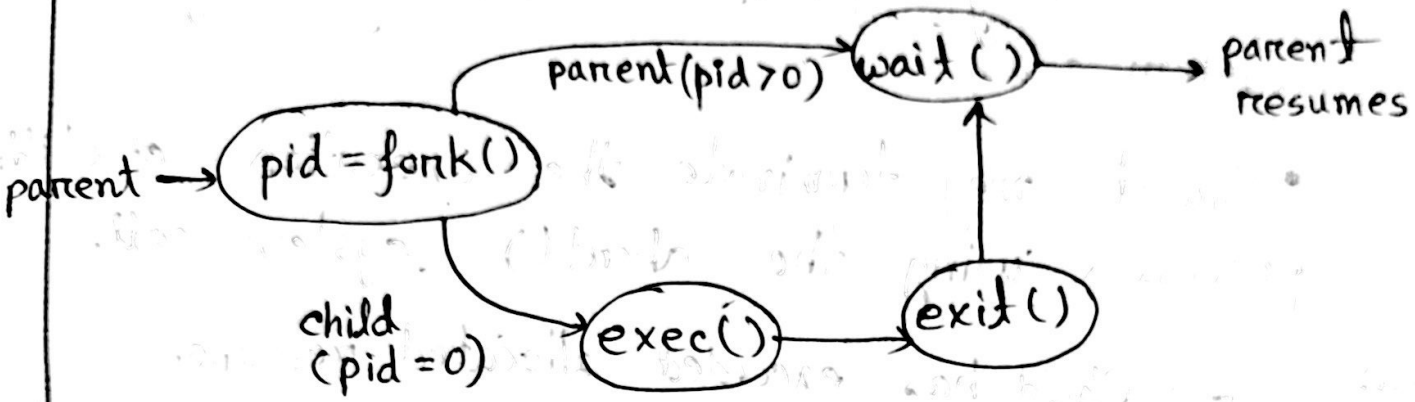
Adress Space:

→ Child duplicate of parent
→ Child has a program loaded into it

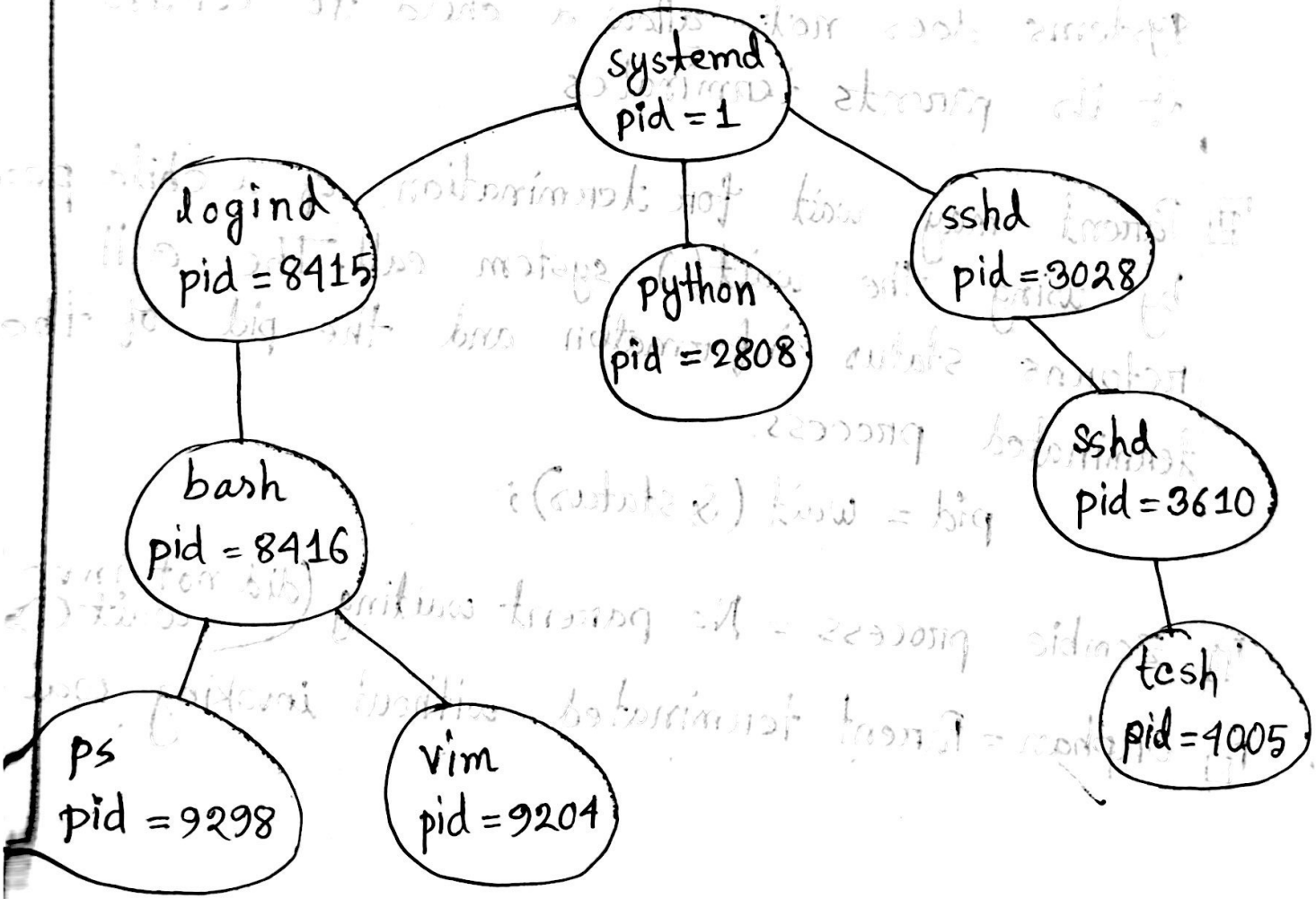
UNIX examples

→ fork() system call creates new process
→ exec() system call used after a fork() to replace the process memory space with a new program.

→ parent process calls wait() waiting ~~the~~ for the child ~~to~~ terminate



A tree of processes in Linux



Process Termination

- Process executes last statement and then asks the operating system to delete it using exit() system call.
- Parent may terminate the execution of children processes using the abort() system call.

asons:

→ Child has exceeded allocated resources

→ Task assigned to child is no longer required

→ The parent is exiting, and the operating systems does not allow a child to continue if its parent terminates

- Parent may wait for termination of a child process by using the wait() system call. The call returns status information and the pid of the terminated process.

```
pid = wait (&status);
```

- Zombie process = No parent waiting (did not invoke wait())
- orphan = Parent terminated without invoking wait()

Interprocess Communication (IPC)

Processes within a system maybe

(i) Independent → cannot affect or be affected by other processes executing in system

(ii) Co-operating → They can affect or be affected by the other processes executing in system

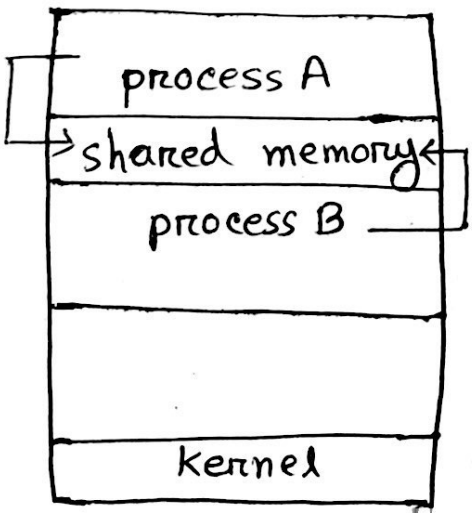
Any process that shares data with other processes is a co-operating process. It needs IPC.

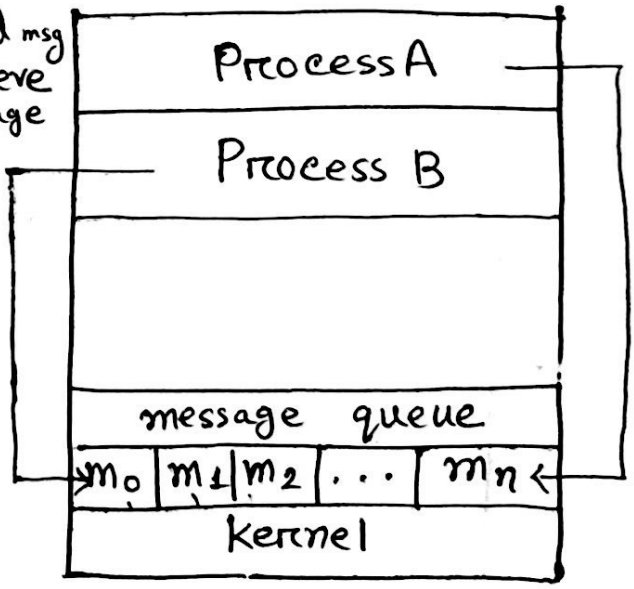
Reasons of co-operating process

(i) Information sharing
(ii) Computation speedup
(iii) Modularity
(iv) Convenience

Two models of IPC :

(i) Shared memory
(ii) Message passing [send msg, recieve message]





Message Passing

~~Context Switch~~

- Process P and Q ~~wis~~ communicate through—
  - Establishing a communication link between them
  - Exchange messages via send/receive

- Implementation of communication link

Physical :	Logical :
• Shared memory • Hardware bus • Network	• Direct or indirect • Synchronous or asynchronous • Automatic/explicit buffering

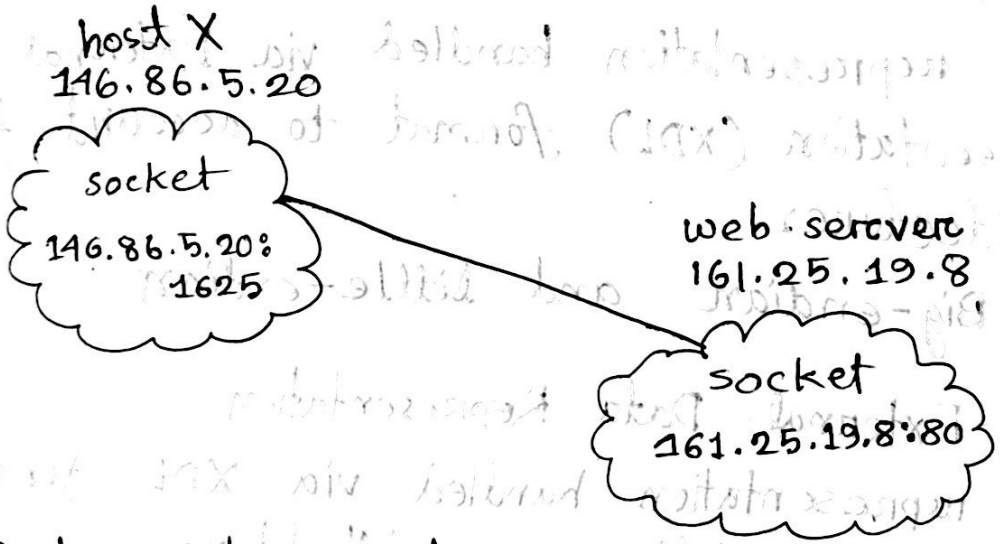
Sockets

- A socket is defined as an endpoint for communicatin
- Concatenation of IP address and port

161.25.19.8 : 1625
IP address port → port below 1024 are used for standard services.

It is a number included at start of message packet to differentiate network services on a host

Socket Communication



3 types of socket:

(i) Connection-Oriented (TCP)
(ii) Connectionless (UDP)
(iii) Multicast Socket — class-data can be sent to multiple receipents.

Remote procedure Calls

RPC abstracts procedure calls between processes on network systems

Stubs - client side proxy for the actual procedure on the server. The client-side stub locates the serve and marshalls the parameters.

- The server - side stub receives this message, unpacks the marshalled parameters and perform the procedure on the server.
- On windows, stub code compile from specification written in Microsoft Interface Definition Language (MIDL)
- Data representation handled via External Data Representation (XDL) format to account for differe architectures.
  → Big-endian and little-endian
- XDL = External Data Representation
  Data representation handled via XDL format to account for different architechtures.
  → Big-endian and little-endian
- OS typically provides a rendezvous service to connect client & server
  ↓
  matchmaker

Execution of RPC